

Universidad de las Ciencias Informáticas

Facultad 1



Solución de centralización y visualización de *logs* para Nova Servidores

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

Autor: Stephany Hernández Castillo

Tutor: Mtr. Yoandy Pérez Villazón

Ing. Anaily Navia López

Habana, junio de 2017

Solución de centralización y visualización de logs para Nova Servidores

Declaración de Autoría

Declaro por este medio que yo **Stephany Hernández Castillo**, con carné de identidad **93102607379** soy autora principal del trabajo titulado “**Solución de centralización y visualización de logs para Nova Servidores**” y autorizo a la Universidad de las Ciencias Informáticas a hacer uso de la misma en su beneficio, así como los derechos patrimoniales con carácter exclusivo. Para que así conste firmo la presente declaración jurada de autoría en La Habana a los días _____ del mes de _____ del año _____.

Stephany Hernández Castillo

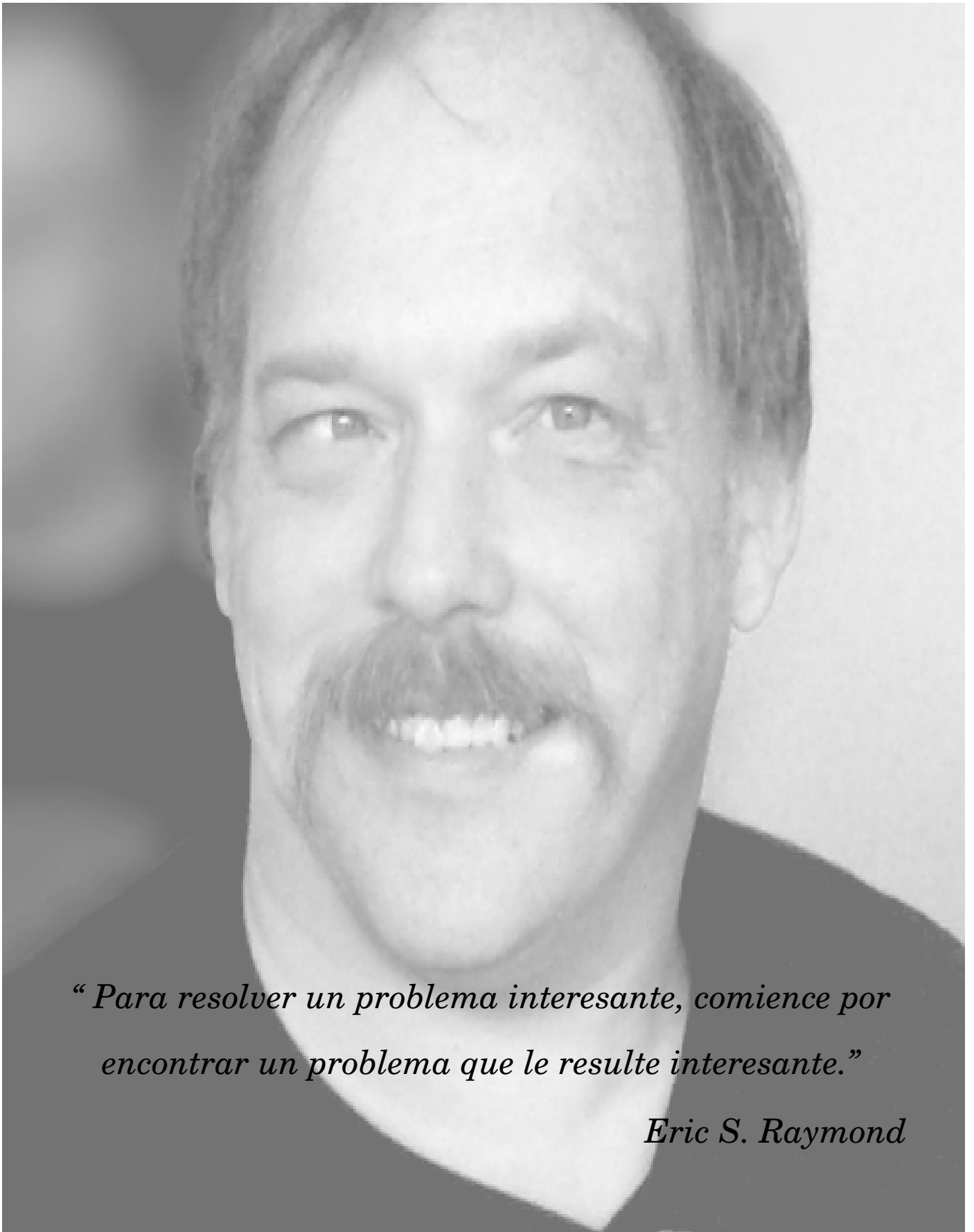
Firma de la autora

Mtr. Yoandy Pérez Villazón

Firma del Tutor

Ing. Anaily Navia López

Firma de la Tutora



“ Para resolver un problema interesante, comience por encontrar un problema que le resulte interesante.”

Eric S. Raymond

Dedicatoria

A mi hermana Melanie Hernández Castillo, mi mamá Yamilet Castillo Barranco, mi papá José Luis Hernández Acosta y mi abuelita Edelmira Acosta Vera por ser las personas más importantes de mi vida, por educarme, quererme y sobre todo por confiar en mi desde el primer día.

Agradecimientos

A mi mamá y mi papá porque gracias a ellos soy todo lo que soy, por apoyarme incluso cuando yo creía que no había salida, por quererme a pesar de no hacer siempre las cosas bien, pero sobre todo por ser los mejores padres del mundo.

A toda mi familia por estar presente en cada momento de estos 5 años.

A mis tutores que fueron mi guía durante estos últimos meses y estar conmigo justo en los momentos más importantes.

A mis tutores por obligación Yosel y Haniel por toda su entrega y dedicación y sobre todo por confiar siempre en mí.

A mis Abuelos Elia y José Luis y mis tías Ibis e Iveth que a pesar de la distancia se están graduando aquí conmigo.

A mis tías María Elena, Maira, Elena, Nené y Cary por todo su apoyo y la paciencia que han tenido conmigo en mis peores momentos.

A mi primo Jorgito que ha sido mi hermano mayor y mi ejemplo a seguir.

A Javier Piñeiro que todos sabemos que sin él no hubiese llegado ni a segundo año.

A Manuel que a pesar de no ser mi pareja en este momento ha sido durante estos últimos 5

años mi mejor amigo y mi mayor compañero y porque solo él sabe que es aguantarme en mis momentos más incómodos.

A mis amigas Rachel y Zuleimys por estar siempre juntas cuando más nos hemos necesitado.

A mi compañera de cuarto Anietsy por aguantarme 5 años sin una pelea o un desacuerdo.

A mis niños Erick, Jordan, Ronny, Braiman y Xiunelly por quererme como a una madre y estar siempre pendiente de mí.

A Elvis, Michael y Asney por ser más que mis compañeros de aula, por sus consejos y sus regaños.

A mis compañeros de aula en general por tantos y tantos momentos importantes de mi vida que pasamos juntos.

A mis amigos Eric, Ramón, Raydel y toda la gente de la facultad 2 por todas las horas de estudio y fiestas que disfrutamos.

A Alejandro por hacerme reír y llorar al mismo tiempo, pero sobre todo confiar siempre en mí.

A todas las personas que de una forma u otra hicieron posible este día.

Resumen

En una computadora se producen eventos que generan *logs*, ya sea por la acción de algún usuario o por el propio sistema. Es importante que dentro de una red sean recopilados todos estos *logs* centralizadamente para que puedan ser gestionados y visualizados detalladamente.

En la Universidad de las Ciencias Informáticas no se realiza la recolección centralizada y sistemática de los *logs* generados en las computadoras de su red y algunos servidores, perdiéndose así un gran volumen de información importante que puede ser procesada con el objetivo de detectar posibles acciones malignas. El objetivo de la investigación es desarrollar una personalización de la distribución cubana de GNU/Linux Nova en su versión para servidores que permita el almacenamiento centralizado de *logs*. Para la confección del sistema se utilizó el lenguaje Bash, mediante la herramienta Debootstrap y se tuvieron en cuenta todos los pasos y métodos que ofrece la metodología variación de AUP para la UCI. La ejecución de las pruebas permitió comprobar el correcto flujo de información entre las herramientas que conforman la solución, así como la correspondencia de esta con los requerimientos del cliente. Al culminar la investigación se obtuvo como resultado una personalización de la distribución cubana de GNU/Linux Nova en su versión para servidores que integra las herramientas Rsyslog, Elasticsearch, Logstash y Kibana facilitando la centralización de *logs* para su posterior procesamiento.

Palabras clave: centralización, GNU/Linux, *logs*, personalización.

Índice de Contenidos

Introducción	7
Capítulo 1: Fundamentación teórica	12
Introducción	12
1.1 Distribución GNU/Linux	12
1.1.1 Personalización de una distribución de GNU/Linux	12
1.2 Registros <i>logs</i>	12
1.2.1 Propósito de los <i>logs</i>	12
1.2.2 Información que almacenan los <i>logs</i>	13
1.2.3 Niveles de prioridad de los mensajes	13
1.3 Centralización de <i>logs</i>	14
1.3.1 Protocolos más utilizados en el transporte de <i>logs</i>	14
1.3.2 Herramientas para la gestión centralizada de <i>logs</i>	16
1.4 Visualización de <i>logs</i>	17
1.4.1 Herramientas para la visualización de <i>logs</i>	17
1.5 Entorno de desarrollo de la personalización	20
1.5.1 Metodología de desarrollo	20
1.5.2 Herramienta de modelado	21
1.5.3 Lenguaje de modelado	21
1.5.4 Herramienta para la creación del Sistema Operativo Base	22
1.5.5 Herramienta para la creación del ISO	22
1.5.6 Lenguaje de desarrollo	22
1.5.7 Editor de texto	23
Conclusiones del capítulo	23
Capítulo 2: Análisis y diseño de la solución propuesta	24
Introducción	24

2.1 Propuesta del sistema a desarrollar	24
2.2 Gestión del proceso de construcción de la personalización	25
2.3 Especificación de requisitos	29
2.3.1 Requisitos funcionales	29
2.3.2 Requisitos no funcionales	30
2.3.3 Historias de usuario	31
2.4 Diagrama de paquetes del análisis de Nova Servidores 6.0	34
2.4.1 Diagrama de paquetes del análisis de Nova Logs Server	37
Conclusiones del capítulo	38
Capítulo 3: Implementación y pruebas	39
Introducción	39
3.1 Descripción del proceso de construcción	39
3.1.1 Primera fase: Construcción del Sistema Operativo Base	41
3.1.2 Segunda Fase: Construcción del Sistema Operativo Final	42
3.1.3 Tercera fase: Empaquetamiento del sistema	46
3.2 Pruebas de <i>software</i>	48
3.2.1 Casos de Pruebas	50
3.2.2 Resultados obtenidos	52
Conclusiones del capítulo	53
Conclusiones generales	54
Recomendaciones	55
Referencia bibliográfica	56
Anexo 1: Historias de Usuario	59
Anexo 2: Acta de aceptación del producto	61

Introducción

Durante los últimos tiempos las Tecnologías de la Información y las Comunicaciones (TIC) se han incorporado a sectores como la administración pública y la educación. Las TIC aportan aspectos innovadores, potenciando nuevas soluciones informáticas en problemas relacionados con el desarrollo de la sociedad. El uso de *Software Libre* en las TIC ha producido relevantes impactos en cuanto a mayor flexibilidad de los procesos de producción y administración, en el aumento de la calidad de los productos y en la eficiencia de la entrega del producto a los clientes.

En el escenario actual de Cuba y el mundo, la migración al *Software Libre* constituye una necesidad para garantizar la seguridad informática que representa no solo un impacto económico, sino el logro de la independencia tecnológica como garantía del desarrollo futuro. La migración hacia plataformas GNU/Linux es de suma importancia debido a la alta disponibilidad, al aseguramiento de una mayor estabilidad, fiabilidad y seguridad, así como la reducción de los costos por licencias, tanto por el Sistema Operativo instalado como por las aplicaciones que se ejecutan en él [1].

En Cuba la migración hacia plataformas de código abierto retomó fuerza tras el acuerdo del 23 de abril del año 2015, cuando la Comisión de Informatización y Ciberseguridad indicó el desarrollo de un diagnóstico de migración en varios Organismos de la Administración Central del Estado (OACE). El diagnóstico concluyó en diciembre de ese propio año; durante el proceso fue obtenida la información sobre el *hardware* y *software* de las computadoras institucionales y de los servidores y servicios telemáticos, dando paso a su análisis y a la elaboración de informes de diagnóstico que facilitó el posterior desarrollo del proceso de migración. Los organismos más grandes que fueron sujetos a este diagnóstico muestran una compleja infraestructura de servicios telemáticos con un alto grado de concurrencia en el acceso a los servidores y servicios.

En la Universidad de las Ciencias Informáticas (UCI), el Centro de *Software Libre* (CESOL), es líder en los procesos de migración hacia tecnologías libres y de código abierto. Además, realiza acciones para alcanzar la independencia y soberanía tecnológica en el país, las cuales se caracterizan por la “posibilidad

de desarrollarse de forma autónoma en el campo de las TIC, teniendo total capacidad de decisión sobre las tecnologías y la forma en que se desarrollan y utilizan las mismas” [2]. La distribución cubana de GNU/Linux Nova es uno de los más grandes resultados de CESOL, enfocada al usuario final, con el objetivo de minimizar el cambio brusco al que se enfrentan las personas familiarizadas con sistemas *Microsoft Windows*.

Con el objetivo de realizar un uso óptimo de las redes, los administradores deben utilizar todas las herramientas disponibles. Una de las mayores fuentes de información cuando se gestionan incidentes de seguridad son los *logs*, también conocidos como registros de eventos, tanto a nivel de comunicaciones (*routers*, cortafuegos, IDS¹) como a nivel de sistema (*Unix*, *Windows*, *Linux*) y de aplicación (servidor web, servidor de correo). Desde que un usuario inicia una sesión, hasta que la cierra, en la computadora se almacenan, a modo de historial, todas las acciones realizadas mientras la sesión se mantuvo abierta, incluyendo los errores ocurridos, el tráfico capturado por un *sniffer*², las aplicaciones abiertas, mensajes de información del *kernel*³, accesos al sistema, o cambio de privilegios a alguna cuenta de usuario [3].

Gracias a la información almacenada en los *logs* se pueden tener pruebas legales de cualquier violación cometida en determinada red o incluso en una computadora específica, por lo que puede ser usada como evidencia digital ante la ley (prueba física que está construida por campos magnéticos y pulsos electrónicos que pueden ser recolectados y analizados con herramientas y técnicas especiales) [3].

La mayoría de los *logs* son almacenados en un formato estándar para que puedan ser leídos y procesados fácilmente. Para garantizar que la información esté siempre actualizada, es necesario la existencia previa de un sistema capaz de centralizarlos y visualizarlos en un único servidor, la centralización debe realizarse en tiempo real, es decir, a medida que se vayan generando los *logs* se deben ir almacenando en el servidor.

Nova, en su versión para servidores actualmente no cuenta con una forma automatizada de gestionar los

1 IDS: Sistema de Detección de Intrusos, del inglés Intrusion Detection System.

2 Sniffer: es una aplicación para redes informáticas, que permite capturar los paquetes que viajan por una red.

3 Kernel: es el *software* encargado de gestionar recursos, a través de servicios de llamada al sistema.

logs de las computadoras clientes, dificultando el análisis de la información que generan los programas instalados y el Sistema Operativo en sí. Acceder a cada estación de trabajo de forma independiente, obstaculiza la toma de decisiones, la aplicación de políticas informáticas o el análisis de comportamientos por parte de los administradores de red.

Tomando como punto de partida la situación problemática descrita se define como **problema de la investigación**: ¿Cómo almacenar y visualizar los *logs* en Nova Servidores de forma centralizada? Se define como **objeto de estudio** de la investigación, las herramientas de centralización y visualización de *logs* en entornos GNU/Linux, enmarcando como **campo de acción** los *logs* generados por estaciones de trabajo que utilicen la distribución de GNU/Linux Nova Servidores.

Para dar solución al problema de investigación se define como **objetivo general**: Desarrollar una personalización de la distribución cubana de GNU/Linux Nova Servidores que posea las herramientas y configuraciones necesarias que permitan el almacenamiento centralizado y la visualización de *logs*.

Para darle cumplimiento al objetivo general se definen los siguientes **objetivos específicos**:

1. Sistematizar sobre las herramientas y técnicas empleadas en los sistemas basados en GNU/Linux para el almacenamiento remoto de *logs* y su visualización.
2. Construir una personalización de Nova Servidores con las herramientas y tecnologías más adecuadas para la centralización y visualización remota de *logs*.
3. Evaluar la solución propuesta a partir de la simulación de varios entornos de pruebas.

Preguntas científicas:

1. ¿Cuáles son los presupuestos teóricos que fundamentan la personalización de Nova Servidores y las configuraciones necesarias para garantizar la centralización, almacenamiento y visualización de *logs*?
2. ¿Cuáles son las herramientas y tecnologías más adecuadas para construir la personalización de Nova Servidores que permita el almacenamiento centralizado y la visualización de *logs*?

3. ¿Qué método aplicar para la evaluación de la personalización de Nova Servidores que centralice y visualice los *logs* de forma remota?

Para darle cumplimiento a los objetivos específicos, se plantean las siguientes **tareas de investigación**:

1. Análisis bibliográfico de la gestión de *logs* a nivel mundial en *Software Libre*.
2. Selección de las aplicaciones y herramientas necesarias para la construcción de la personalización de Nova Servidores.
3. Identificación de los requisitos de la personalización para la centralización y visualización de *logs*.
4. Instalación y configuración de las herramientas para el manejo de los *logs*.
5. Definición de las estrategias de pruebas a realizar a la personalización.
6. Realización de pruebas funcionales para evaluar el funcionamiento de las herramientas de centralización y visualización de *logs*.

Para facilitar el cumplimiento del objetivo propuesto y de las tareas de investigación se emplean métodos teóricos de la investigación científica.

Métodos teóricos:

Histórico-Lógico: este método permitió conocer la evolución y desarrollo histórico de las herramientas para el almacenamiento centralizado y la visualización de *logs*. El análisis de la trayectoria de la teoría y su condicionamiento a los diferentes períodos históricos permitió la adaptación de los conceptos teóricos planteados a la puesta en práctica de una herramienta para la centralización y visualización de *logs* en la versión actual de la distribución cubana de GNU/Linux Nova Servidores 6.0.

Analítico-Sintético: este método permitió la atomización de los principales conceptos y elementos que son utilizados en la investigación para su estudio y análisis en profundidad. Se realizó un análisis general acerca de las herramientas que permiten el almacenamiento centralizado y la visualización de *logs* a nivel

mundial y en la propia Universidad de las Ciencias Informáticas, para luego sintetizar toda la información obtenida y elaborar la propuesta de solución de la investigación que responda a la problemática planteada en la misma.

El presente trabajo de diploma está estructurado en: resumen, introducción, tres capítulos, conclusiones, recomendaciones, referencias bibliográficas y anexos.

Capítulo 1: Fundamentación teórica.

En este capítulo se realiza un estudio del estado del arte referente a las herramientas existentes para el almacenamiento centralizado y la visualización de *logs*. Se describen y caracterizan la metodología, las herramientas y tecnologías empleadas en el desarrollo del sistema.

Capítulo 2: Análisis y diseño de la solución propuesta.

Este capítulo aborda los elementos del análisis y diseño de la propuesta de investigación, tomando como punto de partida las características del proceso de personalización y los elementos planteados por la metodología variación de AUP para la UCI. Se identifican las funcionalidades del sistema y se describen las historias de usuario asociadas a cada requisito.

Capítulo 3: Implementación y pruebas.

Este capítulo se enfoca en la construcción del sistema a partir de los resultados del Análisis y Diseño. Se define y aplica la estrategia de prueba para comprobar el correcto funcionamiento de la propuesta solución.

Capítulo 1: Fundamentación teórica

Introducción

En el presente capítulo se realiza un estudio sobre el estado actual de la gestión de *logs* para Nova Servidores, enfocándose en las herramientas que existen para la centralización y visualización de estos. Se definen los principales conceptos asociados al tema investigativo y se realiza un análisis de las herramientas, brindando la posibilidad de una mejor elección. Se detallan las tecnologías a utilizar para la implementación de la solución y la metodología de desarrollo de *software* empleada.

1.1 Distribución GNU/Linux

Una distribución de GNU/Linux puede ser descrita como una distribución de *software* basada en el núcleo Linux que incluye determinados paquetes para satisfacer las necesidades de un grupo específico de usuarios. Las mismas pueden contener o no aplicaciones o controladores privativos [4].

1.1.1 Personalización de una distribución de GNU/Linux

El uso de una distribución de GNU/Linux permite, la posibilidad de emplearlo o modificarlo para que sea utilizado con determinado objetivo específico, y esto se debe a que se puede contar con su código fuente. A estas modificaciones, cambios y transformaciones que se le realizan a una distribución determinada con el fin de suplir necesidades particulares se le denominan: personalizaciones [5].

1.2 Registros *logs*

Un *log* es un registro oficial de eventos durante un rango de tiempo en particular. En informática, principalmente en seguridad informática, es utilizado para registrar datos o información sobre quién, qué, cuándo, dónde y por qué un evento ocurre para un dispositivo en particular o aplicación [6].

1.2.1 Propósito de los *logs*

Si un *log* tiene la capacidad de almacenar los eventos que ocurren en cierto dispositivo o aplicación, entonces el objetivo de un *log* es proveer a los profesionales de la seguridad informática la habilidad de monitorear las actividades realizadas por quien haya generado el fichero. Revisando los *log*, se puede prevenir ataques informáticos y tomar las decisiones necesarias para corregir problemas de funcionamiento.

1.2.2 Información que almacenan los logs

El sistema de *logs* se encarga de recoger los mensajes generados por los programas, aplicaciones y demonios y enviarlos a un destino predefinido, generalmente a la carpeta `/var/log`, aunque existen aplicaciones que crean su propia carpeta de *logs*. En cada mensaje consta la fuente (el programa que generó el mensaje), la prioridad (nivel de importancia del mensaje), la fecha y la hora [7].

Existen procesos informáticos que se ejecutan en segundo plano en lugar de ser controlados directamente por el usuario, a este tipo de proceso se le denomina demonio o *daemon*⁴. Estos programas se ejecutan de forma continua (infinita) y aunque se intente cerrar o matar el proceso, este continuará en ejecución o se reiniciará automáticamente. Todo esto sin intervención de terceros y sin dependencia de consola alguna [8].

El sistema de *logs* arranca con el script `/etc/init.d/syslogd`, y tiene dos demonios:

syslogd: gestiona los *logs* del sistema. Distribuye los mensajes a archivos, tuberías, destinos remotos, terminales o usuarios, usando las indicaciones especificadas en su archivo de configuración `/etc/syslog.conf`, donde se indica qué se loguea y a dónde se envían estos *logs*.

klogd: se encarga de los *logs* del *kernel*. Lo normal es que *klogd* envíe sus mensajes a *syslogd*, pero no siempre es así, sobre todo en los eventos de alta prioridad, que salen directamente por pantalla [8].

1.2.3 Niveles de prioridad de los mensajes

⁴ Daemon: Disco y Monitor de Ejecución, del inglés *Disk And Execution Monitor*.

Los niveles de prioridad de los mensajes facilitan a los administradores la forma en la que deben ser atendidos los *logs*, existen desde mensajes informativos hasta mensajes críticos del sistema, teniendo prioridad los de mayor riesgos [9]:

Debug: se utiliza para escribir mensajes de depuración. Este nivel no debe estar activado cuando la aplicación se encuentra en producción.

Info: se utiliza para mensajes informativos sobre el avance de la aplicación. Se estima que debe ser el más utilizado de todos los niveles.

Warn: se utiliza para mensajes de alerta sobre eventos que desean mantener constancia, pero que no afectan el correcto funcionamiento del programa.

Error: se utiliza para mensajes de error de la aplicación que se desean guardar, estos eventos afectan al programa, pero lo dejan seguir funcionando.

Fatal: se utiliza para mensajes críticos del sistema, generalmente después de guardar el mensaje el programa abortará.

Trance: se utiliza para mostrar mensajes con un mayor nivel de detalles que *Debug*.

1.3 Centralización de logs

Centralizar los *logs* del sistema en un único servidor, clasificarlos y almacenarlos es una buena práctica que proporciona seguridad a todas las computadoras que se encuentran conectadas a la red. Se recomienda que el servidor solo ofrezca este único servicio. Un servidor de *logs* es el encargado de almacenar los registros de eventos en un repositorio ubicado en la misma PC⁵. De este modo se garantiza que todos los *logs* queden ubicados en un punto.

1.3.1 Protocolos más utilizados en el transporte de logs

⁵ PC: Computadora Personal, del inglés Personal Computer.

Para la recolección y procesamiento de *logs* los sistemas hacen uso de algunos protocolos, a través de los cuales se logra una mayor seguridad y eficiencia en su trabajo. A continuación se muestran algunos de ellos:

Syslog: es un sistema de *logs* diseñado para la administración y control de los eventos generados tanto por el sistema operativo como por las demás aplicaciones y el *kernel*. Está formado por un emisor que se encarga de enviar mensajes a través del protocolo UDP (*User Datagram Protocol*, Protocolo de Datagrama de Usuario) en texto plano a un receptor, que es el llamado Syslogd y constituye el servidor [3].

NTP: Protocolo de Tiempo de Internet del inglés Network Time Protocol, se utiliza para sincronizar la hora de los clientes instalados en las computadoras y en los Servidores, tomando como referencia otro Servidor o fuente de tiempo (como puede ser un receptor de satélite). En los sistemas de recolección de *logs* pueden ser muy importantes ya que proporciona la coordinación necesaria entre el servidor y los clientes para el envío de los *logs*. Así existe fiabilidad en cuanto a la fecha y hora de la ocurrencia de los eventos [3].

SNMP: Protocolo Simple de Administración de Red del inglés Simple Network Management Protocol, se utiliza en la administración de redes donde se encarga de gestionar la configuración de los dispositivos conectados para el intercambio de información a través de esta. Forma parte de la familia de protocolos para internet y es no orientado a la conexión [3].

TCP: Protocolo de Control de Transmisión del inglés Transmission Control Protocol, se utiliza para la transmisión de datos por la red. Es orientado a la conexión, lo cual lo asemeja a la telefonía donde para poder hablar o transmitir datos primeramente se debe establecer la comunicación entre ambas partes [3].

UDP: se utiliza para la transmisión de datos a través de la red. Es un protocolo no orientado a la conexión, característica que lo asemeja al correo postal, donde no se necesita establecer comunicación entre el origen y el destino de los datos para enviarlos ya que estos contienen la información necesaria para encaminarse ellos mismos [3].

1.3.2 Herramientas para la gestión centralizada de logs

Como parte del amplio trabajo que se ha desarrollado a nivel mundial sobre la recolección centralizada de los *logs*, se han desarrollado varias herramientas con este fin. Algunos de los sistemas existentes se especializan en la recolección de tipos específicos de *logs*, mientras que otros son capaces de trabajar con varios formatos. Dentro de las estudiadas para realizar el almacenamiento centralizado se encuentran:

Logwatch

Debido a la gran variedad de *logs* y la cantidad de información que pueden generar, lo más prudente es que sean revisados periódicamente. Logwatch es una herramienta que se encarga de generar reportes de forma sencilla que son enviados por correo electrónico y que pueden ser leídos por el administrador [10].

Logwatch es un analizador de código abierto de registro que puede analizar y convertir los *logs* en un formato estructurado, haciendo un informe personalizable basado en sus casos y requisitos de uso. Esta herramienta se centra en producir un resumen del registro de consumo y no en el procesamiento de registros en tiempo real y el seguimiento. El motor de procesamiento de los *logs* de la herramienta es extensible, en un sentido que, si desea habilitarla para una nueva aplicación, puede escribir una secuencia de comandos de procesamiento y conectarlo con Logwatch. Una desventaja es que no incluye en su informe el tiempo detallado disponible en los *logs* originales [11].

GFI EventsManager

Esta herramienta es la encargada de recolectar los *logs* generados por todas las computadoras y servidores conectados a la red. GFI EventsManager es capaz de realizar una copia de respaldo de la información y a la vez eliminarla de todas las computadoras automáticamente. Cuenta con una base de datos centralizada en la cual se pueden realizar filtrados, análisis de toda la red de forma personalizada y enviar alertas en tiempo real vía *e-mail* a uno o más destinatarios [12].

Esta herramienta automatiza la administración de *logs* permitiendo a los administradores el seguimiento

de los sucesos generados en la red desde los dispositivos que utilicen registros *Windows* y *Syslog*. La administración de *logs* consume mucho tiempo salvo que los administradores tengan una solución que clasifique los eventos por gravedad y emita alertas sobre los más críticos. Proporciona beneficios adicionales en auditoría de sucesos haciendo más sencillo realizar análisis forense [12].

Rsyslog

Rsyslog permite no solo configurar los *logs* de forma local, sino que además acepta conexiones remotas con el fin de recibir *logs* de otros equipos y poder centralizar todas estas operaciones de logeo. La herramienta implementa el protocolo básico con una configuración flexible, es un rápido sistema de procesamiento de registros de sistema que ofrece un diseño modular de alto desempeño y niveles de seguridad apropiados. A diferencia de sus predecesores (*sysklog* y *syslog*), permite el ingreso de datos desde diversas fuentes, transformación de datos y salida de resultados hacia varios destinos. Es lo suficientemente versátil y robusto para ser utilizado en entornos empresariales y tan ligero y sencillo que permite utilizarlo en sistemas pequeños. Permite almacenar las bitácoras en archivos de texto simple o bases de datos MySQL y PostgreSQL, utiliza TCP como protocolo de transporte y provee un control detallado de formato cola de procesamiento y capacidades de filtrado en cualquier parte de los mensajes. Esta herramienta es un *syslog* enfocado en la seguridad y confiabilidad, ofreciendo soporte para diversas operaciones como la demanda de almacenamiento en búfer [13].

1.4 Visualización de logs

Analizar los archivos de *logs* de las aplicaciones es una de las tareas más importantes que se deben realizar, de ahí que se puedan detectar fallas que de otra forma son muy complicadas. La realidad es que leer los archivos de *logs* no es una tarea cómoda, pero esto se puede solucionar configurando una de las herramientas para la visualización de registros de eventos que existe en el mundo.

1.4.1 Herramientas para la visualización de logs

Stack ELK⁶

El *stack* ELK es un paquete compuesto por las herramientas de código abierto Elasticsearch, Logstash y Kibana. Estas herramientas pertenecen a la empresa Elastic y a pesar de ser proyectos independientes que pueden ser usadas por separado, juntas forman un gran equipo. Su principal función es leer y almacenar la información que se necesita y posteriormente consultarla y monitorizarla [14].

Elasticsearch

Elasticsearch es un servidor de búsqueda basado en Lucene (API⁷ de código abierto para recuperación de información). Provee un motor de búsqueda de texto completo (*full-text*), a través de una interfaz web RESTful (interfaz de programa de aplicación que utiliza peticiones HTTP⁸ para obtener datos). Básicamente ofrece un servicio de búsquedas a través de una API RESTful. Mediante peticiones HTTP se puede almacenar información de forma estructurada en Elasticsearch para que esta la indexe y poder realizar búsquedas sobre ella [14].

Logstash

Logstash es una herramienta para la administración de todo tipo de *logs* (*logs* de sistema, de servidor, de errores, de aplicación). Se encarga de recolectar, parsear y filtrar los *logs* para posteriormente almacenarlos en MongoDB, enviarlos por correo electrónico o guardarlos en Elasticsearch. Estos *logs* le pueden llegar a Logstash desde el mismo servidor o desde un servidor externo, por lo que se puede tener un servidor exclusivo para el *stack* ELK [14].

Kibana

Kibana es una herramienta analítica de código abierto que permite interactuar con la información almacenada (por Logstash) en Elasticsearch y monitorizarla [14].

⁶ ELK: Elasticsearch, Logstash and Kibana.

⁷ API: Interfaz de programación de aplicaciones, del inglés Application Programming Interface.

⁸ HTTP: Protocolo de Transferencia de Hipertexto, del inglés Hypertext Transfer Protocol.

El flujo de información/funcionamiento del *stack* es el siguiente:

1. **Logstash:** es realmente el protagonista de estas tres herramientas, se encarga de la recolección de *logs*, parsearlos y almacenarlos en Elasticsearch.
2. **Elasticsearch:** es el encargado de almacenar de forma estructurada e indexada toda la información enviada por Logstash.
3. **Kibana:** se encarga de leer los datos almacenados por Logstash en Elasticsearch para posteriormente monitorizarlos.

Telegraf, InfluxDB y Grafana

Las herramientas Telegraf, InfluxDB y Grafana tienen como propósito original la monitorización de aplicaciones y servidores a través de métricas puramente de series temporales. Pueden visualizar tendencias de uso de los diferentes elementos y asociar datos cuyo comportamiento está relacionado [15].

Telegraf: se encarga de recolectar todos los datos que se le pasen mediante el fichero de configuración, coleccionando el resultado de los outputs que ya están configurados [15].

InfluxDB: es donde Telegraf manda toda esta información, InfluxDB está especialmente diseñado para almacenar de manera eficiente una cantidad importante de información, además se pueden definir períodos de retención de la información en caso de que exista algún problema [15].

Grafana: es el Dashboard que se encarga de mostrar toda la información que InfluxDB tiene almacenado en las Bases de Datos [15].

Partiendo de que el objetivo de la investigación es centralizar y visualizar los *logs* de las computadoras conectadas a la red para disminuir el esfuerzo de los administradores, de las herramientas estudiadas para la centralización se decidió que Logwatch no se puede utilizar porque no incluye en su informe el tiempo detallado disponible en los *logs* originales. Por otra parte, la administración de los *logs* con GFI EventsManager consume mucho tiempo salvo que los administradores tengan una solución que clasifique

los eventos por gravedad y emita alertas sobre los más críticos, por lo que se decidió que la herramienta rsyslog es la más indicada para darle solución al problema de la centralización.

De las herramientas estudiadas para la visualización, se decidió que Telegraf+InfluxDB+Grafana no se debía utilizar puesto que su mayor propósito es el trabajo con métricas puramente de series temporales, específicamente el monitoreo de aplicaciones y servidores, mientras que el *stack* ELK se basa en el monitoreo o diagnóstico contra fuentes de archivos de registros, por lo que se decidió que esta última era el paquete de herramientas adecuadas para dar solución al problema de la visualización de *logs*.

1.5 Entorno de desarrollo de la personalización

Para darle respuesta al objetivo de la presente investigación se describe a continuación la metodología de desarrollo, las herramientas y tecnologías utilizadas.

1.5.1 Metodología de desarrollo

La metodología para el desarrollo de *software* es un modo sistemático de realizar, gestionar y administrar un proyecto para llevarlo a cabo con altas posibilidades de éxito. Una metodología para el desarrollo de *software* comprende los procesos a seguir sistemáticamente para idear, implementar y mantener un producto de *software* desde que surge la necesidad del producto hasta que se cumple el objetivo por el cual fue creado [16].

Para el desarrollo de la solución de centralización y visualización de *logs* para Nova Servidores se emplea la metodología definida por la UCI, resultante de una variación de la metodología ágil AUP⁹.

Variación de AUP para la UCI: al no existir una metodología de *software* universal, ya que toda metodología debe ser adaptada a las características de cada proyecto (equipo de desarrollo, recursos) exigiéndose así que el proceso sea configurable, la universidad decide hacer una variación de la metodología AUP, de forma tal que se adapte al ciclo de vida definido para la actividad productiva de la UCI. Esta metodología define 3 fases, 7 disciplinas y 11 roles [16].

⁹ AUP: Proceso Unificado Ágil, del inglés Agile Unified Process.

Fases de la metodología variación de AUP para la UCI

Inicio: durante la fase de inicio se llevaron a cabo las actividades relacionadas con la planeación del proyecto. Además, se realizó un estudio que permitió obtener información fundamental acerca del alcance del proyecto.

Ejecución: durante la fase de ejecución se pusieron en marcha las actividades que tienen que ver directamente con el desarrollo del *software*. Se realizó la especificación y aceptación de los requisitos, las historias de usuarios y se definieron los casos de pruebas a utilizar en la siguiente fase.

Cierre: durante la última fase se llevaron a cabo las pruebas de funcionalidad y aceptación que permitieron evaluar el correcto comportamiento del sistema.

Se transitaron por las siguientes disciplinas: requisitos, análisis y diseño, implementación, pruebas internas, y las pruebas de aceptación.

1.5.2 Herramienta de modelado

Visual Paradigm 8.0: es una herramienta CASE¹⁰ para el desarrollo de aplicaciones utilizando modelado UML ideal para Ingenieros de *Software*, Analistas de Sistemas y Arquitectos de sistemas que están interesados en construcción de sistemas a gran escala y necesitan confiabilidad y estabilidad en el desarrollo orientado a objetos. La herramienta también ofrece navegación intuitiva entre la escritura del código y su visualización, un ambiente visualmente superior de modelado y la sincronización de código fuente en tiempo real [17].

1.5.3 Lenguaje de modelado

UML 2.0: (Unified Modeling Language, Lenguaje de Modelado Unificado) es un lenguaje para visualizar, especificar, construir y documentar los artefactos de un sistema. Ofrece un estándar para describir los modelos, incluyendo aspectos conceptuales como procesos de negocio, funciones del sistema,

¹⁰ CASE: Herramienta de Ingeniería de *Software* Asistida por Computadora, del inglés Computer Aided *Software* Engineering.

expresiones de lenguajes de programación, esquemas de bases de datos y componentes reutilizables. UML cuenta con un conjunto de notaciones y diagramas para modelar sistemas orientados a objetos y describe la semántica esencial de lo que estos diagramas y símbolos significan [18].

1.5.4 Herramienta para la creación del Sistema Operativo Base

Debootstrap: es una herramienta que instala un sistema basado en una distribución de *software* libre dentro de un subdirectorio de otro sistema ya instalado. Este proceso no requiere un CD de instalación, basta con el acceso a los repositorios de dicha distribución. Anteriormente este proceso se realizaba de forma manual, pero gracias a la automatización que ofrece esta herramienta, ha sido implantada en los proyectos del centro desde hace algún tiempo [19].

1.5.5 Herramienta para la creación del ISO

Genisoimage: se utilizó para la creación del ISO ya que permite crear imágenes de sistemas de archivos ISO-9660 para CD-ROM y grabarse en CD o DVD utilizando un programa específico de quemado o en dispositivos USB que permitan por medio de aplicaciones grabar la imagen booteable [20].

Características [20]:

- Genera a partir de un directorio seleccionado una imagen binaria que corresponderá a un sistema de archivos ISO-9660 y/o sistema de ficheros HFS¹¹ mientras escribe en un dispositivo de bloque.
- Se debe tener en cuenta que Genisoimage no está diseñado para comunicarse directamente con el *hardware*. La mayoría de los quemadores tienen un conjunto de comandos propietarios que pueden variar de un fabricante a otro y necesita una herramienta especializada para grabar el disco.

1.5.6 Lenguaje de desarrollo

Bash 4.4: el intérprete Bash es algo más que una simple consola. Es un lenguaje interpretado de

11 HFS: Sistema de Archivo Jerárquico, del inglés Hierarchecal File System.

programación que ayuda al administrador a realizar la mayor parte de las tareas necesarias, tanto en la automatización como en el arranque del sistema. Se utiliza especialmente (no exclusivamente) en sistemas Unix. Bash es el *shell* por defecto en la mayoría de sistemas GNU/Linux y Sistemas Operativos tipo Unix [21].

1.5.7 Editor de texto

Sublime Text 3: sublime Text es un editor de textos que aporta facilidades a la hora de programar o editar código. El editor está cargado de brindar funcionalidades útiles y cómodas desde el punto de la usabilidad. Destaca por ser ligero, simple, su aspecto visual sencillo, los automatismos para realizar cambios de código y sobre todo por su gran capacidad de personalización [22].

Conclusiones del capítulo

El análisis de los principales conceptos y características relacionados con la centralización y visualización de *logs* permitió una mayor comprensión de su funcionamiento y la necesidad de implementar una solución para Nova Servidores. El estudio de las herramientas posibilitó identificar a Rsyslog y el *stack* ELK como aplicaciones a tener en cuenta en la construcción de la personalización. Se seleccionó como metodología de desarrollo variación de AUP para la UCI, Bash como lenguaje de programación, Sublime Text como editor de textos y Visual Paradigm como herramienta para el modelado.

Capítulo 2: Análisis y diseño de la solución propuesta

Introducción

Este capítulo aborda los elementos del análisis y diseño de la propuesta de investigación, tomando como punto de partida las características del proceso de la personalización y los elementos planteados por la metodología variación de AUP para la UCI. Se identifican las funcionalidades del sistema y se describen las historias de usuario asociadas a cada requisito.

2.1 Propuesta del sistema a desarrollar

En la presente investigación se propone el desarrollo de una personalización de la distribución cubana de GNU/Linux Nova en su versión para servidores con las funcionalidades que permitan centralizar, almacenar y visualizar *logs* en un único servidor. El sistema permite a los administradores acceder a los registros de todas las computadoras conectadas, incluso a los servidores que estén prestando otros servicios.

Para la centralización de los *logs* se configura la herramienta Rsyslog tanto en el cliente como en el servidor. El servidor central es el encargado de capturar y almacenar todos los *logs* generados por las computadoras y servidores de la red. Posteriormente el stack ELK se encarga de la visualización de los eventos. Logstash recolecta todos los *logs* para parsearlos y almacenarlos en Elasticsearch, donde además de estar almacenados, también son indexados y enviados a Kibana para finalmente ser leídos y monitorizados (*Ver Figura 1*).

La centralización y visualización de *logs* permite una vía rápida y muy útil de organizar, asegurar y controlar dichos archivos. El sistema propuesto proporciona facilidades en la administración de los *logs*, proveyendo de la seguridad informática la habilidad de monitorear las actividades realizadas por quien haya generado el fichero.

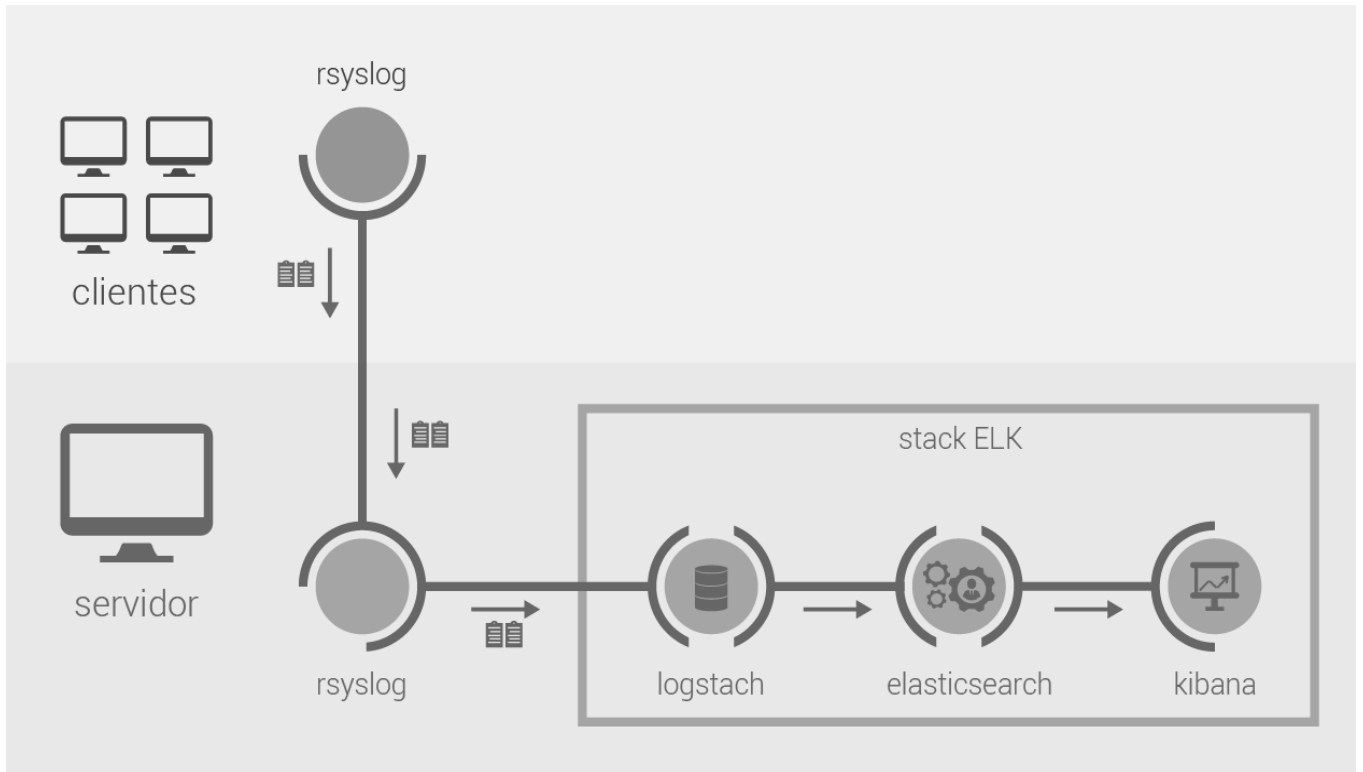


Figura 1: Representación de la propuesta de solución (Elaboración propia).

2.2 Gestión del proceso de construcción de la personalización

Para la gestión del sistema se realiza la planificación del proceso, dando como resultado un tiempo estimado de 7 meses para alcanzar el producto esperado, estableciendo un total de 14 tareas que fueron desarrolladas durante el ciclo de desarrollo. A continuación, se detallan las tareas que describen el ciclo de vida del proyecto (Ver *Tabla 1*).

Solución de centralización y visualización de logs para Nova Servidores

No	Nombre	Duración	Inicio	Terminado
1	Proceso de creación de Nova Logs Server	223 días	29/09/2016	10/05/2017
2	Recopilación de información	190 días	01/10/2016	08/04/2017
3	Capturar requisitos del cliente	15 días	01/10/2016	15/10/2016
4	Especificación y evaluación de requisitos	16 días	16/10/2016	31/10/2016
5	Definición de vista del sistema	15 días	01/11/2016	15/11/2016
6	Definición de vista de seguridad	15 días	16/11/2016	30/11/2016
7	Definición de vista de infraestructura	15 días	01/12/2016	15/12/2016
8	Definición de vista de despliegue	15 días	16/12/2016	30/12/2017
9	Creación de sistema operativo base	30 días	31/12/2017	29/01/2017
10	Instalación de paquetes	14 días	30/01/2017	12/02/2017
11	Configuración del sistema	20 días	13/02/2017	04/03/2017
12	Elaboración de manual de usuario	15 días	05/03/2017	19/03/2017
13	Empaquetar el sistema	10 días	20/03/2017	29/03/2017
14	Pruebas de aceptación	10 días	30/03/2017	08/04/2017

Tabla 1: Tareas para el desarrollo del sistema (Elaboración propia).

A continuación se muestran los riesgos asociados a las tareas correspondiente a la planificación realizada anteriormente y las acciones para mitigarlos (*Ver Tabla 2*).

Asignadas a: Stephany Hernández Castillo

Nombre	Descripción	Fecha-inicio	Fecha-fin	Riesgo	Mitigación
Recopilación de información.	-Estudio del estado del arte. -Estudio de las tecnologías para la centralización y visualización de <i>logs</i> . -Estudio de herramientas para la creación de la personalización de Nova.	01/10/2016	08/04/2017	-Información de procedencia dudosa. -Poca información para consultar.	-Definir métodos de búsqueda de información. -Gestionar correctamente la información.
Levantamiento de requisitos					
Capturar requisitos del cliente.	Se realiza una reunión con el cliente y se recopila información sobre las necesidades del mismo para con el sistema a desarrollar.	01/10/2016	15/10/2016	-Falta de requisitos indispensables para el desarrollo del sistema. -Demora o dificultades para reunirse con el cliente.	-Ser claros en las preguntas al cliente. -Planificar los encuentros con el cliente.
Especificación y evaluación de requisitos.	Siguiendo la metodología de desarrollo variación de AUP para la UCI se redacta el documento oficial que contiene los requisitos funcionales y no funcionales del sistema. (Especificación_de_requisitos_de_software)	16/10/2016	31/10/2016	-Desacuerdo por parte del cliente con los requisitos funcionales. -Poca comprensión por parte del cliente.	-Detallar claramente las acciones expuestas.
Definición de Arquitectura					
Definición de vistas del sistema.	Se definen los paquetes y componentes que conforman la solución y se les asocian los requisitos del sistema. (Arquitectura_vista_sistema)	01/11/2016	15/11/2016	-Demora en la entrega. -Puede obviarse información imprescindible.	-Detallar correctamente las acciones y definiciones expuestas. -Establecer un efectivo control de las tareas.
Definición de vistas de seguridad.	Permite presentar los diferentes métodos para garantizar la seguridad de las distribuciones GNU/Linux a partir de los requisitos a cumplir y teniendo en cuenta los diferentes niveles del producto.	16/11/2016	30/11/2016	-Demora en la entrega. -Puede obviarse información imprescindible.	-Establecer un efectivo control de las tareas.

	(Arquitectura_vista_de_seguridad)				
Definición de vistas de infraestructura.	Se definen las herramientas y tecnologías a utilizar en el desarrollo del sistema y su configuración. (Arquitectura_vista_entorno_de_desarrollo_tecnológico)	01/12/2016	15/12/2016	-Demora en la entrega. -Pueden obviarse herramientas importantes en el desarrollo.	-Establecer un efectivo control de las tareas.
Definición de vistas de despliegue.	Describe las características que deben tener las computadoras y servidores para que Nova Servidores pueda funcionar sobre ellos.	16/12/2016	30/12/2017	-Demora en la entrega.	-Establecer un efectivo control de las tareas.
Desarrollo del sistema					
Creación del sistema operativo base.	Se crea el sistema operativo base con los paquetes requeridos para la configuración.	31/12/2017	29/01/2017	-Demora en la entrega. -Puede olvidarse algún paquete requerido.	-Establecer un efectivo control de las tareas.
Instalación de paquetes.	Se instalan los paquetes necesarios para el funcionamiento del sistema operativo final.	30/01/2017	12/02/2017	-Error de dependencias.	-Buscar e instalar dependencias necesarias.
Configuración del sistema.	Se modifican los archivos de configuración necesarios para que algunas aplicaciones carguen al inicio del sistema.	13/02/2017	04/03/2017	-Demora en la entrega.	-Establecer un efectivo control de las tareas.
Elaboración del manual de usuario.	Constituye una ayuda a los usuarios para un mayor entendimiento del funcionamiento del sistema.	05/03/2017	19/03/2017	-Demora en la entrega.	-Establecer un efectivo control de las tareas.
Empaquetar el sistema.	Se empaqueta el sistema en un ISO, listo para instalarse desde un dispositivo USB, CD o DVD.	20/03/2017	29/03/2017	-No presenta.	
Pruebas al sistema					
Pruebas de aceptación.	El cliente analiza el producto y decide si satisface o no sus necesidades y si no existe ningún elemento que disminuya su calidad.	30/03/2017	08/04/2017	-No presenta.	

Tabla 2: Tareas para el proceso personalización de Nova Logs Server en función de riesgos (Elaboración propia).

2.3 Especificación de requisitos

El Glosario de Terminología Estándar de Ingeniería de *Software* define al requisito como una condición que necesita un usuario para resolver un problema o lograr un objetivo. También como una capacidad que tiene que ser alcanzada o poseída por un sistema o componente de un sistema para satisfacer un contrato, estándar u otro documento impuesto formalmente. Los requisitos pueden ser clasificados en funcionales y no funcionales [23].

2.3.1 Requisitos funcionales

Los requisitos funcionales definen funcionalidades del sistema de *software* o sus componentes. Una funcionalidad es descrita como un conjunto de entradas, comportamientos y salidas. En la Tabla 3 se muestra el listado de los requisitos funcionales del sistema [23].

No	Nombre del requisito	Descripción
Prioridad		Alta
RF1	Modificar la configuración de la herramienta Elasticsearch para indexar y almacenar los <i>logs</i> .	Permitir realizar los cambios necesarios en el archivo de configuración de la herramienta para satisfacer las necesidades del cliente.
RF2	Modificar la configuración de la herramienta Logstash para recolectar y parsear los <i>logs</i> .	Permitir realizar los cambios necesarios en el archivo de configuración de la herramienta para satisfacer las necesidades del cliente.
RF3	Modificar la configuración de la herramienta Kibana para la visualización y de <i>logs</i> .	Permitir realizar los cambios necesarios en el archivo de configuración de la herramienta para satisfacer las necesidades del cliente.
RF4	Modificar la configuración de la herramienta rsyslog para el almacenamiento centralizado de <i>logs</i> .	Permitir realizar el almacenamiento centralizado de <i>logs</i> .
RF5	Permitir la instalación del sistema desde un dispositivo extraíble.	Permitir la instalación de la personalización desde un CD, DVD o dispositivo USB.

Tabla 3: Descripción de los requisitos funcionales(Creación del autor).

2.3.2 Requisitos no funcionales

Los requisitos no funcionales se refieren a cualidades que imponen restricciones en el diseño y la implementación. Un requisito no funcional especifica criterios que pueden usarse para juzgar el funcionamiento de un sistema en lugar de sus comportamientos específicos y sirven de apoyo a los requisitos funcionales. En la Tabla 4 se muestra el listado de los requisitos no funcionales [23].

No	Nombre del requisito no funcional	Atributo de calidad
RNF: 1	Acceder con el usuario root para realizar cambios en la configuración de la herramienta.	Confiabilidad
RNF: 2	El sistema se detiene si se ve afectado por ausencia de conexión a través de la red.	
RNF: 3	Si el funcionamiento del sistema se ve interrumpido por el fluido eléctrico, una vez que vuelva este, el sistema se reanuda y continúa realizando las operaciones normalmente.	
RNF: 4	El sistema va dirigido a administradores de redes y su nivel de complejidad se acopla fácilmente a los conocimientos básicos de un administrador.	Usabilidad
RNF: 5	La finalidad del sistema es permitir a los administradores de redes centralizar y visualizar de forma organizada los <i>logs</i> generados por todas las computadoras conectadas a la red.	
RNF: 6	Emplear como lenguaje de programación Bash.	Funcionalidad
RNF: 7	El sistema se ejecutará sobre el Sistema Operativo GNU/Linux Nova Servidores.	

Tabla 4: Descripción de los requisitos no funcionales (Elaboración propia).

2.3.3 Historias de usuario

Las historias de usuario especifican las tareas que debe realizar el sistema, lo que equivale a los casos de uso en el proceso unificado. Son escritas en lenguaje natural, sin un formato predeterminado, no excediendo su tamaño de unas pocas líneas de texto. Además, guían la construcción de las pruebas de aceptación y son utilizadas para estimar tiempos de desarrollo [24].

Solución de centralización y visualización de logs para Nova Servidores

Número: 2	Nombre del Requisito: Modificar la configuración de la herramienta Logstash para recolectar y parsear los <i>logs</i> .	
Programador: Stephany Hernández Castillo	Iteración Asignada: 1	
Prioridad: Alta	Tiempo Estimado: 30 días	
Riesgo en Desarrollo: Falta de personal calificado. Ausencia de desarrolladores por enfermedad.	Tiempo Real: 30 días	
<p>Descripción: Permite realizar los cambios necesarios en el archivo de configuración de la herramienta para satisfacer las necesidades del cliente.</p> <p>Se debe acceder al archivo de configuración de la herramienta Logstash y realizar las siguientes modificaciones en el <i>input</i>, los filtros y el <i>output</i>:</p> <p>input: path => ["/var/log/*/*"] start_position => beginning</p> <p>filter: match => ["syslog_timestamp", "MMM d HH:mm:ss", "MMM dd HH:mm:ss"]</p> <p>output: elasticsearch { hosts => ["localhost:9200"]</p>		
Observaciones: Para realizar estas modificaciones en el fichero de configuración es necesario acceder con privilegios de administrador.		
Prototipo		

Tabla 5: Historia de usuario del RF 2 Configurar la herramienta Logstash para recolectar y parsear los *logs*.

Solución de centralización y visualización de logs para Nova Servidores

Número: 4	Nombre del Requisito: Modificar la configuración de la herramienta rsyslog para el almacenamiento centralizado de <i>logs</i> .	
Programador: Stephany Hernández Castillo	Iteración Asignada: 1	
Prioridad: Alta	Tiempo Estimado: 15 días	
Riesgo en Desarrollo: Falta de personal calificado. Ausencia de desarrolladores por enfermedad.	Tiempo Real: 15 días	
<p>Descripción: Permite configurar la herramienta rsyslog para el almacenamiento centralizado de <i>logs</i>. Se debe acceder al archivo de configuración de la herramienta rsyslog y realizar las siguientes modificaciones:</p> <p>Activar los protocolos de transporte TCP y UDP:</p> <pre>module(load="imudp") input(type="imudp" port="514") module(load="imtcp") input(type="imtcp" port="514")</pre> <p>Crear una plantilla que genera dinámicamente el nombre del archivo de registro, dependiendo de la dirección IP del cliente:</p> <pre>\$template FILENAME, "/var/log/%fromhost-ip%/syslog.log" *. * ?FILENAME</pre>		
Observaciones: Para realizar estas modificaciones en el fichero de configuración es necesario acceder con privilegios de administrador.		
Prototipo		

Tabla 6: Historia de usuario del RF 4 Configurar la herramienta rsyslog para el almacenamiento centralizado de *logs*.

Número: 5	Nombre del Requisito: Permitir la instalación del sistema desde un dispositivo extraíble.	
Programador: Stephany Hernández Castillo	Iteración Asignada: 1	
Prioridad: Media	Tiempo Estimado: 2 días	
Riesgo en Desarrollo: Falta de personal calificado. Ausencia de desarrolladores por enfermedad.	Tiempo Real: 2 días	
<p>Descripción: Permite la instalación de la personalización desde un dispositivo extraíble.</p> <ul style="list-style-type: none"> • Seleccionar partición instalar: Instalación del sistema operativo en la partición seleccionada. • Pide una nueva contraseña para el usuario Administrador y valida que tenga la fuerza requerida: Insertar la contraseña con al fuerza requerida. • Pide repetir la contraseña y valida que coincida con la anteriormente introducida: Insertar nuevamente la contraseña para verificar que coincida con la anterior. • Pide confirmación para proceder con la instalación: Pulsar botón continuar para seguir instalando. 		
Observaciones:		
Prototipo		

Tabla 7: Historia de usuario del RF 5 Permitir la instalación del sistema desde un dispositivo extraíble.

2.4 Diagrama de paquetes del análisis de Nova Servidores 6.0

En Figura 2 se muestra el diagrama de paquetes del análisis que representa los principales componentes de la distribución cubana de GNU/Linux Nova Servidores y el paquete Nova Logs Server que se agrega como parte de la propuesta de solución. Los componentes de este diagrama agrupan para un mejor entendimiento las funcionalidades y servicios de la distribución [25].

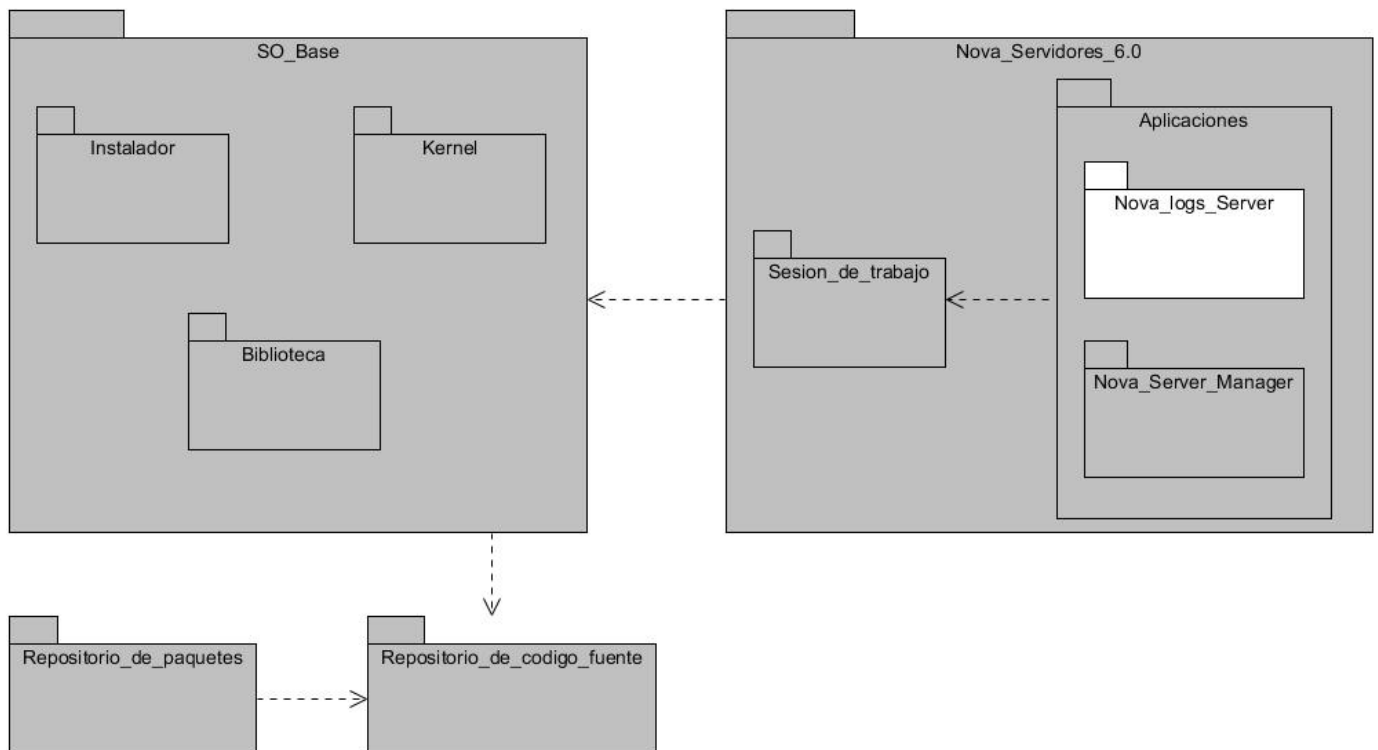


Figura 2: Diagrama de paquetes de Nova Servidores 6.0.

Descripción de los componentes del sistema [25]:

Repositorio de código fuente: es una colección de paquetes de programas en la distribución Nova que contienen archivos de código fuente que pueden ser descargados, modificados, analizados y reutilizado por los usuarios.

Repositorio de paquetes: es una colección de paquetes de programas de la distribución Nova que contiene archivos binarios precompilados que pueden ser descargados e instalados por los usuarios.

Sistema Operativo Base: estructura mínima de interacción con los componentes de funcionamiento del núcleo del sistema y los de la interfaz de usuario final. Se dedica al desarrollo de las capas tecnológicas básicas del sistema operativo, la definición de la arquitectura, la persistencia o gestión de los paquetes de

software, así como el desarrollo de tecnologías y servicio para la construcción de los productos.

- Instalador: herramienta que contiene módulos de gestión del meta-paquete debian-installer donde se modifican los paquetes necesarios para la versión de Nova Servidores 6.0 y se gestionan las tablas de peticiones, sistemas de archivos, instalación y configuración del sistema operativo.
- Kernel: es el componente central de un sistema operativo GNU/Linux. Se encarga de manejar los recursos *hardware* como la CPU, la memoria y los discos duros, y proporciona abstracciones que le dan a las aplicaciones una versión consistente de esos recursos.
- Biblioteca base: conjunto de archivos y programas precompilados en forma de biblioteca que brindan las herramientas y funcionalidades básicas para la construcción de una distribución GNU/Linux desde cero.

Nova Servidores 6.0: aplicación desarrollada para Servidores donde el trabajo en consola con la ayuda de interfaces gráficas ofrece al usuario una interacción cómoda con facilidades de acceso y configuración.

- Sesión de trabajo: es una colección de información que indica al sistema los archivos y carpetas a los que puede obtener acceso. Las sesiones de trabajo permiten compartir un equipo con varias personas pero manteniendo sus propios archivos y configuraciones. Cada persona obtiene acceso a su propia cuenta de usuario con un nombre de usuario y una contraseña.
- Aplicaciones: conjunto de aplicaciones comunes en las variantes de Nova-Base y Nova-Servidores.
- Nova server manager: es el gestor de administración de nova servidores. Se encarga de la instalación, configuración y desinstalación de los servicios para servidores con que cuenta Nova Servidores 6.0. Posee varios paquetes de servicios, los cuales abarcan las funcionalidades que posibilitan el uso de los servicios para los distintos tipos de servidores contemplados dentro de la arquitectura del sistema. Se agrupan las funcionalidades referidas a los 17 servicios usables y

Solución de centralización y visualización de logs para Nova Servidores

configurables por la herramienta Nova-server-manager. Se maneja también todo lo relacionado con las validaciones de configuraciones, las utilidades e internacionalización de la herramienta y las vistas con las que interactuarán los usuarios.

- Nova Logs Server: contiene los paquetes de la personalización que permite centralizar y visualizar los *logs* generados por las computadoras conectadas a la red.

2.4.1 Diagrama de paquetes del análisis de Nova Logs Server

En la Figura 3 se muestra el diagrama de paquetes del análisis. En dicho diagrama se representan los componentes que darán respuesta a los requisitos funcionales identificados para la personalización de Nova Servidores que se espera como resultado de la investigación.

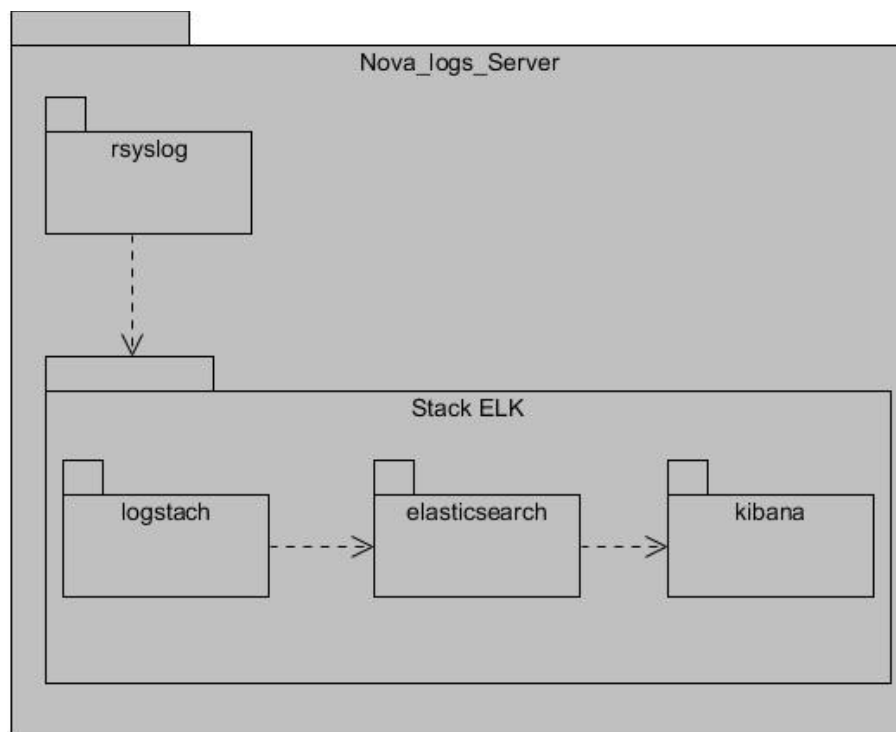


Figura 3: Diagrama de paquetes de la personalización (Elaboración propia).

Rsyslog: contiene las configuraciones necesarias para realizar el almacenamiento centralizado los *logs* generados por las computadoras conectadas.

Stack ELK: contiene las herramientas Elasticsearch, Logstash y Kibana que permiten la administración y visualización de *logs*.

- Logstash: es la herramienta que se encarga de recolectar, parsear y enviar los *log* a Elasticsearch.
- Elasticsearch: es la herramienta que se encarga de indexar y almacenar los *logs* recolectados por Logstash.
- Kibana: es la herramienta que se encarga de la gestión y visualización de los *logs*.

Conclusiones del capítulo

En el desarrollo del capítulo se definieron las características y funcionalidades de la personalización de la distribución cubana de GNU/Linux Nova Servidores para el almacenamiento centralizado y visualización de *logs*. El diagrama de paquetes permitió representar las relaciones y dependencias entre los elementos que componen la propuesta de solución, se obtuvieron 10 requisitos funcionales y 7 no funcionales cumpliendo con las necesidades del cliente. La gestión del proceso de construcción de la personalización permitió identificar las tareas, el tiempo y los riesgos de la implementación.

Capítulo 3: Implementación y pruebas

Introducción

Durante este capítulo se documentan las fases de construcción de la personalización según la metodología empleada. Se determina el funcionamiento y objetividad de la propuesta desarrollada con el objetivo de evaluar su calidad mediante pruebas; para determinar si los requisitos identificados realmente definen el sistema que se debe construir. Los elementos teóricos definidos en capítulos anteriores constituyen el punto de partida en la construcción de la personalización de la distribución GNU/Linux Nova Servidores destinada a la visualización centralizada de *logs*. Como parte del proceso de evaluación de la propuesta, se describen las pruebas realizadas al sistema para comprobar su correcto funcionamiento.

3.1 Descripción del proceso de construcción

Antes de iniciar la construcción de la personalización es necesario empaquetar cada a una de las herramientas con sus configuraciones previas y colocarlas en el repositorio de Nova Servidores 6.0. A continuación se describe como fue todo el proceso de empaquetado de una de las herramientas.

Como primer paso se crea una carpeta local para trabajar sobre ella y asignarle permisos de escritura, lectura y ejecución, para ello se ejecutan los siguientes comandos:

```
mkdir elasticsearch  
chmod 777 -R elasticsearch
```

Luego dentro de la carpeta a empaquetar se crea la ruta contenedora del paquete donde se encuentra el sistema una vez instalado. Ejemplo /usr/share (dentro del directorio share está el código fuente de la herramienta). Se crea un comprimido de la carpeta elasticsearch utilizando la extensión tar.xz y se le realiza una copia con el formato paquete_versión.orig.tar.xz. Para realizar la copia se utiliza el comando:

```
cp elasticsearch-2.3.3.tar.xz elasticsearch-2.3.3.orig.tar.xz
```

En el directorio donde se encuentra la herramienta, en este caso elasticsearch/elasticsearch-2.3.3 se

crean dos variables de entorno para especificar las credenciales del empaquetador:

Ejemplo:

```
export DEBEMAIL=shcastillo@estudiantes.uci.cu
export DEBFULLNAME= "Stephany Hernández"
```

Se genera la plantilla de la carpeta Debian y se eliminan todos los ficheros con extensión *.ex y *.EX que no sean necesarios. Si alguno de estos ficheros es de utilidad para la herramienta Elasticsearch, se les quita la extensión ".ex". En preinst se definen instrucciones que se ejecutan antes de la instalación del paquete, en postinst luego de la instalación, en prepm antes de la eliminación y en postpm luego de la eliminación. La plantilla de la carpeta Debian se genera con el comando:

```
dh_make -e $DEBEMAIL -c gpl3 -s
```

Las modificaciones realizadas en el código fuente se archivan en el changelog. Dentro de la carpeta Debian se crea un fichero llamado *install* donde se especifica para qué lugar del sistema operativo irá cada elemento del paquete, en este caso: `usr/share/* usr/share/`. Una vez concluidas con todas las modificaciones en el directorio de Debian, se construye el fichero de control de fuente de debian (**.dsc**), ejecutando el comando:

```
dpkg-source -b .
```

Se crea el fichero **.pbuilderrc** en **/home/user/** y se modifican los siguientes parámetros.

```
MIRRORSITE=http://nova.f10.uci.cu/nova/2017/
```

```
DISTRIBUTION=2017
```

```
COMPONENTS="principal extendido"
```

```
DEBOOTSTRAPOPTS=(
```

```
  '-arch=all'
```

```
  '-variant=buldd'
```

```
  '-components=principal,extendido'
```

```
  '-keyring=/usr/share/keyrings/nova-archive-keyring.gpg')
```

Para crear y construir el fichero se ejecutan los comandos:

```
sudo pbuilder create
```

```
sudo pbuilder build elasticsearch_2.3.3-1ubuntu1nova1.dsc
```

Finalmente para instalar los paquetes generados se utiliza el comando:

```
sudo dpkg -i elasticsearch_2.3.3-1ubuntu1_amd64.deb
```

Para la construcción de la personalización se decidió dividir el proceso en tres fases principales:

- Primera fase: construcción del Sistema Operativo Base.
- Segunda fase: construcción del Sistema Operativo Final.
- Tercera fase: empaquetamiento del sistema.

3.1.1 Primera fase: Construcción del Sistema Operativo Base

Para la construcción de una personalización de Nova, el primer paso consiste en la instalación del Sistema Operativo Base (SOB). La mayoría de los comandos utilizados durante esta primera fase de construcción requieren privilegios de administrador, por lo que deben ser ejecutados con la orden “sudo” al inicio de la invocación.

Para la creación del SOB inicialmente se crea el directorio donde se va a alojar y se instala la herramienta debootstrap. Para ello se utilizan comando como:

```
$mkdir /mnt/nova_server
```

```
$sudo apt-get install debootstrap
```

Durante la creación del SOB se especifica la opción --components que indica las ramas del repositorio de las que se utilizan paquetes durante el proceso de la creación del SOB, la dirección del repositorio de donde se descargan los paquetes que en este caso es <http://novaf10.uci.cu/nova/2017> y una vez concluido este proceso el SOB se encuentra instalado en /mnt/nova_server. Todo esto se realiza ejecutando el comando:

```
$sudo debootstrap --components=principal,extendido 2017 /mnt/nova_server/  
http://nova.f10.uci.cu/nova/2017
```

3.1.2 Segunda Fase: Construcción del Sistema Operativo Final

Durante la segunda fase de construcción se realizan las configuraciones y modificaciones necesarias para que el sistema cumpla con los requerimientos especificados. Inicialmente debe montar /dev en la carpeta donde se encuentra el SOB. El directorio /dev contiene los archivos de dispositivos especiales para todos los dispositivos de *hardware*.

```
$sudo mount --bind /dev /mnt/nova_server/dev
```

Para conectarse y descargar de los repositorios se debe copiar el fichero resolv.conf del equipo anfitrión (este contiene las direcciones de los servidores de DNS del dominio) e inicializar el fichero initctl en una constante true para que una vez que las aplicaciones que se instalen en el sistema que está en proceso de creación, soliciten el estado de un servicio y puedan comprobar que este está activo:

```
$sudo cp /etc/resolv.conf /mnt/nova_server/etc/  
$sudo mv /mnt/nova_server/sbin/initctl /mnt/nova_server/sbin/initctl.nim_blocked  
$sudo ln -s /mnt/nova_server/bin/true /sbin/initctl
```

Luego se montan los procesos. El directorio /proc contiene un sistema de archivos imaginario o virtual. Este no existe físicamente en disco, sino que el núcleo lo crea en memoria. Se utiliza para ofrecer información relacionada con el sistema (originalmente acerca de procesos, por eso su nombre). Se monta /sys, el cual contiene parámetros de configuración del sistema que se está ejecutando, se monta /dev/pts, este sistema de ficheros vive exclusivamente en la memoria, y se exportan algunas variables del sistema.

```
$sudo mount -t proc none /mnt/nova_server/proc  
$sudo mount -t sysfs none /mnt/nova_server/sys  
$sudo mount -t devpts none /mnt/nova_server/dev/pts  
$sudo chroot /mnt/nova_server export LC_ALL=C
```

Se actualizan los índices de la caché del sistema para luego poder instalar paquetes que se necesitan en

Solución de centralización y visualización de logs para Nova Servidores

el sistema operativo a construir y se instala el protocolo dbus para permitir la comunicación entre los procesos de las aplicaciones a instalar, para esto se utilizan los comandos:

```
$sudo chroot /mnt/nova_server apt-get update  
$sudo chroot /mnt/nova_server apt-get install --yes dbus
```

Finalmente se monta el entorno chroot que este permite interactuar con el sistema que se está creando como si se tratara de un sistema en marcha. Una vez dentro de este entorno, se tienen privilegios de root:

```
$sudo chroot /mnt/nova_server/
```

- **Instalación de paquetes**

Una vez realizadas las configuraciones iniciales se procede a la instalación de los paquetes que debe tener el Sistema Operativo Final, así como las configuraciones que se deseen realizar para adaptarlo a necesidades específicas. A continuación se muestra el listado de aplicaciones que fueron instaladas y seleccionadas en el capítulo anterior (Ver Tabla 8).

Paquetes	Descripción
Rsyslog	Herramienta para la centralización de <i>logs</i>
Elasticsearch	Herramienta para indexar y almacenar <i>logs</i>
Logstash	Herramienta para recolectar y parsear <i>logs</i>
Kibana	Herramienta para la visualización y administración de <i>logs</i>

Tabla 8: Paquetes para configurar (Elaboración propia).

- **Configuraciones del sistema**

Para obtener el sistema deseado no solo se depende de los requerimientos establecidos o la instalación de aplicaciones, sino de su configuración, para que la interacción de los usuarios con las mismas sea lo más fácil posible. Basándose en lo anterior se procede de la siguiente manera para realizar la configuración de las aplicaciones instaladas:

Configuración de rsyslog

Se accede al fichero de configuración de la herramienta rsyslog con privilegios de administración para poder realizar los cambios necesarios:

```
#nano /etc/rsyslog.conf
```

Se descomentan las líneas para utilizar el protocolo UDP y TCP por el puerto 514 y se crea una plantilla que genera dinámicamente el nombre del archivo de *logs*, dependiendo de la dirección IP del cliente.

```
$module(load="imudp")  
$input(type="imudp" port="514")  
$module(load="imtcp")  
$input(type="imtcp" port="514")  
$template FILENAME, "/var/log/%fromhost-ip%/syslog.log"  
*.* ?FILENAME
```

Configuración de Elasticsearch

Para que la herramienta Elasticsearch tenga un correcto funcionamiento y trabaje en equipo con Logstash y Kibana es necesario realizar las siguientes modificaciones en su fichero de configuración.

```
nano /elasticsearch/config/elasticsearch.yml
```

```
network:
```

```
host: localhost
```

```
path:
```

```
logs: /var/log/elasticsearch
```

```
data: /var/data/elasticsearch
```

Configuración de Logstash

La herramienta Logstash es la que se encarga de filtrar, parsear y enviar a Elasticsearch los *logs* recolectados en el servidor. A continuación se muestra la configuración que se utilizó en este caso:

```
nano /elasticsearch/conf/logstash.conf
```

```
input {
```

```
file {
```

```
path => ["/var/log/*/*"]
start_position => beginning
}
}
filter {
  if [type] == "syslog" {
    grok {
      match => { "message" => "%{SYSLOGTIMESTAMP:syslog_timestamp} %
{SYSLOGHOST:syslog_hostname} %{DATA:syslog_program}(?:\[%{POSINT:syslog_pid}\])?: %
{GREEDYDATA:syslog_message}" }
      add_field => [ "received_at", "%{@timestamp}" ]
      add_field => [ "received_from", "%{host}" ]
    }
    date {
      match => [ "syslog_timestamp", "MMM d HH:mm:ss", "MMM dd HH:mm:ss" ]
    }
  }
}
output {
  elasticsearch { hosts => ["localhost:9200"] }
  stdout { codec => rubydebug }
}
```

Configuración de Kibana

Kibana por su parte es la herramienta que se encarga de la gestión y la visualización de los *logs* por lo que es necesario especificar en su fichero de configuración el puerto por el cual va a estar disponible para que los administradores puedan acceder a través de su navegador.

```
nano /kibana/config/kibana.yml
server.port: 5601
```

- **Restauración de las configuraciones del sistema:**

Para que el Sistema Operativo Final creado funcione correctamente, se deben restaurar los valores adecuados así como borrar los archivos temporales para disminuir el tamaño del sistema y luego se desmontan las unidades montadas inicialmente:

```
$sudo rm -f /mnt/nova_server/etc/resolv.conf
```



```
$sudo rm -rf /mnt/nova_server/tmp/*
$sudo rm -f /mnt/nova_server/sbin/initctl
$sudo mv /mnt/nova_server/initctl.nim_blocked /mnt/nova_server/sbin/initctl/sbin/
$sudo umount /mnt/nova_server/proc
$sudo umount /mnt/nova_server/sys
$sudo umount /mnt/nova_server/dev/pts
$sudo umount /mnt/nova_server/dev/
```

3.1.3 Tercera fase: Empaquetamiento del sistema

Primeramente se crean las carpetas donde se deben instalar los archivos necesarios para la creación de la imagen ISO-9660, en este caso se creó en “/mnt” la carpeta “iso” la cual contiene las subcarpetas “casper”, “isolinux”, e “install” y se mueven los archivos del núcleo para la carpeta “casper”:

```
$sudo mkdir -p /mnt/iso/{casper,isolinux,install}
$sudo mv /mnt/nova_server/boot/vmlinuz-3.8.0-35-generic /mnt/iso/casper/vmlinuz
$sudo mv /mnt/nova_server/boot/initrd.img-3.8.0-35-generic /mnt/iso/casper/initrd.lz
```

Se copia el contenido de la carpeta “isolinux” presente en cualquier ISO de Nova, que ya está previamente configurada con un splash y un menú en concordancia con el patrón de diseño de Nova, para la carpeta creada con el mismo nombre. Se copia la lista de los paquetes instalados en el sistema dentro de la carpeta “casper” en el fichero “filesystem.manifest”:

```
$sudo chroot /mnt/nova/ dpkg-query -W --showformat='${Package} ${Version}\n' | tee/mnt/iso/casper/filesystem.manifest
```

De este grupo de paquetes instalados para el sistema *Live* no todos son necesarios una vez instalado el mismo en una terminal cliente, por lo que se deben excluir estos paquetes en aras de tener un sistema limpio de elementos innecesarios.

Los paquetes a excluir son:

- El instalador (serere).
- Todas las herramientas que proveen el funcionamiento de un sistema Live (casper, live-initramfs, user-setup).

Solución de centralización y visualización de logs para Nova Servidores

```
REMOVE='serere3-slideshow serere3-gtk3 casper user-setup live-initramfs'
for i in $REMOVE;
do
sudo sed -i "${i}/d" image/casper/filesystem.manifest-desktop;
done
```

Luego se necesita el “filesystem.squashfs”, que no es otra cosa que el sistema comprimido en el formato squashfs:

```
$sudo mksquashfs /mnt/nova_server/ /mnt/iso/casper/filesystem.squashfs -comp xz
$sudo printf $(du -sx --block-size=1 /mnt/nova_server | cut -f1) >/mnt/iso/casper/filesystem.size
```

Se escriben las definiciones del sistema *Live* en el archivo “README.diskdefines”, donde se especifican la arquitectura de computadora utilizada, la numeración del disco y una descripción del sistema:

```
$sudo touch /mnt/iso/README.diskdefines
$sudo nano /mnt/iso/ README.diskdefines
```

Quedando de la manera siguiente:

```
#define DISKNAME Nova Logs Server i386
#define TYPE binary
#define TYPE binary 1
#define ARCH i386
#define ARCH i386 1
#define DISKNUM 1
#define DISKNUM1 1
#define TOTALNUM 0
#define TOTALNUM0 1
```

Se crea la carpeta “.disk” y se escriben en ella otros datos necesarios:

```
$sudo mkdir /mnt/iso/.disk
$sudo touch /mnt/iso/.disk/base_installable
$sudo echo “full_cd/single” > /mnt/iso/.disk/cd_type
$sudo echo “Nova Logs Server” > /mnt/iso/.disk/info
$sudo echo “http://www.nova.cu” > /mnt/iso/.disk/release_notes_url
```

Se escribe la suma de control MD5:

```
$cd /mnt/iso
$find . -type f -print0 | xargs -0 md5sum | grep -v “\./md5sum.txt” > md5sum.txt
```

Por último se crea la imagen de instalación ISO-9660 como se muestra a continuación:

```
$cd /mnt/iso  
$sudo mkisofs -r -V "Nova Logs Server" -cache-inodes -J -l -b isolinux/isolinux.bin -c isolinux/boot.cat -no-emul-boot -boot-load-size 4 -boot-info-table -o /mnt/nova-logs-server-$(date+"%Y%m%d").iso
```

Como resultado se obtiene en la carpeta '/mnt' la imagen iso del sistema.

3.2 Pruebas de software

Las pruebas de *software* es un concepto que a menudo es conocido como verificación y evaluación del *software*. Integra las técnicas de diseño de casos de prueba en una serie de pasos bien planificados que dan como resultado una correcta construcción del *software* [26].

Para la ejecución de las pruebas a Nova Logs Server se definió el método de caja negra, definiendo como tipo de prueba la prueba de funcionalidad, realizada para detectar no conformidades; las pruebas son aplicadas en diferentes niveles, de los cuales se utiliza el nivel de prueba de aceptación, que permitió la ejecución de casos de pruebas.

- **Pruebas de caja negra**

Se refiere a las pruebas que se llevan a cabo sobre la interfaz del *software*, por lo que los casos de prueba pretenden demostrar que las funciones del *software* son operativas, que la entrada se acepta de forma adecuada y que se produce una salida correcta, así como que la integridad de la información externa se mantiene. Esta prueba examina algunos aspectos del modelo fundamentalmente del sistema sin tener mucho en cuenta la estructura interna del *software* [26].

- **Prueba de aceptación**

Son básicamente pruebas funcionales sobre el sistema completo, y buscan comprobar que se satisfacen los requisitos establecidos. Su ejecución es facultativa del cliente, y en el caso de que no se realicen explícitamente, se dan por incluidas dentro de las pruebas de sistema. Es decir, las pruebas de aceptación son, a menudo, responsabilidad del usuario o del cliente, aunque cualquier persona involucrada en el

Solución de centralización y visualización de logs para Nova Servidores

negocio puede realizarlas. La ejecución de las pruebas de aceptación requiere un entorno de pruebas que represente el desarrollo de producción [26].

La aplicación del método de caja negra a la solución estará centrado en comprobar el correcto flujo de información entre las herramientas que la conforman (Elasticsearch, Logstash y Kibana). El objetivo de la prueba consiste en evaluar el manejo y representación de la información, comprobando así la correspondencia del sistema con las especificaciones o requerimientos del cliente. Para ello se define un entorno de pruebas que permita conocer con exactitud que los *logs* generados por las estaciones clientes correspondan con los que se estén centralizando en el servidor.

Entorno de pruebas:

Computadoras	Propiedades
Servidor de <i>logs</i>	Inetl core i7-4610 CPU: 3.10 GHz RAM: 8gb disco duro: 500 GB
Clientes (8)	Inetl core i5-4440 CPU: 3.10 GHz RAM: 8gb disco duro: 500 GB

Tabla 9: Entorno de prueba (Elaboración propia).

Condiciones de ejecución:

- Inicialmente se eliminan todos los *logs* de las estaciones clientes y del servidor para comprobar la correspondencia entre la cantidad de *logs* generados por los clientes y los centralizados en el servidor.
- Se utiliza el comando `tail -f nombre_log` para visualizar desde el servidor los *logs* de los clientes en tiempo real, donde `nombre_log` se refiere al *logs* que desea visualizar.

Resultados de la ejecución:

Duración	48 horas
Estaciones Clientes	Cantidad de logs generados
Cliente 1	567
Cliente 2	423
Cliente 3	475
Cliente 4	389
Cliente 5	562
Cliente 6	468
Cliente 7	423
Cliente 8	526
Total de logs almacenados en el servidor	3 833

Tabla 10: Resultados de la ejecución (Elaboración propia).

Se demostró que la suma de la cantidad de *logs* generados por cada estación cliente es exactamente igual a la cantidad de *logs* almacenados en el servidor y que se reciben en tiempo real.

3.2.1 Casos de Pruebas

Un caso de prueba es una serie de pruebas de entrada, condiciones de ejecución y resultados esperados desarrollados para un objetivo definido, tal como ejecutar una ruta particular de un programa o evaluar el cumplimiento con un requerimiento en específico [27].

A continuación se muestran los casos de pruebas planificados para el desarrollo de la evaluación del sistema a desarrollar.

Caso de prueba 2 (Ver Tabla 11)

CPR2: Configurar la herramienta Logstash para recolectar y parsear los *logs*.

Condiciones de ejecución: Modificar el archivo de configuración de la herramienta Logstash.

Solución de centralización y visualización de logs para Nova Servidores

Escenario	Descripción	V1	Respuesta del sistema	Flujo central
EC 1.1 Configurar la variable <i>input</i> .	Modificar la variable <i>input</i> para establecer la ruta de la cual serán extraído los <i>logs</i> .	V	El sistema es capaz de indexar los <i>logs</i> a partir de la ruta especificada.	Configurar la variable <i>input</i> de la herramienta Logstash.
		I	El sistema no es capaz de indexar los <i>logs</i> aún cuando esté corriendo.	
		Se establece una ruta incorrecta.		
EC 1.2 Configurar los filtros de la herramienta.	Modificar los filtros para indexar los <i>logs</i> almacenados a partir de su fecha.	V	La herramienta realiza un correcto indexado y filtrado de <i>logs</i> .	Configurar los filtros de la herramienta Logstash para el indexado por fecha de los <i>logs</i> almacenados.
		I	La herramienta no es capaz de realizar el indexado correcto de las fechas.	
		Establecer un formato incorrecto para los filtros de la herramienta.	Formato inválido para el filtro de la herramienta.	
EC 1.3 Configurar la variable <i>output</i> de la herramienta.	Modificar la variable <i>output</i> para enviar los <i>logs</i> almacenados a la herramienta Elasticsearch.	V	La herramienta envía los <i>logs</i> almacenados a la herramienta Elasticsearch.	Configurar la variable <i>output</i> para enviar los <i>logs</i> almacenados a la herramienta Elasticsearch.
		I	La herramienta es incapaz de enviar los <i>logs</i> almacenados a la herramienta Elasticsearch.	
		Se establece la ruta correcta en que se encuentra la herramienta Elasticsearch.	Se establece una ruta inválida para comunicarse con la herramienta Elasticsearch.	

Tabla 11: Caso de prueba del RF2 (Elaboración propia).

Caso de prueba 5 (Ver Tabla 12)

CPR5: Permitir la instalación del sistema desde un dispositivo extraíble.

Condiciones de ejecución: Se debe contar con el sistema en un dispositivo booteable.

Solución de centralización y visualización de logs para Nova Servidores

Escenario	Descripción	V1	Respuesta del sistema	Flujo central
EC 1.1	Seleccionar partición a instalar.	V	Instala en la partición seleccionada.	Instalación del sistema operativo en la partición seleccionada.
		Seleccionar partición a instalar.		
		I	Instala en todo el disco.	
		No selecciona partición a instalar.		
EC 1.2	Pide una nueva contraseña para el usuario Administrador y valida que tenga la fuerza requerida.	V	Pedir repetir contraseña.	Insertar la contraseña con la fuerza requerida.
		Contraseña fuerte.		
		I	Mostrar mensaje de error y volver a pedir contraseña nuevamente.	
		Contraseña débil.		
EC 1.3	Pide repetir la contraseña y valida que coincida con la anteriormente introducida.	V	Pedir información al usuario para instalar.	Insertar nuevamente la contraseña para comprobar que coincida con la anterior.
		Coincide contraseña.		
		I	Mostrar mensaje de error y volver a pedir contraseña.	
		No coincide contraseña.		
EC 1.4	Pide confirmación para proceder con la instalación.	V	Instalar el sistema operativo.	Pulsar botón continuar para seguir instalando.
		Presionar botón para continuar.		
		I	Salir del programa de instalación.	
		No presionar botón continuar.		

Tabla 12: Caso de prueba del RF5 (Elaboración propia).

3.2.2 Resultados obtenidos

Se realizaron un total de 3 iteraciones de pruebas que arrojaron como resultados 12 no conformidades durante la primera iteración. Luego de trabajar durante un período en su solución y efectuar la segunda iteración se redujo a 4 no conformidades, los cuales fueron eliminados antes de concluir la personalización.

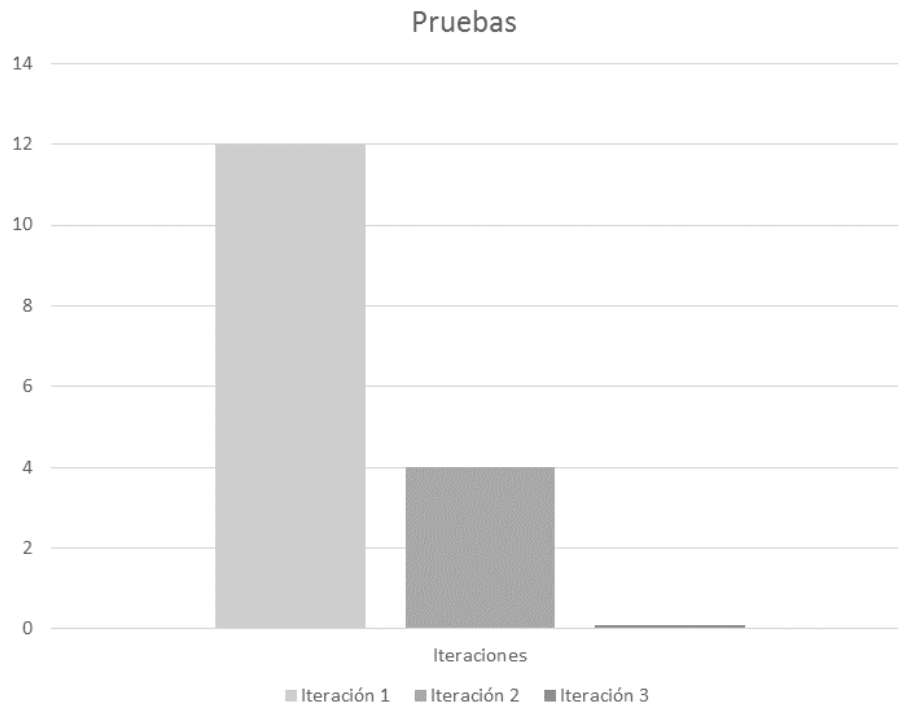


Figura 4: No conformidades en las pruebas al sistema (Elaboración propia).

Tal como se muestra en la siguiente Figura 4, la resolución de las no conformidades detectadas en esta etapa permitió el aseguramiento de una mayor calidad en el producto.

Conclusiones del capítulo

Durante este capítulo se describió detalladamente el proceso de construcción de la personalización de la distribución de GNU/Linux Nova en su versión para servidores para el almacenamiento centralizado y la visualización de *logs*. El proceso llevado a cabo y las pruebas realizadas al sistema a partir de los casos de prueba definidos teniendo en cuenta los requisitos funcionales, permitieron evaluar luego de tres iteraciones una personalización que cumple con las necesidades planteadas por el cliente.

Conclusiones generales

Durante el desarrollo de la investigación y la obtención del producto final los principales resultados obtenidos fueron:

- El estudio realizado sobre las herramientas existentes que permitían dar solución a la problemática de la investigación arrojó que la combinación de las herramientas Rsyslog y el *stack* ELK (Elasticsearch, Logstash y Kibana) cubría todas las necesidades de almacenamiento centralizado y visualización de *logs* en Nova Servidores.
- La arquitectura de despliegue utilizada en la solución, permitió realizar el análisis sobre los *logs* generados por las estaciones clientes que sean administradas desde la personalización de Nova Servidores que se propone como resultado de la presente investigación.
- Las pruebas diseñadas y ejecutadas evaluaron el correcto funcionamiento del sistema así como el cumplimiento de todos los requisitos definidos.

Recomendaciones

- Implementar un mecanismo que garantice la seguridad en la comunicación y transferencia de los datos entre las estaciones clientes y el servidor.

Referencia bibliográfica

- [1] Viera Hernández, Amaury y De La Rosa Gómez, Leonardo. MigrateAD: Migración del Directorio Activo a plataforma libre. 2009.
- [2] Universidad de las Ciencias Informáticas. Misión | Portal de la Universidad de las Ciencias Informáticas. <<http://www.uci.cu/?q=mision>> [Accedido 8 de noviembre de 2016].
- [3] Alexey García Arévalo, Osmani López Cardoso. *Propuesta para la Recolección Centralizada de Logs*. Universidad de las Ciencias Informáticas, 2008.
- [4] Pierra Fuentes Alan. Nova, Distribución Cubana de GNU/Linux. Reestructuración estratégica de su proceso de desarrollo. [online]. La Habana, 2011. Available from: <http://catalogoenlinea.uci.cu/cgi-bin/koha/opac-detail.pl?biblionumber=11161>
- [5] Mosqueda, Ariannis Lafita and Guzmán, Ángel Camilo Guillén. NOVAMEDIA: Personalización de Nova orientada al diseño gráfico y la animación para FreeViUX. [online]. La Habana : Universidad de las Ciencias Informáticas, 2012. [Accessed 20 Diciembre 2016]. Available from: <http://catalogoenlinea.uci.cu/cgi-bin/koha/opac-detail.pl?biblionumber=11684>
- [6] Bruce Schneier, John Kelsey. Cryptographic Support for Secure Logs on Untrusted Machines. 2015.
- [7] Bruce, John Kelsey. Secure Audit Logs to Support Computer Forensics. 2014
- [8] syslog(3): send messages to system logger - Linux man page. [online]. [Accessed 13 Ene 2017]. Available from: <https://linux.die.net/man/3/syslog>
- [9] Jason E. Holt. Logcrypt: Forward Security and Public Verification for Secure Audit Logs. 2014.
- [10] Logwatch - Community Help Wiki. [online]. [Accessed 1 Feb 2017]. Available from: <https://help.ubuntu.com/community/Logwatch>
- [11] www2.logwatch.org. [online]. [Accessed 1 Feb 2017]. Available from: <http://www.stellarcare.net/logwatch/tabs/docs/HOWTO-Customize-LogWatch.html>
- [12] Event Log View and Analyzer, Network Monitoring and Management Software | GFI EventsManager. [online]. [Accessed 5 Mar 2017]. Available from: <https://www.gfi.com/products-and-solutions/network->

[security-solutions/gfi-eventsmanager](#)

[13] rsyslog 8.25.0 (v8-stable) released. [online]. [Accessed 5 Mar 2017]. Available from: <http://www.rsyslog.com/rsyslog-8-25-0-v8-stable-released/>

[14] Open Source Search & Analytics · Elasticsearch | Elastic. [online]. [Accessed 13 Abr 2017]. Available from: <https://www.elastic.co/>

[15] Grafana Features | Grafana Labs. [online]. [Accessed 7 June 2017]. Available from: <https://grafana.com/grafana>

[16] Tamara Rodríguez Sánchez. Metodología de desarrollo para la Actividad productiva de la UCI. . 2015.

[17] Visual Paradigm Features. [online]. [Accessed 6 Abr 2017]. Available from: <https://www.visual-paradigm.com/features/>

[18] Cabrera, Lianet. 2012. Extensión de Visual Paradigm for UML para el Desarrollo Dirigido por Modelos de aplicaciones de gestión de información. [En línea] 2012. [Citado el: 2 de 10 de 2016.] https://www.redib.org/recursos/Record/oai_articulo983403-extension-visual-paradigm-uml-desarrollo-dirigido-modelos-aplicaciones-gestion-informacion.

[19] Debootstrap - Debian Wiki. [online]. [Accessed 13 May 2017]. Available from: <https://wiki.debian.org/Debootstrap>

[20] genisoimage - Debian Wiki. [online]. [Accessed 13 May 2017]. Available from: <https://wiki.debian.org/genisoimage>

[21] BASH Programming - Introduction HOW-TO. [online]. 27 July 2000. [Accessed 10 Feb 2017]. Available from: <http://www.tldp.org/HOWTO/Bash-Prog-Intro-HOWTO.html>

[22] Sublime Text: The text editor you'll fall in love with. [online]. [Accessed 6 Abr 2017]. Available from: <http://www.sublimetext.com/>

[23] IEEE 610-1990. Instituto de Ingenieros Eléctricos y Electrónicos. IEEE Computer Dictionary. Software Engineering Terms. 1990.

[24] Palma Pérez, Nurisel. Módulo para la administración de los servidores web en HMAST. Universidad de las Ciencias Informáticas, 2013.

Solución de centralización y visualización de logs para Nova Servidores

[25] Colectivo de Autores. *Modelo de diseño Proyecto Nova-Servidores 6.0*. 6 March 2017.

[26] S. Pressman, Roger. *Ingeniería de software. Un enfoque práctico*. Quinta edición.

[27] NATALIA JURISTO, ANA M. MORENO and SIRA VEGAS. *Técnicas de Evaluación de Software* [online]. 17 October 2005. Available from: [http://www.google.com.cu/url?](http://www.google.com.cu/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&sqi=2&ved=0CBwQFjAA&url=http%3A%2F)

[sa=t&rct=j&q=&esrc=s&source=web&cd=1&sqi=2&ved=0CBwQFjAA&url=http%3A%2F](http://www.google.com.cu/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&sqi=2&ved=0CBwQFjAA&url=http%3A%2F)

[%2Fwww.grise.upm.es%2Fsites%2Fextras%2F12%2Fpdf](http://www.google.com.cu/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&sqi=2&ved=0CBwQFjAA&url=http%3A%2F)

[%2FDocumentacion_Evaluacion_7.pdf&ei=4gFdVdyzDcjZsATY2oHQBw&usg=AFQjCNFQq1YCIEX4LCA1_ZdckCMaall1Wg&bvm=bv.93756505.d.cWc](http://www.google.com.cu/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&sqi=2&ved=0CBwQFjAA&url=http%3A%2F)

Anexo 1: Historias de Usuario

Número: 1	Nombre del Requisito: Modificar la configuración la herramienta Elasticsearch para indexar y almacenar <i>logs</i> .	
Programador: Stephany Hernández Castillo	Iteración Asignada: 1	
Prioridad: Alta	Tiempo Estimado: 25 días	
Riesgo en Desarrollo: Falta de personal calificado. Ausencia de desarrolladores por enfermedad.	Tiempo Real: 20 días	
<p>Descripción: Permite realizar los cambios necesarios en el archivo de configuración de la herramienta para satisfacer las necesidades del cliente.</p> <p>Se debe acceder al archivo de configuración de la herramienta Elasticsearch y realizar las siguientes modificaciones:</p> <p>network : host : 127.0.0.1</p> <p>path: logs: /var/log/elasticsearch data: /var/data/elasticsearch</p> <p>cluster: name: cluster1</p> <p>node: name: nodo1</p>		
Observaciones: Para realizar estas modificaciones en el fichero de configuración es necesario acceder con privilegios de administrador.		
Prototipo		

Tabla 11: Historia de usuario del RF 1 Configurar la herramienta Elasticsearch para indexar y almacenar *logs*.

Número: 3	Nombre del Requisito: Modificar la configuración la herramienta Kibana para la visualización de <i>logs</i> .	
Programador: Stephany Hernández Castillo	Iteración Asignada: 1	

Solución de centralización y visualización de logs para Nova Servidores

Prioridad: Alta	Tiempo Estimado: 20 días
Riesgo en Desarrollo: Falta de personal calificado. Ausencia de desarrolladores por enfermedad.	Tiempo Real: 20 días
Descripción: Permite realizar los cambios necesarios en el archivo de configuración de la herramienta para satisfacer las necesidades del cliente. Se debe acceder al archivo de configuración de la herramienta Kibana y realizar las siguientes modificaciones: server.port: 5601	
Observaciones: Para realizar estas modificaciones en el fichero de configuración es necesario acceder con privilegios de administrador.	
Prototipo	

Tabla 12: Historia de usuario del RF 3 Configurar la herramienta Kibana para la visualización de logs.

Anexo 2: Acta de aceptación del producto


 **Acta de aceptación de productos de trabajo**

ACTA DE ACEPTACIÓN DE PRODUCTOS DE TRABAJO

En cumplimiento del **Convenio de colaboración** establecido entre el **Centro de Software Libre (CESOL)** y la estudiante **Stephany Hernández Castillo** de la Facultad 1 de la Universidad de las Ciencias Informáticas y en función de la ejecución del proyecto: **Nova Logs Server**, se hace entrega de los productos que se relacionan a continuación:

- Nova Logs Server

La parte Cliente, luego de haber revisado los productos de trabajos relacionados anteriormente procede a firmar la aceptación de los mismos en total conformidad.

Entrega	Recibe
Nombre y apellidos: Stephany Hernández Castillo	Nombre y apellidos: Nello Véliz Pedraza
Cargo: Estudiante Facultad 1	Cargo: Jefe de Departamento
Firma: 	Firma: 

Fecha: 31/05/2017