Universidad de las Ciencias Informáticas

Centro de Ideo-Informática (CIDI)

Facultad 1

Sistema para la representación gráfica de estadísticas generadas por el servidor Squid

Autor:

Daniel Ripoll Monteagudo

Tutores:

Ing. Luis Enrique Sánchez Arce

Ing. Dovier Antonio Ripoll Méndez

La Habana, Junio de 2013.

Declaración de autoría

Declaramos que somos los únicos autores del presente trabajo y autorizamos a la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmamos la presente a los ___ días del mes de_____ del año 2013.

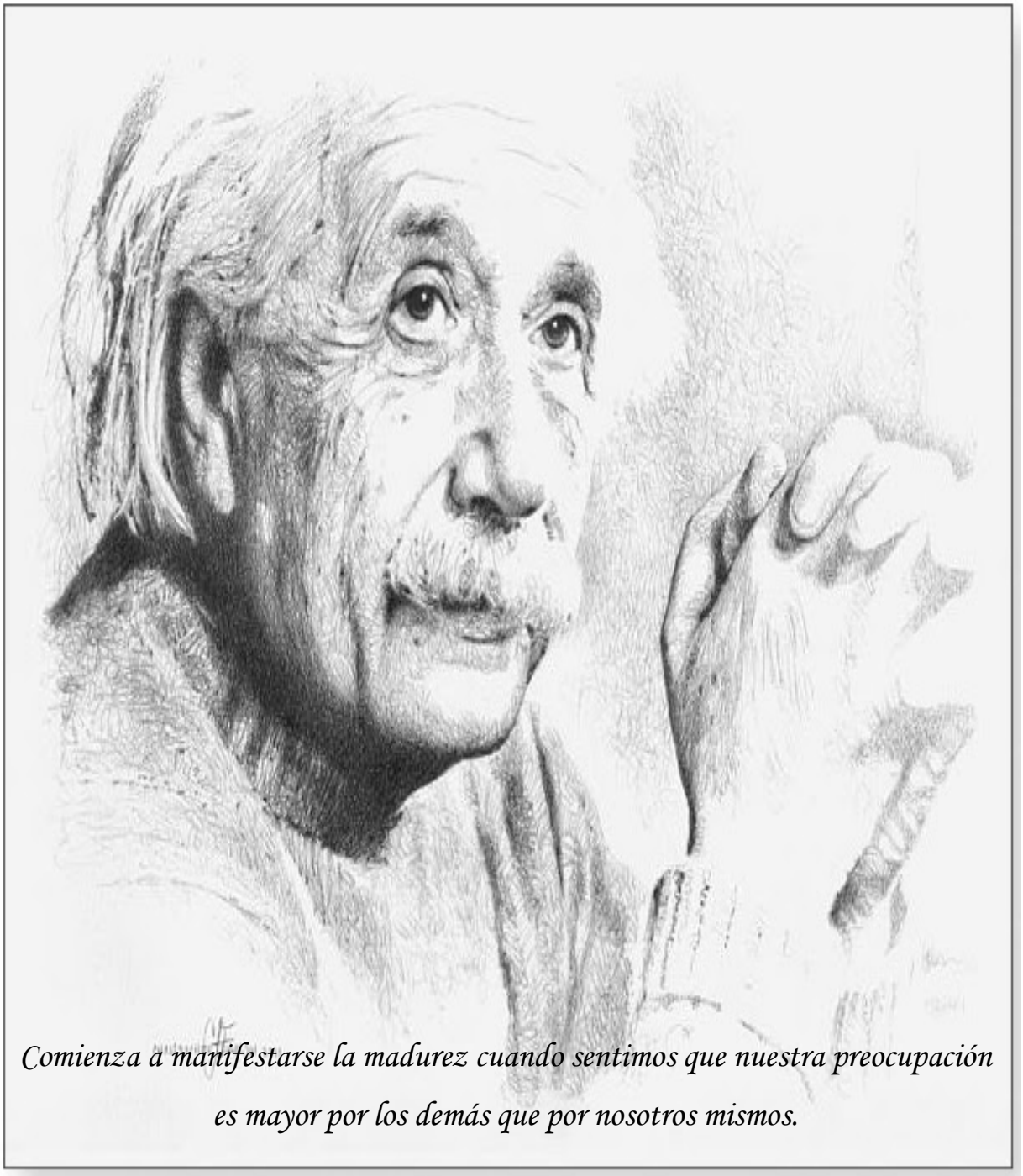
_____

Autor

Tutores:

_____ _____

Tutor principal Co-tutor



Comienza a manifestarse la madurez cuando sentimos que nuestra preocupación es mayor por los demás que por nosotros mismos.

Datos de contacto

Ing. Luis Enrique Sánchez Arce: Ingeniero en Ciencias Informáticas, graduado de 2008. Líder y arquitecto principal del proyecto Filtro de Contenido de Internet. Profesor instructor, 5 años de experiencia.
Correo electrónico: lesanchez@uci.cu.

Ing. Dovier Antonio Ripoll Méndez: Ingeniero en Ciencias Informáticas, graduado de la Universidad de Ciencias Informáticas (UCI) en el 2008 con 5,10 puntos de promedio académico (Título de Oro y Premio Mella). Se desempeñó en la UCI como Jefe del Departamento Soluciones Informáticas para Internet (SINI) y Profesor de la Asignatura Práctica Profesional. Entre sus principales resultados destacan: miembro del Equipo UCI que participó en el X Maratón Regional Suramericano de Programación ACM-ICPC en Venezuela, donde obtuvieron el mejor lugar histórico de Cuba en tales concursos hasta ese momento; autor de publicaciones en la Serie Científica de la UCI y en la Revista de Software Libre UXI; fundador/organizador de la Copa Void de Programación y de la Competencia Estudiantil por Invitación de la UCI; fundador/desarrollador del Proyecto Filtrado de paquetes por contenido (Filpacon), participando en la implantación de dicho producto tanto en Cuba como en Venezuela; autor de trabajos presentados y/o defendidos en eventos científicos, tales como: Jornada Científica de la UCI, Concurso Nacional de Computación, Exposición de las BTJ, Seminario Iberoamericano de Seguridad en las Tecnologías de la Información, entre otros. Ha realizado investigaciones en los siguientes temas: Aprendizaje Colaborativo, Programación Competitiva, Filtrado de Contenido de Internet, Categorización Automática de Textos, Agentes o Robots Web y Cibermetría. Desde inicios del año 2009 se desempeña como Director del ACM-ICPC en el Caribe, por lo cual ha organizado varios eventos del ACM-ICPC en dicha región y ha participado en cuatro Finales Mundiales (Suecia 2009, China 2010, Estados Unidos 2011 y Polonia 2012).

Correo electrónico: daripoll@uci.cu.

Agradecimientos

La ardua escalada hasta la cima para ver realizados mis sueños a pesar de mi sacrificio no hubiese sido posible sin la ayuda de muchas personas a las cuales quiero agradecer en esta pequeña sección:

Mi primer agradecimiento va para mis padres Ana Rosa Monteagudo Ramos y Luis Daniel Ripoll Marrero, las dos personas más importantes e influyentes en mi vida, quienes me han apoyado y dado ánimo en todo momento y que tanto amor y ayuda me han brindado; a ustedes gracias porque sin su guía y apoyo incondicional no hubiese podido llegar hasta aquí.

A mi compañera de vida, amiga y novia Yurisleidy Rodríguez Isla por tanto amor, paciencia y cariño que me ha brindado.

A mis abuelos, tíos, hermanas, primos y al resto de mi familia por el apoyo y confianza que me han brindado siempre.

A Juan y Leonor que los quiero y sé que me quieren como si fuesen mis padres también.

A Jesús (Susena) y Muiño que son como mis abuelos también, quienes me han dado su amor y me han acogido como si fuera su nieto realmente.

A mis tutores Luis Enrique Sánchez y Dovier Antonio Ripoll por toda su ayuda y guía durante el desarrollo del trabajo de diploma y a mi "casi oponente" Yuneisy por su ayuda incondicional.

A mis compañeros de aula y de apartamento por compartir buenos y malos momentos y ayudarme en todo lo que pudieran; especialmente a los más allegados Zaldivar, Osiel , Juan, Alexander, Víctor, Angel, Kmilo, los mellizos, Emilio, Yasmany, Diorges, el niche, Adrián y las niñas Dalianne, Yaremi, Evelyn y María.

Además a mis niñas lindas Danelia, Dariela, Maylen y Yudaymi por cada momento especial que compartimos.

Un agradecimiento especial a Lázaro (Lachy), Leonardo y mi prima Yuri por su ayuda incondicional cada vez que la necesité.

A todos los profesores que de una forma u otra contribuyeron a mi formación como profesional.

A todos muchas gracias.

Dedicatoria

E presente trabajo de diploma se lo dedico:

A mis padres ***Ana Rosa*** *y* ***Luis Daniel*** *quienes son mi razón de ser, los mejores padres del mundo y las personas que más admiro.*

A mi novia ***Yurisleidy****.*

Al resto de mi ***familia****.*

A mis hermanito ***Jesús Eduardo*** *(Susi) y mi abuelo* ***Evelio*** *(abuelo papi), que aunque ya no se encuentren entre nosotros, siempre han tenido y tendrán un lugar especial en mi corazón y mis pensamientos.*

A mis hermanas ***Mabel*** *y* ***Daniela*** *por darme su cariño y amor.*

A mis otros hermanos ***Alejandro*** *(alí),* ***Osbal*** *(el bolo),* ***Dalia*** *(yayi),* ***Cesar*** *y* ***Yasmany*** *que los quiero como si fueran mis hermanos realmente.*

A todas las personas que han confiado en mí y que de una manera u otra me han apoyado y compartido a mi lado en los buenos y malos momentos de mi vida.

Resumen

Para controlar el acceso de los usuarios a Internet se emplean generalmente los sistemas de filtrado de contenidos web. La Universidad de las Ciencias Informáticas no se ha quedado atrás en este sentido y desarrolla desde el año 2005 un filtro de contenido web cuyo componente fundamental lo constituye el servidor proxy Squid. Han existido problemas de funcionamiento en el despliegue de dicho filtro cuya principal causa es la ausencia de una herramienta que brinde la posibilidad de monitorizar en conjunto las características fundamentales del funcionamiento de Squid y del tráfico de los usuarios que acceden a la red a través de él.

La problemática anteriormente planteada justifica la presente investigación por lo que se define como objetivo principal desarrollar un sistema para la representación gráfica de las estadísticas generadas por Squid. Dicho sistema contribuirá a amenizar el seguimiento del desempeño de Squid, elemento importante una correcta toma de decisiones por parte de los administradores de red. Para cumplir con el objetivo principal planteado se realizó un estudio de varias tecnologías, herramientas y lenguajes de programación utilizados en la actualidad para el desarrollo de aplicaciones Web. Las herramientas empleadas facilitaron el desarrollo de un software funcional y eficiente que contribuirá a facilitar las tareas administrativas para el servidor Squid.

Palabras claves: control, filtrado, monitorizar, proxy, squid, web.

Índice

Tabla de contenidos

Índice

Índice

Índice de tablas

Índice de figuras

Índice

Introducción

Introducción

El administrador de un sistema informático tiene la responsabilidad de decidir sobre lo que es mejor o no para el correcto funcionamiento de una aplicación. Una buena toma de decisiones parte de contar con las herramientas adecuadas que le permitan conocer los aspectos más importantes y decisivos que conforman el sistema y su funcionamiento. Comúnmente a este tipo de herramientas se le conoce como generadores de reportes, los cuales pueden ser de información o funcionamiento.

Los reportes de información muestran las principales acciones realizadas por los usuarios. Ejemplo de ello lo constituyen las trazas, donde se aprecia el momento de acceso, el recurso accedido, el usuario que realiza la operación, entre otras. Por su parte, los reportes de funcionamiento brindan estadísticas sobre características puntuales que resumen el funcionamiento del sistema y revelan a partir de indicadores definidos si existen alteraciones. Generalmente dichas estadísticas se representan en forma de gráficos y con valores históricos.

Especialmente los sistemas que brindan servicios a varios usuarios deben cuidar con mayor rigor aspectos como la estabilidad y el correcto funcionamiento. Un ejemplo práctico se puede visualizar en los filtros de contenido, los cuales son herramientas que brindan servicios de acceso regulado a Internet y permiten implementar las políticas de uso aceptable de las Instituciones que los utilizan. Alguna interrupción o mal desempeño de estos sistemas puede ocasionar pérdidas de tiempo y recursos, inconformidades en los usuarios, incumplimiento de las tareas, retraso en los objetivos, entre otros. Por estas razones es de vital importancia que se brinde dicho servicio de la manera más segura y confiable.

En la Universidad de las Ciencias Informáticas (UCI) se desarrolla desde el año 2005 un filtro de contenido web (Smart Keeper) que permite regular el acceso de los usuarios a Internet y cuyo componente fundamental lo constituye el servidor *proxy* Squid. Dicho servidor posee un conjunto de características que lo han convertido en una herramienta de uso común por distintos filtros de contenidos. Actualmente varios centros se apoyan en dicho filtro para regular el acceso a Internet de sus usuarios, entre los que se destacan la UCI, el Ministerio de las Comunicaciones (MC), la sede central de la Unión de Jóvenes Comunistas (UJC), la casa de la Federación Estudiantil Universitaria (FEU) y el Archivo General de la Nación (AGN) del hermano país Venezolano.

Introducción

Squid por su parte proporciona una serie de estadísticas que permiten analizar su funcionamiento y desempeño. Dichas estadísticas son mostradas de manera desorganizada y carentes de elementos visuales que faciliten su lectura, por lo que se hace difícil seguirlas y procesarlas sin ayuda de un sistema que transforme dicha información en tablas o gráficas de fácil entendimiento. Otro elemento a destacar es que mucha de la información ofrecida solo tienen significado para los desarrolladores del sistema con carácter de depuración de errores al mismo tiempo que carecen de importancia para un administrador de red. Se precisaría entonces de especialistas con un alto nivel de conocimientos sobre el tema para poder detectar posibles anomalías.

A partir de la puesta en práctica de Smart Keeper se detectaron problemas en su funcionamiento, reflejado en caídas inesperadas del sistema, ralentización de la red, violaciones de seguridad, consumo innecesario de recursos, sobrecarga de memoria y CPU, afectaciones directas sobre el *hardware,* entre otras.

Estos problemas desencadenaron como resultados negativos, pérdida de productividad, demoras en el envío/recibo de información importante, retraso en el cumplimiento de los objetivos de dichas instituciones, entre otros que conllevaron a una pérdida general de productividad.

Los inconvenientes antes mencionados, incluyendo sus causas y efectos, están dados por la ausencia de una herramienta que brinde (a los administradores de Squid), de manera fácil e intuitiva, información del desempeño de dicho servidor, así como el tráfico de los usuarios por la red. Estas estadísticas juegan un papel importante a la hora de detectar posibles violaciones de seguridad o fallos en el sistema, lo que ayudaría significativamente en la toma de decisiones por parte de los administradores.

Teniendo en cuenta lo planteado anteriormente, se formula como **problema de investigación:** ¿Cómo contribuir a una mejor representación de estadísticas generadas por el servidor Squid para apoyar las decisiones de los administradores?

Donde el **objeto de estudio** es**:** el proceso de gestión de reportes estadísticos, dentro del cual se enmarca como **campo de acción**: la gestión de estadísticas generadas por el servidor Squid.

Para dar solución al problema de investigación se establece como **objetivo general:** Desarrollar un sistema que contribuya a una mejor comprensión de estadísticas generadas por el servidor Squid para apoyar las decisiones de los administradores.

Se puede plantear entonces la siguiente **idea a defender:** Si se desarrolla un sistema que represente

gráficamente, de manera fácil e intuitiva, las estadísticas generadas por el servidor Squid contribuirá a una mejor comprensión y toma de decisiones por parte de los administradores de red.

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

- Caracterizar los diferentes sistemas de generación de reportes estadísticos de funcionamiento disponibles para el servidor proxy Squid.
- Seleccionar las herramientas y tecnologías a utilizar en el desarrollo del sistema propuesto.
- Seleccionar y diseñar los reportes de funcionamiento para el servidor proxy Squid.
- Implementar los reportes de funcionamiento para el servidor Squid seleccionados.
- Validar la implementación de los reportes de funcionamiento para el servidor Squid.

Métodos científicos

Para arribar al resultado final se utilizaron los siguientes métodos teóricos:

El **Analítico-Sintético** se utilizó para la identificación de las principales características de sistemas similares así como la extracción de los elementos más relevantes que debería poseer la aplicación a desarrollar.

El **Histórico-Lógico** permitió obtener una mejor comprensión del estado actual y de la evolución que han tenido los sistemas de generación de reportes estadísticos del servidor proxy Squid.

El método teórico **Modelación** ofrece parte de la información necesaria acerca del objeto que se estudia, se trata de explicar la realidad con la creación de diagramas; los cuales pueden ser presentados en sustitución de la realidad. Mediante su utilización se elaborarán diferentes tipos de diagramas que brindarán información clara sobre el tema de estudio, mediante el descubrimiento de nuevas relaciones y cualidades del objeto de estudio.

El método de **Observación** Se emplea para conocer la situación actual que existe referente a la generación de estadísticas en sistemas web similares.

Estructuración del contenido

Para dar cumplimiento al objetivo general de manera satisfactoria el presente trabajo de diploma estará

estructurado en tres capítulos donde se refleja todo el trabajo investigativo, así como todo lo referente al diseño de la base de datos y la implementación de la solución propuesta, distribuido de la siguiente manera:

Capítulo 1: Fundamentación teórica. Se abordan los conceptos fundamentales que están relacionados con los generadores de reportes. Se realiza un estudio del estado del arte de las principales herramientas que existen nacional e internacionalmente para la generación de reportes del servidor proxy Squid. Se hace referencia a la metodología por la cual se guiará todo el proceso de desarrollo del *software*. Además se presenta la fundamentación de las herramientas utilizadas para el diseño del sistema y las propuestas para su implementación y desarrollo.

Capítulo 2: Características, Análisis y Diseño del sistema. En el capítulo, como parte de la propuesta de solución, se describen las características del sistema, se presenta el modelo del negocio, la especificación de los requerimientos del sistema, determinándose a su vez los casos de uso y los actores. Para los casos de uso más importantes se expone una descripción de los mismos.

En el capítulo se encuentra también lo referente al análisis y diseño del sistema. Como parte de la solución se modelan los diagramas de clases del análisis, los diagramas de clases del diseño, así como los diagramas de interacción correspondientes. En el mismo se exponen los patrones de diseño empleados en la solución, con una breve descripción de sus características y el propósito de su uso en el sistema. Se muestra la estructura del sistema a través de la arquitectura del mismo.

Capítulo 3: Implementación y pruebas. Aborda todo lo relacionado con el flujo de trabajo de implementación. Se expone el diagrama de componente y el estándar de codificación. Además, se tratan las pruebas realizadas para comprobar las funcionalidades del sistema quedando construido y validado el mismo.

1 Capítulo 1

Introducción

Los sistemas generadores de reportes estadísticos son herramientas de una importancia significativa para los administradores de un sistema de *software*. El uso y adecuado entendimiento de tales sistemas contribuye a una correcta toma de decisiones. Por tanto, resulta necesario realizar un estudio del estado del arte en torno a esos sistemas, así como realizar una selección adecuada de las tecnologías y herramientas para el desarrollo de la aplicación.

1.1 Elementos conceptuales

A continuación se explicitan los principales elementos conceptuales que tributan a un mejor entendimiento del objeto de estudio:

Estadística: “La Estadística es la disciplina que le facilita al hombre el estudio de datos masivos, para obtener conclusiones válidas y efectuar predicciones razonables de ellos; y así mostrar una visión de conjunto clara y de fácil apreciación. De forma práctica, la Estadística proporciona los métodos científicos para la recopilación, organización, resumen, representación y análisis de datos, o de hechos, que se presenten a una evaluación numérica; tales como: características biológicas o sociológicas, fenómenos físicos, producción, calidad, población, riqueza, y otros” [1].

La estadística también se define como: “La rama de las matemáticas que describe, analiza e interpreta ciertas características de un conjunto de individuos llamado población. Constituye un elemento clave en el análisis de la información que se recoge de las encuestas, del mismo modo que ayuda a las demás ciencias a generar modelos matemáticos donde se haya considerado el componente aleatorio” [2].

Reportes: Un reporte es un informe o una noticia. Este tipo de documento (que puede ser impreso, digital, audiovisual, etc.) pretende transmitir una información, aunque puede tener diversos objetivos. Existen reportes divulgativos, persuasivos y de otros tipos.

En el ámbito de la Informática, los reportes son informes que organizan y muestren la información contenida en una base de datos. La función de esos reportes es aplicar un formato determinado a los datos para mostrarlos por medio de un diseño atractivo y que sea fácil de interpretar por los usuarios [3].

Capítulo1 | Fundamentación teórica de la investigación

Proxy: El término *proxy*, en el contexto de las redes informáticas, hace referencia a un programa o dispositivo que realiza una acción en representación o en nombre de otro. Su finalidad más habitual es la de servidor *proxy*, que sirve para permitir el acceso a Internet de usuarios de una organización cuando sólo se puede disponer de un único equipo conectado, esto es, una única dirección IP (*Internet Protocol address*).

Cuando un equipo de la red desea acceder a una información o recurso es el *proxy* quien realiza la comunicación y seguidamente traslada el resultado al equipo inicial [4].

Lo anterior se pone de manifiesto del siguiente modo: Si una hipotética máquina A solicita un recurso a una C, lo hará mediante una petición a B; C entonces no sabrá que la petición provino originalmente de A. Esta situación estratégica de punto intermedio suele ser aprovechada para soportar una serie de funcionalidades como: proporcionar caché, control de acceso, registro del tráfico y prohibir cierto tipo de tráfico.

La finalidad más habitual de un servidor *proxy* consiste en interceptar las conexiones de red que un cliente hace a un servidor, por motivos tan variados como: seguridad, rendimiento, anonimato, etc. [5].

Squid: Programa de *software* libre que implementa un servidor *proxy* y un dominio para caché de páginas web. Entre sus principales características se destacan:

- Permite el acceso a través de los protocolos HTTP (hyper text transfer protocol), HTTPS (Hypertext Transfer Protocol Secure), FTP (File Transfer Protocol), entre otros.
- Posibilita un mejor aprovechamiento del ancho de banda al almacenar en caché parte de las peticiones de los usuarios, evitando así recuperar de Internet las páginas más solicitadas.
- Incluye varias utilidades, como por ejemplo: acelerador del servidor web.
- Utilizado por varios proveedores de Internet con el fin de proporcionar a sus usuarios el acceso a la web.
- Publicado bajo la Licencia Pública General de la GNU (GNU/GPL) [6].
- Permite la especificación de otros servidores intermediarios, utilizando para ello la caché en una jerarquía como padres o hermanos; dependiendo de la topología de la red, estos pueden operar en cascada (padres) o en paralelo (hermanos) [7].

Representación gráfica de las estadísticas generadas por Squid: Consiste en obtener ciertas estadísticas de interés que proporciona Squid (con el fin de desechar las que no son importantes en la

administración de red). Luego son mostradas mediante elementos visuales de fácil entendimiento como tablas y gráficas que ayuden a detectar con mayor claridad posibles problemas de funcionamiento o violaciones de seguridad.

1.2 Sistemas generadores de reportes

Con el objetivo de solucionar el problema de investigación se estudió un conjunto de sistemas generadores de reportes similares con el fin de encontrar una solución capaz de cumplir con las siguientes condiciones:

- Proporcionar reportes específicos del desempeño de Squid en el sistema como uso de memoria, de disco, CPU (C*entral Processing Unit*), número de objetos en caché, entre otras con el fin de detectar, de manera fácil e intuitiva, posibles problemas de funcionamiento del *proxy.*
- Proporcionar reportes del tráfico de los usuarios que navegan a través del proxy (recurso accedido, momento de acceso, usuario que accede, velocidad a la que navega, tamaño de descarga, entre otras).
- Es importante que no sea necesario la instalación de varias herramientas para dar cumplimiento a las especificaciones antes mencionadas, ya que esto llevaría a un excesivo consumo de recursos.
- Se tendrá en cuenta también a la hora de la selección, que existe la posibilidad de integrar esta aplicación a la próxima salida de la versión 3.0 de Smart Keeper por lo que se estudiará la posibilidad de su integración a dicho filtro de contenido web.

1.2.1 Ámbito internacional

A nivel internacional existen varios sistemas generadores de reporte para Squid, dentro de los que se destacan:

Calamaris

Principales características:

- Permite analizar los *logs* generados por el servidor *proxy* Squid.
- Genera informes tanto en texto plano como en páginas HTML, que incluyen estadísticas que son ordenadas por tablas. Existen versiones de Calamaris que generan gráficos de estas estadísticas.
- Herramienta disponible bajo licencia GNU/GPL.

- Codificado en el lenguaje de programación Perl [8].
- Muy útil para especialistas de redes debido a la cantidad de información que puede mostrar.

Se considera que Calamaris tiene como limitante que sus reportes son muy generales y enfocados a la administración de la navegación aunque posee un grupo de funcionalidades que pueden ser utilizadas en la propuesta de solución y en general se considera a Calamaris como un buen punto de partida a la hora de tomar ideas pues representa las estadísticas de una forma visual que resulta atractiva para el usuario.

Sarg

Principales características:

- Lleva un control de la navegación que realizan los usuarios en Internet.
- Proporciona información acerca de las actividades de los usuarios que navegan a través del servidor *proxy*: tiempo de navegación, bytes consumidos, sitios visitados, entre otras.
- Construye estadísticas que son específicas a cada usuario.
- Genera reportes en diferentes idiomas (internacionalización).
- Herramienta disponible bajo licencia GNU/GPL.

Sarg está configurado para generar periódicamente reportes web del uso de Internet. Además, los administradores pueden ejecutarlo manualmente para generar reportes de fechas, usuarios o dominios en específico [9]).

El sistema genera informes en formato HTML. Posee un sencillo ambiente de trabajo, recomendado por el alto nivel de usabilidad que presenta y debido a la facilidad con que se puede operar. Una desventaja de la herramienta es que, debido a la cantidad de información que manipula, puede llegar a funcionar con lentitud. Además, no cumple con todas las exigencias determinadas; no obstante, se puede tomar como referencia a la hora de diseñar la solución propuesta.

Logrep

Principales características:

- Herramienta disponible bajo licencia GNU/GPL.
- Centraliza los logs de los servidores proxy, tanto en ambientes GNU/Linux como en Windows.
- Presenta los logs en formato HTML.
- El cliente es capaz de recolectar logs de 30 sistemas diferentes, entre los que se encuentran:

Snort, Squid, Postfix, Apache, Syslog, Ipchains, Qmail, Sendmail, Iptables, Servidores Windows, Firewall-1, Wtmp, Xferlog, Oracle y Pix.

- Utiliza un protocolo seguro de comunicación.
- Guarda copias de todos los logs en un sistema central.
- Sus logs se muestran gráficamente [10].

Logrep es una herramienta ideal para administradores de sistemas que necesiten centralizar la gestión de logs y tener un sistema de acceso rápido a todos ellos. Una de las principales limitantes de dicho sistema es que cuenta con escasa documentación, dificultando su estudio. Otro elemento negativo de esta herramienta es que no se centra únicamente en Squid, lo cual dificulta la interconexión desde el punto de vista técnico.

Sawmill

Principales características:

- Herramienta de análisis jerárquico optimizado para reportes web.
- Soporta 818 formatos de log.
- Genera y agrega informes de filtrado de forma dinámica, todos a través de una interfaz web [11].

El hecho de ser un *software* altamente configurable propicia significativas facilidades de uso. Dicho sistema posee dinamismo a la hora de filtrar la información procesada de los registros, lo cual permite extraer estadísticas que no pueden obtenerse con otros analizadores de logs. Posee una estructura y características que son de interés para la solución propuesta, aunque la principal limitante es que Sawmill está disponible bajo licencia propietaria por lo que sería muy costosa su instalación y mantenimiento. No obstante no se debe pasar por alto pues posee varias funcionalidades de interés.

Internet Access Monitor (IAM)

Principales características:

- Supervisa el uso de Internet y genera reportes para redes corporativas.
- Registra las acciones realizadas por los usuarios.
- Procesa los logs para ofrecer varias opciones de construcción de reportes.
- Construye reportes para usuarios individuales, mostrando una lista de sitios web visitados y la actividad de Internet (descargas, lectura de textos, fotografías, películas, música, etc.).
- Posibilita la creación de reportes con análisis del consumo total de ancho de banda.

- Construye tablas de fácil entendimiento que sugieren las áreas donde se puede aprovechar mejor el ancho de banda [12].

Se considera que la mayor limitante o desventaja de IAM es que muestra solo una pequeña parte de los datos que se necesitan para un buen control (es muy específica).

Internet Access Control (IAC)

Principales características:

- Ofrece visualmente y en tiempo real algunas estadísticas de monitorización y facturación, entre otras.
- Constituye una de las principales herramientas para el control, el bloqueo y la restricción de Internet.
- Limita el acceso a Internet en horarios específicos.
- Protege por contraseña la conexión a Internet y la red.
- Configura límites de uso diario de Internet aplicados a todos los usuarios o solo a usuarios en específico [13].

Una de las principales limitantes de esta herramienta es la dificultad en su instalación, configuración y uso. Por otro lado, no muestra información sobre su desempeño en el sistema (solo el tráfico en la red), elemento importante para evitar caídas o funcionamientos anormales del sistema.

Munin

Principales características:

- Resulta muy útil para observar estadísticas de uso de los recursos, tales como: memoria, disco duro y servicios.
- Utiliza la herramienta RRDtool para generar gráficas de rendimiento de los parámetros analizados del sistema.
- Herramienta flexible que se utiliza para crear gráficos de datos de la red, realizando monitoreo periódico (cada cinco minutos).
- Utiliza una interfaz web para mostrar las gráficas generadas, permitiendo el trabajo de manera distribuida y mostrando a su vez la información de varios servidores.
- Soporta varios *plugins*, por lo que se considera como una herramienta muy versátil [14].

Munin posee la dificultad de recolectar información sin requerir una previa autenticación en la interfaz web, lo cual posibilita un acceso no autorizado a información que puede catalogarse como sensible o confidencial.

Pandora

Pandora Free Monitoring System (FMS) es un sistema encargado de monitorizar servidores y servicios mediante un agente que se instala en la máquina a monitorizar. Permite analizar (de forma visual y mediante un navegador web) el rendimiento y estado de algunos parámetros en diferentes sistemas operativos, servidores, aplicaciones y sistemas de hardware [15]. Entre las características principales de Pandora FMS se destacan:

- Monitoreo remoto y multiplataforma (Solaris, GNU/Linux, Windows, IPSO, AIX, HP-UX).
- Posee una arquitectura Cliente-Servidor.
- Alta capacidad de procesamiento.
- Gestión de alertas adaptables.
- Genera informes personalizados.

La principal desventaja de esta herramienta es que consume muchos recursos pues visualiza parámetros de diferentes sistemas, provocando que sea más lento. Resulta de interés obtener un sistema que se centre solo en el funcionamiento de Squid.

Nagios

Principales características:

- Herramienta disponible bajo licencia GNU/GPL.
- Resulta útil en la monitorización de servicios y *host* de redes, tanto locales como remotas.
- Ofrece datos sobre cuáles ordenadores y dispositivos están activos y fallando.
- Permite el monitoreo de servicios de red (SMTP, POP3, HTTP, NNTP, PING, entre otros).
- Supervisa recursos de un equipo (carga en el procesador, uso de disco duro, entre otros).
- Posee un diseño simple en forma de *plugins,* permitiendo a los usuarios un desarrollo fácil de sus propias verificaciones de servicios.
- Posibilita una monitorización remota a través de túneles SSL/SSH e incluye la habilidad de definir una jerarquía de los equipos de la red.
- Envía notificaciones, vía correo electrónico o algún otro método definido por el usuario, a contactos cuando un servicio o equipo presenta problemas.

- Brinda soporte para implementar equipos redundantes en temas de monitoreo.
- Tiene opcionalmente una interfaz web para ver el estado actual de la red, notificaciones, historial de problemas, archivo log, entre otros [17].

Una de las principales limitantes de esta herramienta es la dificultad en su instalación, configuración y uso. Además cuando es monitorizada una red amplia, los requerimientos de hardware que demanda son bastante exigentes.

1.2.2 Ámbito nacional

Durante la investigación, se realizó una búsqueda de los sistemas de reportes de funcionamiento en el país. Solo se encontró información de la existencia del desarrollo de un sistema de similar que es el del trabajo de diploma realizado por Thais Londres Viltres y Reinier Rodríguez Lobaina graduados en la UCI en el año 2011.

Sistema de reportes de funcionamiento de Filpacon v2.0

Muestra, mediante gráficas, un historial del comportamiento de Squid mediante reportes solicitados periódicamente al proxy. Es un buen punto de partida para el desarrollo de una parte del sistema deseado pero la información que brinda es muy escasa**,** además no cuenta con todas las funcionalidades deseadas en el sistema a desarrollar como por ejemplo reportes de tráfico en la red, importante para un buen control del tráfico en la red. Otra limitante que presenta es que no es un sistema independiente, se realizó en forma de módulo para integrarlo a Filpacon v2.0.

En la actualidad las empresas nacionales, en dependencia de sus necesidades, utilizan las herramientas existentes a nivel mundial. En el ámbito local de la UCI se utiliza en el nodo central el sistema de monitorización Nagios mediante una de sus interfaces, Centreon. La selección de Nagios está sustentada por la cantidad de servicios y estaciones de trabajos que se necesitan monitorizar. Además, permite la generación de gráficos en tiempo real e informes detallados de los eventos que se deseen monitorizar, detectando las paradas de descargas y las anomalías mucho antes que los usuarios [16].

1.3 Metodología de desarrollo

Las metodologías de desarrollo de *software* surgen ante la necesidad de utilizar una serie de procedimientos, técnicas, herramientas y soporte documental a la hora de desarrollar un *software* [18]. Cuentan con un conjunto de pasos y procedimientos que deben seguirse. Una metodología responde a las

siguientes interrogantes [19]:

- ¿Cómo dividir un proyecto en etapas?
- ¿Qué tareas se llevan a cabo en cada etapa?
- ¿Qué restricciones deben aplicarse?
- ¿Qué técnicas y herramientas se emplean?
- ¿Cómo se controla y gestiona un proyecto?

En el Centro de Ideo-Informática (CIDI), donde se realiza el presente Trabajo de Diploma (TD), se han definido dos metodologías para guiar el desarrollo de *software* (Open UP y RUP). El autor decide utilizar la primera por las siguientes razones:

- Posee una elevada capacidad de respuesta a cambios de requisitos a lo largo del desarrollo.
- Permite un trabajo conjunto entre el cliente y el equipo de desarrollo, muy importante para conseguir el objetivo trazado.
- Es muy simple, pues elimina el trabajo innecesario.
- Se centra más en el resultado final deseado por el cliente que en el proceso poco flexible de planificación de las metodologías “pesadas” [20].
- Se encarga de valorar al individuo y a las iteraciones del equipo más que a las herramientas o los procesos utilizados.
- Es más prioritario crear un producto de *software* funcional que escribir mucha documentación, en ocasiones innecesarias.
- El cliente está en todo momento colaborando en el sistema [17].

1.3.1 Open UP

Constituye un Proceso Unificado de desarrollo de corta duración, aplicado de manera iterativa e incremental dentro de un ciclo de vida estructurado en tres capas. Open UP adopta una pragmática y ágil filosofía centrada en el proceso colaborativo de desarrollo de *software* que puede ser aplicado a una variedad considerable de proyectos en diferentes direcciones.

El Open UP es un proceso mínimo y suficiente, lo que significa que solo el contenido fundamental y necesario es incluido. Por tanto, no provee lineamientos para todos los elementos que se manejan en un proyecto, pero tiene los componentes básicos que pueden servir de base a procesos específicos. La

mayoría de los elementos de Open UP están declarados para fomentar el intercambio de información entre los equipos de desarrollo y mantener un entendimiento compartido del proyecto, incluyendo sus objetivos, límites y avances.

Otras características:

- En un proyecto organizado sobre Open UP, los esfuerzos personales se convierten en micro incrementos.
- Tales incrementos proveen un ciclo de retroalimentación relativamente corto que permite flexibilidad y mejor adaptación a las decisiones tomadas dentro de cada iteración.
- Esta metodología divide el proyecto en iteraciones que son planeadas sobre intervalos de tiempo definidos en semanas, dichas iteraciones se centran en el cumplimiento de los objetivos definidos previamente en el plan para cada una.
- Open UP, en cada iteración del ciclo de vida, incrementa progresivamente los objetivos de las iteraciones anteriores añadiendo nuevas funcionalidades a las versiones estables del *software* que se tiene hasta el momento.
- Dentro del ciclo de vida tiene 4 fases: Intercepción, Elaboración, Construcción y Transición [21].

En la UCI se han obtenido resultados importantes utilizando Open Up para agilizar la producción de *software*. Por otra parte, el sistema propuesto no constituye un *software* de alta complejidad ni está determinado por procesos críticos donde se necesite poseer documentación sobre su ciclo de desarrollo.

1.4 Herramientas CASE

Las herramientas *CASE (Computer Aided Software Engineering)*, en español Ingeniería de *Software* Asistida por Computadoras, son una serie de aplicaciones informáticas cuya función consiste en aumentar la productividad en el desarrollo de *software*, lo que a la postre significa un ahorro de tiempo y dinero. Tales herramientas son muy útiles en todo el ciclo de desarrollo del *software* pues ayudan en la realización del diseño, la implementación de parte del código a partir del diseño, compilación automática, detección de errores, entre otras.

1.4.1 Visual Paradigm 8.0

Es una herramienta CASE multiplataforma con licencia libre, capaz de generar diferentes tipos de diagramas que pueden ser exportados como imágenes en formato JPG, PNG, entre otros. Proporciona a

los desarrolladores una plataforma con interfaz amigable que les permite diseñar de forma muy rápida un producto con calidad. Presenta soporte al diseño personalizado, permitiendo incorporar nuevas formas y notaciones, mediante el uso de imágenes o iconos importados.

Principales Características:

- Fácil de instalar y actualizar.
- Tiene capacidad de ingeniería directa e inversa.
- Soporta UML (*Unified Modeling Language*) y ORM (*Object-Relational mapping*).
- Brinda funcionalidades para la generación de código fuente a partir de diagramas de clases.
- Diseñado para una amplia gama de usuarios interesados en la construcción de sistemas de *software* de forma fiable, a través de la utilización de un enfoque de POO (Programación Orientada a Objetos).

1.5 Sistema Gestor de Bases de Datos (SGBD)

Un SGBD es una colección de datos interrelacionados y un conjunto de programas para acceder a ellos. El Objetivo primordial de un SGBD es proporcionar un entorno que sea conveniente y eficiente para ser utilizado al extraer y almacenar información de la base de datos.

1.5.1 PostgreSQL-v9.1

Es un gestor de bases de datos relacional orientado a objetos, libre y gratuito. Está liberado bajo la licencia BSD (*Berkeley Software Distribution*), lo que significa que se puede disponer de su código fuente, modificarlo a voluntad y redistribuirlo libremente. Su utilización está dada principalmente debido a que Smart Keeper lo utiliza para la gestión y almacenamiento de datos, con esto se evita la utilización de otra base de datos. La selección de PostgreSQL está sustentada además por sus características, las cuales se mencionan a continuación.

Principales Características:

- Ampliamente utilizado en el trabajo con tecnologías Web.
- Fácil de Administrar.
- Su sintaxis SQL es estándar y fácil de aprender.
- Multiplataforma.
- Capacidades de replicación de datos.

- Tiene interfaces nativas de programación para C/C++, Java, .Net, Perl, Python, Ruby, Tcl, ODBC y otros.
- Amplia documentación
- Funciona muy bien con grandes cantidades de datos y una alta concurrencia de usuarios accediendo a la vez al sistema.
- Cuenta con numerosos tipos de datos y la posibilidad de definir nuevos.
- Soporta el almacenamiento de objetos binarios grandes (gráficos, vídeos, sonido) [22].

1.5.2 Base de Datos RRDtool-v1.4.7 (*Round Robin Database Tool*)

A partir del análisis de los sistemas homólogos se pudo detectar que la base de datos RRD es un punto común para realizar gráficos de funcionamiento.

RRDtool se emplea para el almacenamiento de datos, tales como: el tráfico de red, temperatura, carga del CPU, entre otros. Además, permite definir la resolución de los datos y generar gráficos (en imágenes de formatos PNG y GIF) con atributos definidos por el usuario. RRDtoolb posee una cantidad fija de datos y un puntero al elemento actual. El modo en que trabaja una base de datos utilizando *Round Robin* es el siguiente: una vez alcanzada la capacidad de la base de datos, esta es tratada como si fuera una lista circular sobrescribiendo los datos almacenados (la capacidad de la base datos depende de la cantidad de información como historial que se desee conservar).

Puede almacenar prácticamente cualquier tipo de datos, siempre que se trate de una serie temporal de datos. Esto permite poder realizar medidas en algunos puntos de tiempo y proveer esta información a la RRDtool para que la almacene satisfactoriamente.

Un concepto que está vinculado con RRDtool es el de SNMP (*Simple Network Management Protocol*). Protocolo utilizado por varias herramientas para monitorizar el estado de algún dispositivo o servicio. Además, permite a sus administradores inspeccionar el funcionamiento del dispositivo que se analice para así buscar y solucionar cualquier problema que surja [23].

Fue seleccionada esta herramienta por su flexibilidad, versatilidad y facilidad de programación. Además, por lo práctica que resulta para guardar estadísticas en series de tiempo y la particularidad de que las base de datos RRD (en lo adelante BD RRD) que esta genera no crecen con el paso del tiempo, así como que permite guardar información en la generación de gráficas. Por último, esta herramienta es utilizada de

manera común por la mayoría de los sistemas homólogos que fueron analizados con anterioridad.

1.6 *Framework* de desarrollo

En el desarrollo de *software*, un *framework* es una estructura que simplifica el desarrollo de una aplicación mediante la automatización de algunos de los patrones utilizados para resolver las tareas comunes, haciendo uso de buenas prácticas de programación. Proporciona estructura al código fuente, forzando al desarrollador a crear código más legible, más fácil de mantener y facilita la implementación de aplicaciones, ya que encapsula operaciones complejas en instrucciones sencillas [24].

1.6.1 Symfony 2.1.7

Symfony ha sido probado en varios proyectos reales y se utiliza, por ejemplo, en sitios web de comercio electrónico de primer nivel. Es compatible con varios gestores de bases de datos, tales como: MySQL, PostgreSQL, Oracle y SQL Server. La posibilidad existente de que la solución propuesta se pueda acoplar con la versión 3.0 de Smart Keeper (filtro de contenidos desarrollado en la UCI) fue uno de los aspectos que se consideraron en la elección de este *framework* [25].

Principales características:

- Fácil de instalar y configurar en la mayoría de las plataformas.
- Independiente del Sistema Gestor de Bases de Datos (SGBD). Su capa de abstracción y el uso de Doctrine permiten cambiar con facilidad de SGBD en cualquier fase del proyecto.
- Utiliza la POO.
- Fácil de usar o aplicar.
- Aunque utiliza MVC (Modelo-Vista-Controlador), tiene su propia forma de trabajo con variantes de ese patrón arquitectónico, como son: la capa de abstracción de base de datos, el controlador frontal y las acciones.
- Basado en la premisa de “convenir en vez de configurar”, en la que el desarrollador sólo debe configurar aquello que no es convencional.
- Preparado para aplicaciones empresariales y adaptables a las políticas y arquitecturas propias de cada empresa, además de ser lo suficientemente estable como para desarrollar aplicaciones a largo plazo.
- Código fácil de leer que incluye comentarios de *phpDocumentor* y que permite un mantenimiento

no muy exigente.

- Fácil de extender, lo que permite su integración con bibliotecas de otros fabricantes.
- Posee una potente línea de comandos que facilita la generación de código, contribuyendo al ahorro de tiempo y esfuerzo de trabajo.

Es analizada solo esta herramienta debido a que se planea su futura integración a la próxima salida de la versión 3.0 de Smart Keeper que es desarrollada utilizando el *framework* mencionado.

1.7 Lenguajes

El framework de desarrollo Symfony tiene predefinido varios lenguajes y tecnologías para el desarrollo de las aplicaciones web, entre los que se encuentran:

1.7.1 PHP 5-v5.4.9 (*Hypertext Pre-processor*)

PHP es un lenguaje interpretado, de alto nivel, embebido en páginas HTML que se ejecuta en el lado del servidor. Ofrece la integración con varias bibliotecas externas, que permiten lograr diversas funcionalidades [26].

Principales características:

- Protección contra diversos ataques a través de diferentes niveles de seguridad.
- Sofisticado manejo de variables que lo hacen muy robusto y estable.
- Multiplataforma (GNU/Linux, Unix, Solaris, Mac OS, Windows, entre otros).
- Alta velocidad de ejecución sin introducir demoras en la máquina, a la vez que un bajo consumo de recursos y una adecuada integración con Apache.
- Lenguaje *Open Source*.
- Capacidad de expandir su potencial utilizando una amplia gama de módulos (llamados ext's o extensiones).
- Capacidad de leer y manipular datos desde diversas fuentes.
- Permite la aplicación de técnicas de POO, así como la creación de formularios para la Web.
- Alternativa de fácil acceso, debido a que es *software* libre.
- Permite crear páginas dinámicas de maneras rápida y fácil; además de poseer una amplia documentación en su sitio web oficial.

Además de sus características y ventajas, la utilización de PHP está dada a que el sistema propuesto se desarrollará en *Symfony*.

1.7.2 HTML 5 (*HyperText Markup Language*)

HTML es el lenguaje de marcado para el desarrollo de páginas web; se utiliza en la traducción y descripción de la estructura y la información en forma de texto, así como para complementar el texto con objetos tales como imágenes.

HTML5, la quinta actualización de HTML, el lenguaje en el que es creada la web. También es un término de marketing para agrupar las nuevas tecnologías de desarrollo de aplicaciones web (que son HTML5, CSS3 y Javascript). Aunque aún se encuentra en modo experimental, los principales navegadores, en sus últimas versiones reconocen muchos de los elementos que aporta. Entre sus características resalta la reducción de la necesidad de plugins externos y un mejor manejo de errores.

HTML5 establece una serie de nuevos elementos y atributos que reflejan el uso típico de los sitios web modernos. Algunos de ellos son técnicamente similares a las etiquetas y , pero tienen un significado semántico, como por ejemplo <nav> (bloque de navegación del sitio web) y <footer>. Otros elementos proporcionan nuevas funcionalidades a través de una interfaz estandarizada, como los elementos <audio> y <video>.

Mejoras en el elemento <canvas>, capaz de renderizar en los navegadores más importantes (Mozilla, Chrome, Opera, Safari e IE) elementos 3D.

Algunos elementos de HTML 4.01 han quedado obsoletos, incluyendo elementos puramente de presentación, como <font> y , cuyos efectos son manejados por el CSS. También hay un renovado énfasis en la importancia del scripting DOM para el comportamiento de la web.

1.7.3 JavaScript

Es un lenguaje de programación que se utiliza principalmente para crear páginas web dinámicas, siendo ésta una de las principales ventajas que presenta sobre el HTML; además, es muy fácil de aprender. Técnicamente es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos. Permite la integración directamente con páginas HTML. Los programas escritos en JavaScript se pueden probar directamente en cualquier navegador sin necesidad de procesos

intermedios. A pesar de su nombre, no guarda ninguna relación directa con el lenguaje de programación Java.

Otras características:

- Es interpretado por el navegador web del cliente.
- Está basado en objetos; por lo cual emplea herencia, técnica que es típica de la (POO).
- No es necesario declarar los tipos de variables.
- Las referencias a objetos se comprueban en tiempo de ejecución, por lo tanto no se compila [27].

Es utilizado para otorgarle dinamismo a ciertos reportes que son consultados en tiempo real como por ejemplo tráfico en la red.

1.7.4 Bash-v4.2

Bash (*Bourne-Again Shell*) es un intérprete de comandos de tipo Unix (*shell*) escrito para el proyecto GNU. Es el *shell* por defecto en la mayoría de las distribuciones de GNU/Linux. Se encarga de interpretar las órdenes para su proceso por el kernel (núcleo de ese sistema operativo). Bash es el responsable de interpretar y ordenar que la ejecución de los comandos [28].

Principales características:

- Ejecución síncrona (una a continuación de la otra) o asíncrona (en paralelo) de órdenes.
- Distintos tipos de redirecciones de entradas y salidas para el control y filtrado de la información.
- Control del entorno de los procesos.
- Ejecución interactiva y desatendida de mandatos, aceptando entradas desde teclado o desde ficheros.
- Lenguaje de programación de alto nivel que incluye distinto tipos de variables, operadores, matrices, estructuras de control de flujo, entrecomillado, sustitución de valores y funciones.
- Control de trabajos en primer y segundo planos.
- Edición del histórico de mandatos ejecutados.
- Posibilidad de usar una *shell* para el uso de un entorno controlado.

El lenguaje Bash es seleccionado para realizar algunas acciones en el sistema propuesto (como el trabajo con ficheros), con el objetivo de mejorar su rapidez. Por otra parte, casi todos los comandos para utilizar la

base de datos RRD (*Round Robin Database*) se ejecutan desde la consola (para la utilización del programa Squidclient) y en ello es muy práctico el uso del lenguaje Bash.

1.8 Tecnologías

Para el desarrollo de *software* es necesario desde el inicio seleccionar correctamente las tecnologías a utilizar, esto favorece y evita que: (I) el desarrollo consuma más tiempo del necesario, (II) aumenten los costos y (III) disminuya la calidad del producto.

1.8.1 Entorno Integrado de Desarrollo (IDE)

Las principales características de un Entorno Integrado de Desarrollo (IDE) son:

- Permite a los usuarios la creación rápida de *software*.
- Permite el formateo de código.
- Reduce el tiempo de trabajo al sugerir y luego autocompletar el código.
- Posee funciones para renombrar variables, lo que se conoce como refactorización.
- Buscador de contenidos, nombres de archivo y extensiones.
- Advertencias y errores de sintaxis en pantalla al interpretar o compilar.
- Permite crear proyectos para visualizar los archivos de manera gráfica.
- Brinda ayuda a las pruebas unitarias.

NetBeans es un IDE para que los programadores puedan escribir, compilar, depurar y ejecutar programas. Puede servir para una amplia variedad de lenguajes de programación. Además, existe un número importante de módulos para extenderlo [29]. Entre sus principales características destacan:

- Producto de licencia libre.
- Posee un completamiento de código y resaltado de opciones.
- Incluye soporte para Java, PHP, C/C++, Ruby on Rails, Groovy, Python y JavaScript.
- Enlaza datos con el Swing GUI.
- Tiene características visuales para el desarrollo web.
- Posee mejoras para SOA y UML.
- Es Multiplataforma.
- Puede ser integrado con Symfony 2.

- Soporta PHP 5, CCS 3 y HTML 5.

NetBeans IDE 7.3 (versión utilizada en el desarrollo del sistema propuesto) ofrece un rendimiento mejorado y una mayor experiencia de codificación y autocompletado para páginas .twig. También incluye características notables, como la integración con el generador de escena para la creación visual de formas JavaFX y apoyo a múltiples marcos de PHP [30].

La selección del IDE estuvo determinada (además de las características antes mencionadas) por ser fácil de instalar y usar. Además, porque es una herramienta apoyada por una amplia comunidad de desarrolladores, ofreciendo una documentación completa y variados recursos de capacitación.

1.8.2 Bootstrap 2.0

Bootstrap 2.0 es un *framework* diseñado para simplificar el proceso de creación de diseños web. Para ello ofrece una serie de plantillas CSS y de ficheros JavaScript que permiten crear interfaces web que funcionen en los navegadores existentes. Posee una mejor integración con las librerías usadas habitualmente, como por ejemplo jQuery; además de un diseño sólido basado en herramientas actuales y potentes.

1.8.3 CSS 3 (*Cascading Style Sheet*)

CSS es un lenguaje para definir el estilo o la apariencia de las páginas web escritas con HTML o de los documentos XML. Permite separar el contenido de la forma, a la vez que proporciona a los diseñadores el mantenimiento de un control más preciso sobre la apariencia de las páginas web. Su novedad más importante consiste en la incorporación de mecanismos para mantener un mayor control sobre el estilo con el que se muestran los elementos de las páginas.

1.9 Servidor web Apache 2-v2.2.22

Los servidores web permiten alojar sitios para ser accedidos por los clientes utilizando un navegador que se comunica con el servidor utilizando el protocolo HTTP (HyperText Transfer Protocol). Básicamente, un servidor web consta de un intérprete HTTP que se mantiene a la espera de peticiones de clientes y le responde con el contenido según sea solicitado. El cliente, una vez reciba el código, lo interpreta y lo exhibe en pantalla [31].

Apache es un servidor web flexible, rápido y eficiente, al mismo tiempo que proporciona interfaz a un número significativo de bases de datos. Facilita la integración con *plugins* de los lenguajes de programación de páginas web dinámicas, además de poseer una interfaz con la mayoría de los sistemas de autenticación. En la actualidad es uno de los servidores HTTP más utilizados. Entre sus principales características destacan:

- Multiplataforma.
- Servidor de web conforme al protocolo HTTP/1.1.
- Modular. Puede ser adaptado a diferentes entornos y necesidades.
- Su versión 2 incluye elementos de multi-hilos.
- Incentiva la retroalimentación de los usuarios, con lo cual se obtienen nuevas ideas, informes de fallos y parches para la solución de los mismos.
- Se desarrolla en comunidad (de forma abierta).
- Extensible. Debido a su característica de ser modular se han desarrollado diversas extensiones entre las que destaca PHP, un lenguaje de programación del lado del servidor [32].

1.10 Conclusiones parciales

- Los sistemas homólogos descritos en el estudio realizado no constituyen una solución al problema identificado por lo que se decide desarrollar un sistema que represente gráficamente estadísticas sobre el funcionamiento de Squid y del tráfico de los usuarios que navegan a través de él.
- En el análisis de esos sistemas sale a relucir una serie de características que debe tener el producto a desarrollar, tales como varios reportes comunes para estos, y que no pueden faltar en el sistema propuesto. Otro elemento a destacar es la utilización de la BD RRD para generar reportes gráficos; esta constituye una herramienta ideal para el control de valores temporales con un mínimo de consumo.
- La elección del *framework* Symfony ayudó a determinar la mayor parte de los lenguajes de programación, tecnologías y patrones a utilizar.

2 Capítulo 2

Introducción

En el presente capítulo se describe el proceso llevado a cabo durante la fase de análisis y diseño; en el cual se definen los Requisitos Funcionales (RF) y No Funcionales (RNF) del *software*. Paso es importante, pues el éxito del *software* se define a partir de una buena identificación de los requerimientos. A partir de los RF se identifican y describen los casos de uso y actores. Como parte del proceso que se realiza durante esta fase se generan los diferentes diagramas y artefactos correspondientes para dar cumplimiento a los objetivos trazados. Además se especifica la estructura física de la solución, realizándose una descripción de las principales clases del sistema y de los componentes existentes en la solución propuesta.

2.1 Descripción del problema

Los filtros de contenido de Internet son herramientas de relevante importancia para empresas, centros de estudio, hospitales y en general cualquier institución que brinde servicios de acceso a la Red de Redes. Estas herramientas apoyan la implementación de las políticas de uso adecuado del Internet en la empresa, mediante el control del acceso de sus usuarios a dicho recurso. La mayoría de tales filtros generan reportes sobre su funcionamiento y el tráfico de los usuarios en Internet. Dichos reportes facilitan la toma de decisiones por parte de administradores, evitando posibles fallos o comportamientos anómalos del sistema.

En el capítulo anterior se reflejó la necesidad de contar con un sistema generador de reportes para Squid que permita mostrar los principales reportes existentes de una forma más comprensible. En la actualidad, el generador de reportes utilizado por Smart Keeper muestra solamente una pequeña parte de los necesarios; por ejemplo, no visualiza información respecto al tráfico de los usuarios a través del servidor.

2.2 Solución propuesta

La propuesta de solución a la problemática planteada consiste en elaborar una aplicación con las siguientes características:

- Capaz de generar gráficos con los datos almacenados en las BD RRD y que se relacionan con el funcionamiento y rendimiento de Squid, así como otros datos importantes (cantidad de usuarios conectados, cantidad de conexiones concurrentes, entre otras).
- Capaz de generar, en tiempo real, tablas e imágenes que representen datos solicitados a Squid (tráfico en la red, servidores padre, ACLs externas, entre otros).
- Las BD RRD son las encargadas de almacenar los valores de cada parámetro a seguir.
- La interfaz web es la encargada de mostrar las gráficas y tablas creadas, así como de proporcionar la opción de filtrar dichas gráficas en un rango de tiempo deseado por el administrador (última hora, último día, última semana o último mes).
- Para almacenar los datos de Squid en las BD RRD se hará uso del planificador de tareas CRON, que en intervalos de 5 minutos (tiempo que utilizan la mayoría de los sistemas homólogos) ejecuta los *scripts* encargados.
- Gestión de usuarios por parte de los administradores con ese nivel de privilegio.

Para facilitar la comprensión del sistema propuesto, a continuación se presenta el Modelo del Dominio.

2.3 Modelo del Dominio

El Modelo de Dominio es una representación visual estática de los objetos del sistema; o sea, es un diagrama con los objetos (reales) que existen relacionados con el proyecto que se va a acometer y las relaciones que existen entre ellos. No son clases de *software* (aunque algunos objetos del Modelo de Dominio pueden llegar a serlo). Tiene como objetivo principal ayudar a comprender los conceptos que utilizan los usuarios y con los que debe trabajar la aplicación [33]. A continuación se muestra el Modelo de Dominio referente a la solución propuesta:

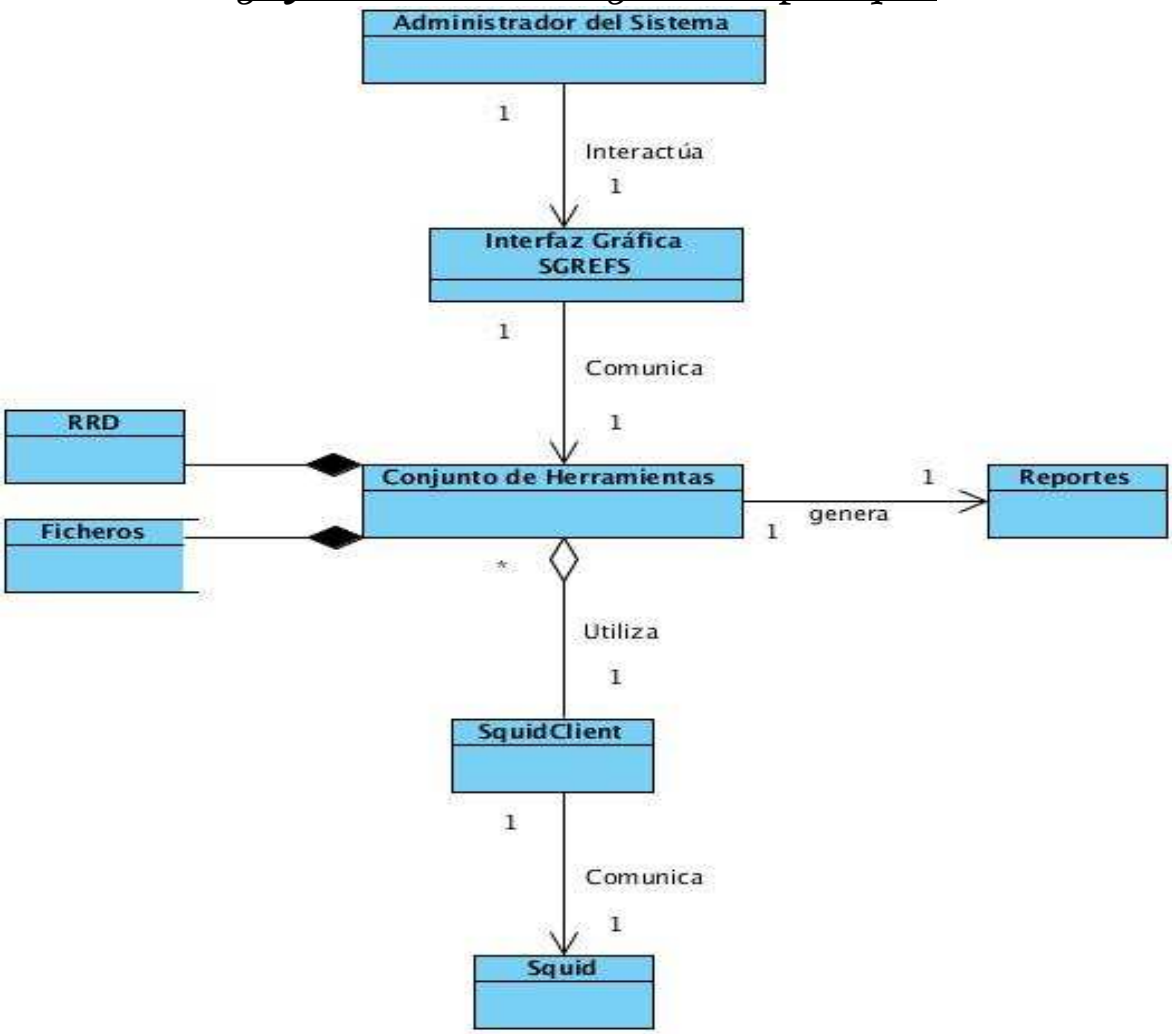


Ilustración 1: Modelo del dominio.

El administrador del sistema desea obtener los reportes de funcionamiento o del tráfico en la red en un momento dado, para lo cual interactúa con la interfaz del SGREFS. Esta herramienta, en dependencia de la petición, ejecuta varias acciones con el fin de mostrarle al actor la petición solicitada. Los datos que se recogen son obtenidos a través de la herramienta SquidClient, cuyo objetivo es solicitar a Squid reportes de su funcionamiento y del tráfico de la red.

A continuación se detallan los elementos que forman el Modelo del Dominio.

- **Administrador del Sistema:** Persona responsable y capacitada para acceder y utilizar el sistema.

- **SGREFS (Sistema Generador de Reportes Estadísticos del Funcionamiento de Squid)**: Herramienta desarrollada en Symfony que tiene como propósito fundamental la obtención de reportes acerca del funcionamiento de Squid y del tráfico de usuarios en Internet. Para ello se encarga de la comunicación con Squid y de manipular las BD RRD.
- **Interfaz Gráfica del SGREFS:** Constituye la vía fundamental de intercambio de los administradores con el sistema.
- **SquidClient:** Herramienta que se comunica mediante el protocolo SNMP con el Proxy Squid, recolectando los datos necesarios que ilustran el funcionamiento de dicho servidor.
- **Squid:** Programa que implementa un servidor Proxy y un dominio para caché de páginas web.
- **RRD:** Sistema para almacenar y presentar datos en series temporales.
- **Ficheros:** Los resultados de algunas peticiones de estadísticas a Squid se guardan en un fichero para luego manipularlos y cargarlos en tablas u otros elementos visuales.
- **Reportes:** Resultado de la petición del usuario. Se obtienen en forma de gráficos en formato .PNG y tablas.

2.4 Requerimientos del sistema

Un requerimiento puede definirse como un atributo necesario dentro de un sistema. Puede representar una capacidad, una característica o un factor de calidad del sistema de tal manera que le sea útil a los clientes o a los usuarios finales [34].

2.4.1 Requisitos funcionales (RF)

Los RF son capacidades o condiciones que el sistema debe cumplir. Especifica lo que el sistema debe ser capaz de realizar. Para la selección de los requisitos del sistema propuesto se realizaron las siguientes acciones:

Mediante el estudio de sistemas similares se identificaron los reportes más utilizados por dichos generadores de reporte. A partir de una lista de los requisitos más utilizados por los diferentes sistemas se procede a realizar pruebas con tales requerimientos, seleccionando los de mayor impacto e importancia para la administración de Squid, los cuales se incluyen en los RF de la propuesta de solución.

Para cumplir los objetivos planteados el sistema propuesto debe poseer los RF que se muestran a continuación:

RF 1	Visualizar reportes de funcionamiento del sistema uso de memoria por el Sistema.
RF 2	Visualizar reportes de funcionamiento del sistema uso de CPU.
RF 3	Visualizar reportes de funcionamiento del sistema número de clientes conectados.
RF 4	Visualizar reportes de funcionamiento del sistema cantidad de objetos en la cache.
RF 5	Visualizar reportes de funcionamiento del sistema conexiones concurrentes.
RF 6	Visualizar reportes de funcionamiento del sistema utilización de memoria por Squid.
RF 7	Visualizar reportes de funcionamiento del sistema solicitudes HTTP recibidas.
RF 8	Filtrar reportes por intervalos de tiempo.
RF 9	Visualizar reportes de cache servidores padres configurados.
RF 10	Visualizar reportes de cache ACL externas.
RF 11	Visualizar tráfico en la red en tiempo real.
RF 12	Visualizar reportes de cache delay polls.
RF 13	Crear nuevo usuario.
RF 14	Editar usuario.
RF 15	Eliminar usuario.
RF 16	Buscar un usuario determinado.
RF 17	Listar usuarios con rol ROLE_ADMIN.
RF 18	Listar usuarios con rol ROLE_USER.

RF 19	Cambiar contraseña.
RF 20	Actualizar BD RRD con las consultas realizadas a Squid.
RF 21	Modificar conexión.
RF 22	Autenticar Usuario.

Tabla 1: Requisitos funcionales

2.4.2 Requerimientos no Funcionales (RNF)

Los RNF son propiedades o cualidades que el producto debe tener. Se debe pensar en estas propiedades como las características que hacen al producto atractivo, confiable, rápido y usable. Para el sistema propuesto se definen los siguientes RNF:

Requerimientos de *software*

Para que el sistema funcione correctamente deben cumplirse las siguientes características:

- Plataforma de sistema operativo: GNU/Linux, preferentemente Debian GNU/Linux 5.0 o superior.
- Paquetes: Squid3, squidclient, apache2, php5, rrdtool y las respectivas dependencias de todos ellos.
- Usuario con privilegios de administración del sistema operativo.

Requerimientos de *Hardware*

Para el funcionamiento óptimo del SGREFS se debe contar con una computadora que posea:

- Al menos 40 GB de capacidad en el disco duro.
- Al menos 256 MB de RAM.

Se recomienda utilizar:

- 160 GB de capacidad en el disco duro.
- 2GB de RAM.

Restricciones del diseño

El sistema deberá hacer uso de los lenguajes PHP (versión 5.0 o superior) y Java Script. Como *framework*

de desarrollo se usará Symfony (en su versión 2.1) que propone una arquitectura modular en tres capas: el modelo, la vista y el controlador. Como herramienta de desarrollo se usará NetBeans (7.3) y como sistema gestor de base de datos se utilizará PostgreSQL y RRD. Otra componente importante es SquidClient, que constituye el elemento fundamental del proceso de comunicación del sistema con Squid para la generación de reportes.

Apariencia o interfaz externa:

Interfaz sencillo amigable con facilidades para su uso por usuarios de cualquier nivel, reduciendo el tiempo de entrenamiento a los mismos. Funcionalidades visibles en todo momento que faciliten la navegación. Elementos organizativos para las funcionalidades (como íconos).

Confiabilidad: La información de los usuarios del sistema no podrá ser modificada por ningún usuario no autorizado, protegiendo así la integridad de los datos. Además los reportes obtenidos serán reales en su totalidad.

Disponibilidad: El sistema está diseñado para un funcionamiento constante, permitiendo el acceso a los servicios que brinda la aplicación en cualquier momento dado. Lo anterior depende únicamente del correcto funcionamiento y configuración de Squid.

Usabilidad: El sistema debe ser fácil de utilizar por todo tipo de usuarios, siempre que posea un nivel básico en conocimientos de Informática; además, las opciones principales deben estar visibles y accesibles para que constituya una navegación fácil de entender y de manejar. El sistema está pensado fundamentalmente para administradores de Squid, por lo que se supone tengan conocimientos básicos acerca de ese tipo de proxy. El administrador podrá permanecer en el sistema el tiempo que estime conveniente.

Portabilidad: El SGREFS puede ser utilizado por otros proyectos e instituciones que lo requieran y utilicen Squid.

Soporte: Una vez terminado el producto se instalará por el personal capacitado, incluyendo las pruebas y el mantenimiento necesario para verificar el buen funcionamiento del sistema.

2.5 Definición de los actores

En el sistema existen tres tipos de actores:

Actor	Descripción
CRON	Encargado de ejecutar los *scripts* utilizados para guardar en la BD RRD el resultado de las consultas realizadas a Squid.
Administrador Común (rol ROLE_USER)	Podrá visualizar los reportes generados por el sistema y realizar las transacciones que éstos contienen.
Administrador del Sistema (rol ROLE_ADMIN)	Además de poder realizar las acciones de los administradores comunes, podrá gestionar los usuarios que acceden al sistema (adicionar, eliminar, editar y listar) y modificar las conexiones con el Squid.

Tabla 2: Definición de los actores.

2.6 Definición de los casos de uso

Un diagrama de caso de uso es una representación gráfica de parte o el total de los actores y casos de usos del sistema, incluyendo sus interacciones. Los diagramas de casos de uso sirven para especificar la funcionalidad y el comportamiento de un sistema mediante su interacción con los usuarios u otros sistemas. Un diagrama de casos de uso consta fundamentalmente de los siguientes elementos:

Actor: Se le llama actor a toda entidad externa al sistema que guarda una relación con éste y que le demanda una funcionalidad. Es un rol que un usuario juega con respecto al sistema. Necesariamente no representa a una persona en particular, sino más bien la labor que realiza frente al sistema. Incluye usuarios humanos y otros sistemas computarizados.

Casos de Uso: Los casos de uso son fragmentos de funcionalidad que el sistema ofrece para aportar un resultado de valor para sus actores. De manera más precisa, un caso de uso especifica una secuencia de transacciones que el sistema puede desarrollar en respuesta a un evento que inicia un actor sobre el propio sistema.

Relaciones: Una relación es una conexión entre los elementos del modelo. En un diagrama de casos de uso, además de las relaciones entre casos de uso y actores, llamadas asociaciones y las relaciones entre casos de uso de dependencias (<<include>> y <<extends>>), pueden existir relaciones de herencia ya sea entre casos de uso o entre actores.

2.6.1 Diagramas de Casos de Uso del Sistema.

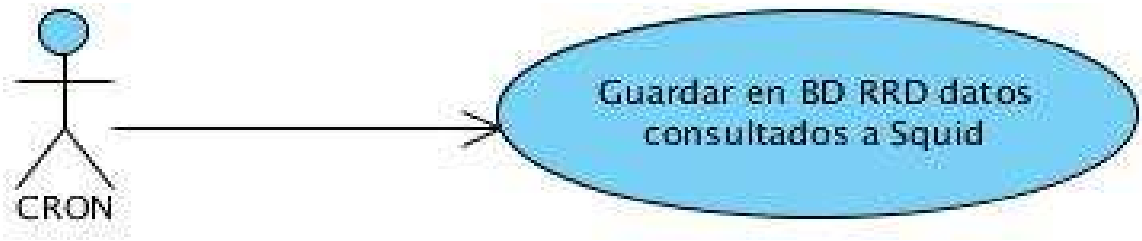


Ilustración 2: Actualizar base de datos con reportes de funcionamiento de Squid.

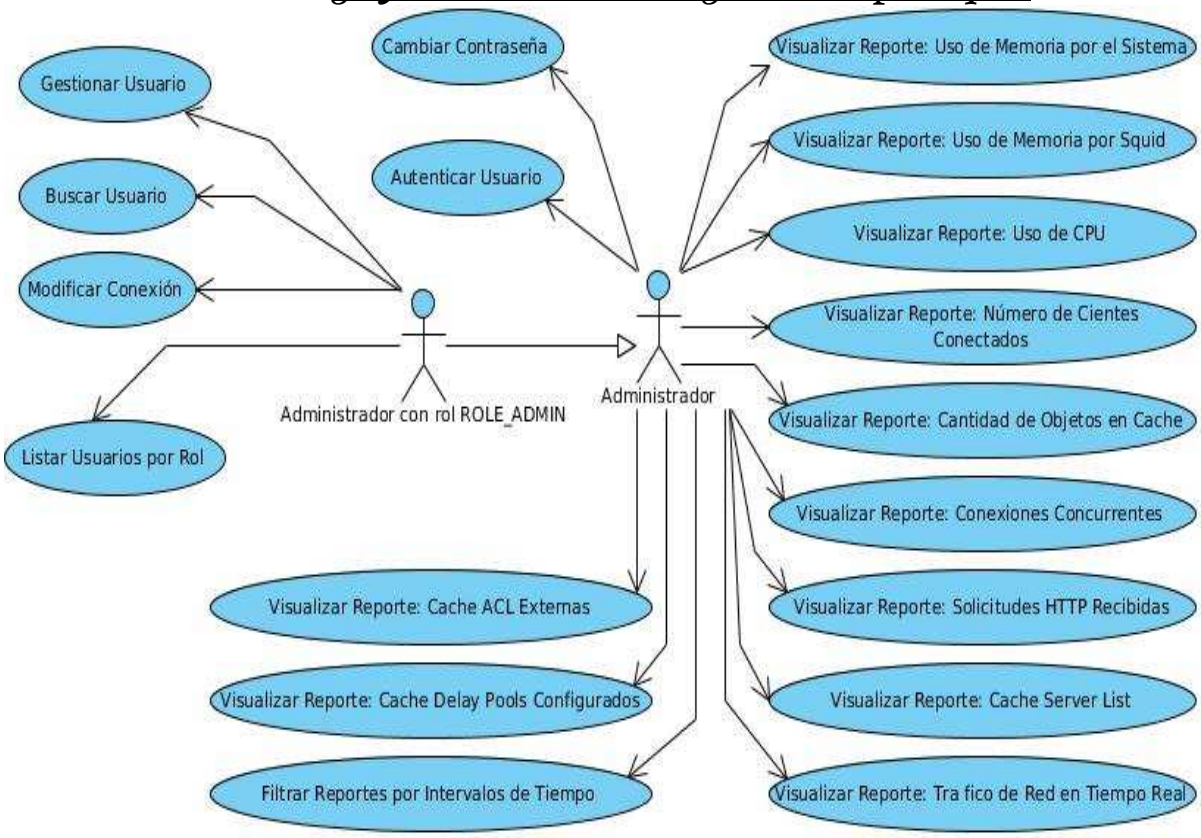


Ilustración 3: DCU de la Interfaz Web.

Especificación de los principales casos de uso.

CU-Guardar en Base de Datos los reportes consultados a Squid**.**

Objetivo	Guardar en Base de Datos los reportes consultados a Squid.	
Actores	CRON	
Resumen	El caso de uso se inicia cuando el actor recupera los datos estadísticos de Squid y los introduce en la BD RRD.	
Complejidad	Alta	
Prioridad	Alta	
Precondiciones	La base de datos debe estar creada.	
Postcondiciones	La base de datos debe ser actualizada.	
Flujo de eventos		
Flujo básico		
	Actor	**Sistema**
	1. El actor ejecuta el actualizador de la base de	1.1 El actualizador, mediante consultas, solicita los

	datos.	datos deseados y los almacena en la BD RRD, terminando así el CU.

Tabla 3: Descripción del CU Actualizar BD.

CU-Visualizar reportes.

Objetivo		Visualizar reportes de funcionamiento y de tráfico en la red.
Actores		Usuario
Resumen		El caso de uso se inicia cuando el actor accede al sistema para visualizar los reportes de funcionamiento de Squid.
Complejidad		Alta
Prioridad		Alta
Precondiciones		El usuario tiene que estar autenticado.
Postcondiciones		Debe mostrar el reporte solicitado.
Flujo de eventos		
Flujo básico Embeber metadatos		
	Actor	**Sistema**
	El actor selecciona el reporte que desea visualizar. a) Si selecciona un reporte de la sección Gráficos con valores Históricos véase la sección “Gráficos con valores Históricos”. b) Si selecciona un reporte de la sección Tablas con peticiones en tiempo real véase la sección **“**Tablas con peticiones en tiempo real**”**.	
Sección 1: “Gráficos con valores Históricos”		
	Actor	**Sistema**
		1.1 El sistema ejecuta un *script* encargado de generar el reporte solicitado (el reporte se muestra en un espacio de tiempo predeterminado, que es para el último día). 1.2 El sistema muestra el reporte en forma de gráfico en formato .PNG, terminando así el CU.
Sección 2: “Tablas con peticiones en tiempo real”		
	Actor	**Sistema**
1.		1.1 El Sistema realiza una petición mediante cachemrg a Squid, utilizando al programa

		SquidClient. 1.2 El Sistema muestra el reporte en forma de tabla y con algún otro elemento visual.

Tabla 4: Reportes de Funcionamiento de Squid.

CU. Filtrar reportes por tiempo.

Objetivo	Filtrar reportes por tiempo.	
Actores	Usuario	
Resumen	El caso de uso se inicia cuando el actor desea obtener reportes de funcionamiento del sistema en otro espacio de tiempo que no es el predefinido (el último día).	
Complejidad	Alta	
Prioridad	Alta	
Precondiciones	El usuario tiene que estar autenticado.	
Postcondiciones	Debe filtrar el reporte sin problema alguno.	
Flujo de eventos		
Flujo básico Añadir aspectos		
	Actor	**Sistema**
	1. El actor selecciona, del reporte en cuestión, el tiempo que desea consultar (última hora, último día, última semana, último mes).	
		1.1 El sistema ejecuta un *script* encargado de graficar el reporte solicitado en el intervalo de tiempo deseado. 1.2 El sistema muestra el reporte en forma de gráfico en formato .PNG, terminando así el CU.

Tabla 5: Filtrar reportes por intervalos de tiempo.

2.7 Modelo de Análisis

El Modelo de Análisis se utiliza para obtener una visión del sistema sobre los RF, expresados en lenguaje técnico; o sea, en el lenguaje del programador. Tiene como ventajas principales que modera y facilita la transición al diseño, además de servir para tener una visión general de la propuesta del sistema.

A continuación se muestran cómo se estereotipan las clases del Análisis:

Nombre	Características	Representación

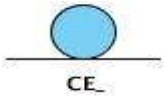
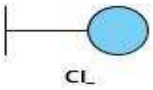
Clase Entidad	Modela la información del sistema y el comportamiento asociado a una información que por lo regular es persistente.	
Clase Controladora	Esta clase coordina el trabajo de las otras clases; encapsula el comportamiento de un CU, y sus funciones son complejas.	
Clase Interfaz	Modela la interacción entre el actor y el sistema, como las ventanas y formularios.	

Tabla 6: Estereotipos del Análisis.

2.7.1 Diagrama de clases del análisis

Interfaz web:

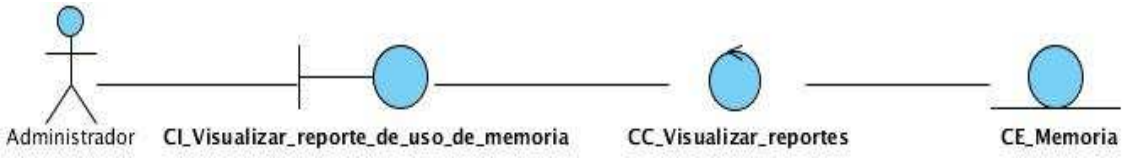


Ilustración 4: Diagrama de clases del Análisis CU Visualizar Uso de Memoria.

Los restantes diagramas se encuentran en los anexos (Anexo I).

2.8 Modelo de diseño

El diseño tiene como principales objetivos comprender detalladamente los RF y los RNF, sistemas operativos, tecnologías de distribución, restricciones relacionadas con los lenguajes de programación, tecnologías de interfaz de usuario, entre otros elementos. Además, crea un punto de partida para las actividades de implementación que siguen. Es el punto de mayor importancia al final de la fase de elaboración y el comienzo de las iteraciones de construcción. Permite facilitar una arquitectura estable y sólida, crear un plano muy cercano del modelo de implementación. Es necesario mantener el modelo de diseño a través de todo el ciclo de vida del *software*.

Propósitos del diseño:

- Adquirir una comprensión de los aspectos relacionados con los RNF y restricciones relacionadas con los lenguajes de programación, tecnologías de distribución y concurrencia, sistemas operativos, componentes reutilizables y tecnologías de interfaz de usuario.
- Crear una entrada apropiada y un punto de partida para actividades de implementación, capturando los requisitos o subsistemas individuales, interfaces y clases.
- Descomponer los trabajos de implementación en partes más manejables que puedan ser instrumentadas por diferentes equipos de desarrollo.
- Capturar las interfaces entre los subsistemas, que es muy útil cuando utilizamos interfaces como elementos de sincronización entre diferentes equipos de desarrollo [35].

A continuación se muestra cómo se estereotipan las clases del Diseño:

Nombre	Características	Representación
Server Page	Representa la página web que tiene código que se ejecuta en el servidor. Dicho código interactúa con recursos en el servidor.	
Client Page	Mezcla los datos, la presentación y la lógica. Son páginas interpretadas por el navegador. Sus atributos son las variables declaradas dentro del *script* que son accesibles para cualquier función dentro de la página.	
Formulario	Colecciona los elementos de entrada que son parte de una página cliente. Se relaciona directamente con la etiqueta de igual nombre del HTML.	

Tabla 7: Estereotipos del Diseño.

2.8.1 Diagrama de Clases del Diseño

El diagrama de clases del diseño se encarga de representar los métodos y atributos de cada una de las clases del sistema, para mostrar de forma simple la colaboración y las tareas de cada una de ellas en relación al sistema que conforman. Partiendo de la descripción detallada de los casos de uso del sistema

se modelan los diagramas de clases del diseño:

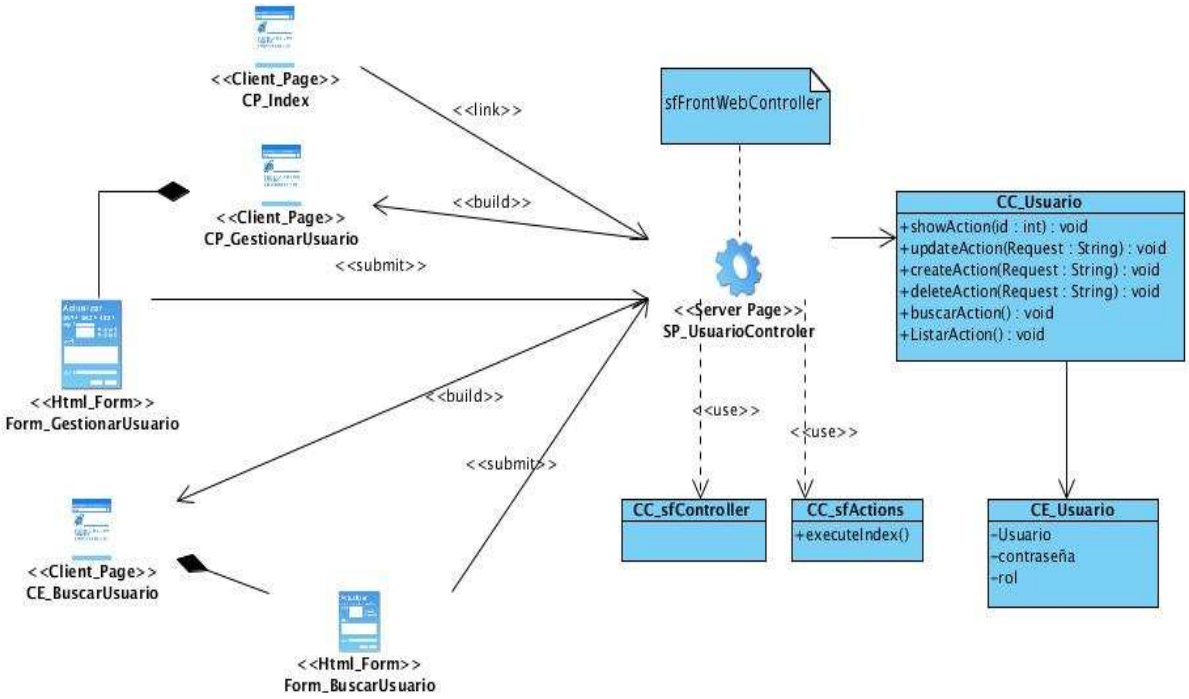


Ilustración 5: CU-Gestionar Usuario.

El diagrama de la figura anterior representa el caso de uso Gestionar Usuario donde se realizan las acciones básicas de un gestionar común, se encuentran además las funcionalidades de listar los usuarios por rol y buscar los datos de un usuario específico.

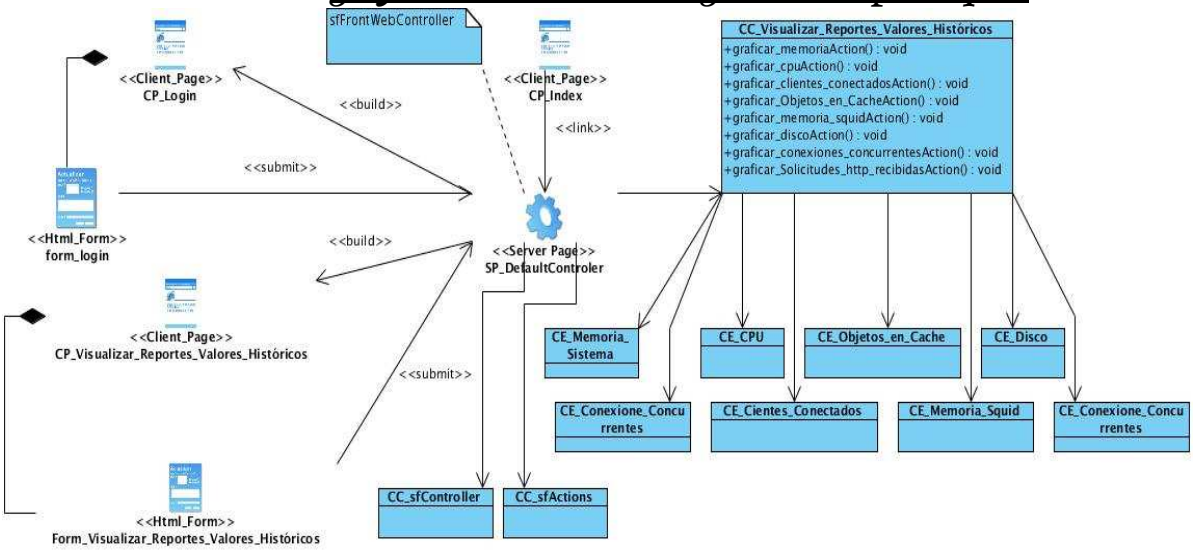


Ilustración 6: CU-Mostrar Reportes Gráficos con Valores Históricos.

El anterior diagrama representa el caso de uso Mostrar Reportes Gráficos con Valores Históricos, donde se recogen los diferentes casos de uso de su tipo por la similitud que presentan estos. En él, al elegir una de las opciones que se encuentran en el apartado “Reportes con Valores Históricos”, el sistema ejecuta las acciones pertinentes para mostrar al usuario el reporte solicitado mediante gráficos.

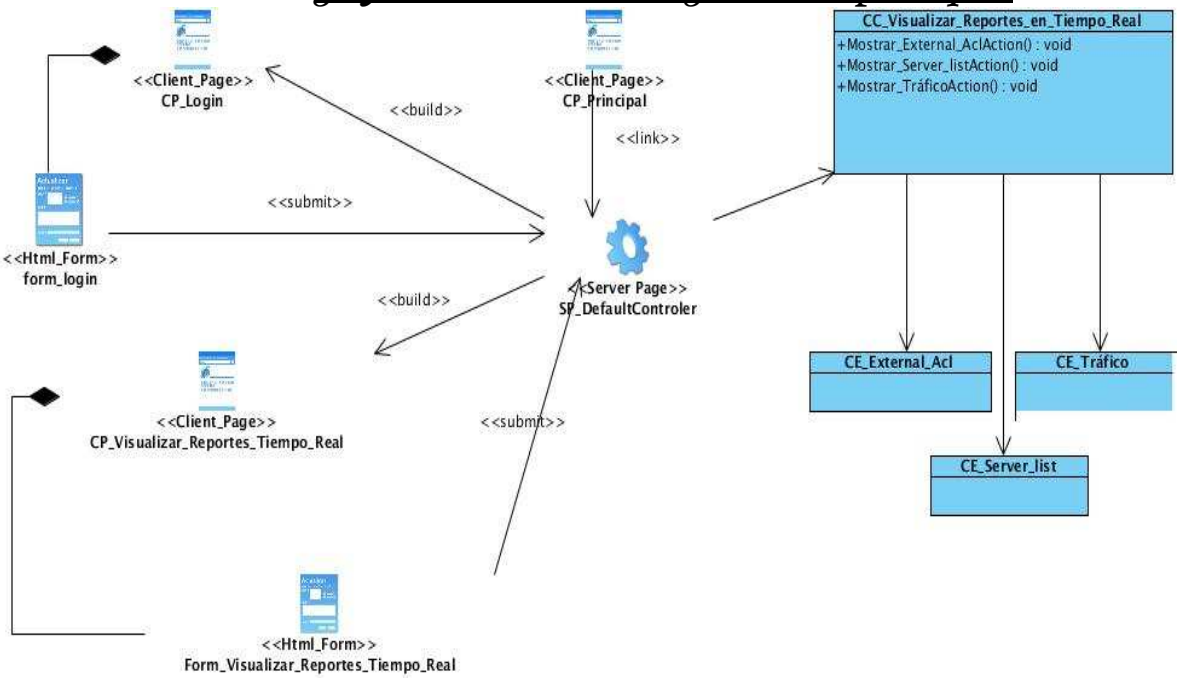


Ilustración 7: CU-Mostrar Reportes en Tiempo Real.

El anterior diagrama representa el caso de uso Mostrar Reportes Gráficos en Tiempo Real, donde se recogen los diferentes casos de uso de su tipo por la similitud que presentan estos. En él, al elegir una de las opciones que se encuentran en el apartado “Reportes en Tiempo Real”, el sistema ejecuta las acciones pertinentes para mostrar al usuario el reporte solicitado mediante tablas y otros elementos visuales.

2.8.2 Fundamentación del uso de patrones

Los patrones son una descripción de un problema y la esencia de su solución, de forma que esta pueda ser reutilizada en diferentes situaciones.

Según Grady Booch "Una arquitectura orientada a objetos bien estructurada está llena de patrones. La calidad de un sistema orientado a objetos se mide por la atención que los diseñadores han prestado a las colaboraciones entre sus objetos. Los patrones conducen a arquitecturas más pequeñas, más simples y más comprensibles".

2.8.2.1 Patrón arquitectónico

Los patrones arquitectónicos son los encargados de definir la estructura de un sistema. Estos se

componen de subsistemas con sus responsabilidades; además poseen una serie de directivas para organizar los componentes con el objetivo de facilitar la tarea del diseño [36].

MVC (Modelo Vista Controlador)

La aplicación será desarrollada con el *framework* Symfony, el cual implementa el patrón arquitectónico MVC. A continuación se muestran algunas características de dicho patrón arquitectónico.

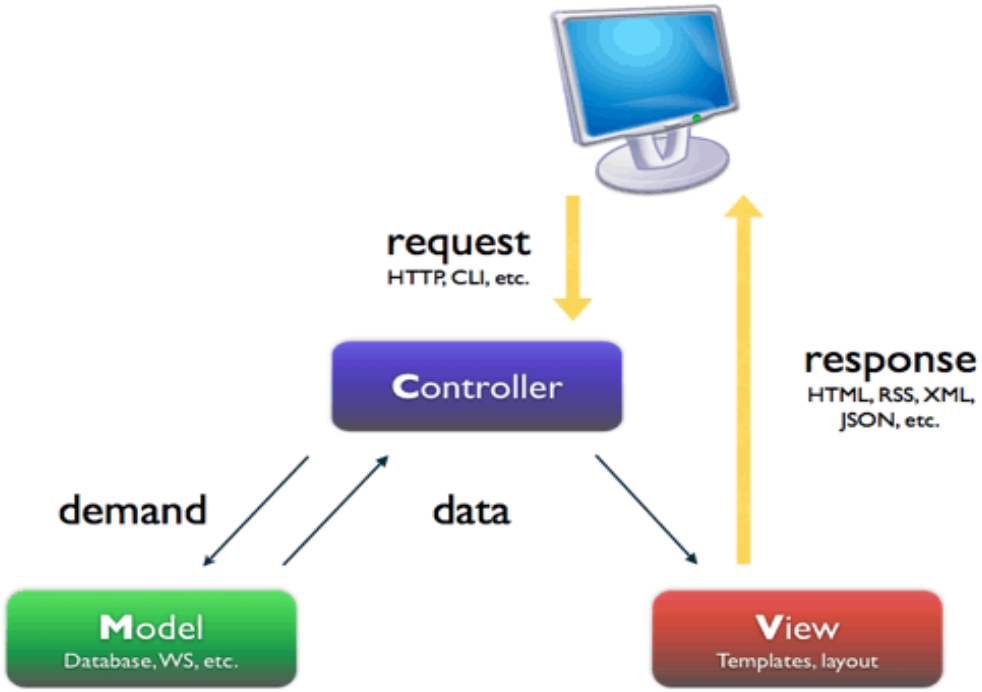


Ilustración 8: Patrón Arquitectónico.

El patrón arquitectónico MVC permite realizar la programación multicapa. Separa el sistema en tres componentes distintos: los datos de una aplicación, la interfaz del usuario, y la lógica de control. Aparece usualmente en aplicaciones web, donde:

- **Modelo:** Representación específica de la información con la cual el sistema opera. La lógica de datos asegura la integridad de estos y permite derivar nuevos datos.
- **Vista:** Presenta el modelo en un formato adecuado para interactuar, usualmente la interfaz de usuario. Maneja la presentación visual de los datos representados por el Modelo.

- **Controlador:** Responde a eventos, usualmente acciones del usuario, e invoca cambios en el modelo y en la vista.

La arquitectura MVC separa la lógica de negocio (el modelo) y la presentación (la vista), lo que permite un mantenimiento más sencillo de las aplicaciones. El controlador es el encargado de aislar al modelo y a la vista de los detalles del protocolo usado para las peticiones (HTTP, consola de comandos, *e-mail*, etc.). El modelo se encarga de la abstracción de la lógica referida a los datos, lo que permite que la vista y las acciones sean independientes de, por ejemplo, el tipo de gestor de bases de datos que la aplicación utiliza [37].

Entre sus características se encuentran:

- Existe una clara separación entre los componentes de un programa, lo cual permite implementarlos por separado.
- Existe un API bien definido. Cualquiera que use el API podrá, sin aparente dificultad, reemplazar el Modelo, la Vista o el Controlador.
- La conexión entre el Modelo y sus Vistas es dinámica y se produce en tiempo de ejecución.

Entre sus ventajas se encuentran:

- Posee soporte para múltiples vistas, debido a que la Vista se separa del Modelo y no hay ninguna dependencia directa entre ambos.
- Proporciona un mecanismo de configuración a componentes complejos mucho más tratable que el puramente basado en eventos.
- Permite un mayor soporte a los cambios, debido a que los requisitos de interfaz tienden a cambiar más rápidamente que las reglas de negocio. Puesto que el modelo no depende de las vistas, la adición de nuevos tipos de vista al sistema generalmente no afectan al modelo; por tanto, el ámbito del cambio se limita a la vista [38].

2.8.2.2 Patrones de diseño

Los patrones de diseño son abstracciones de alto nivel que documentan soluciones de diseño exitosas. Son fundamentales para reutilizar el diseño en el desarrollo orientado a objetos. Proveen un esquema para refinar los subsistemas o componentes de un sistema de *software*, o las relaciones entre ellos. Cada patrón se adapta a un cierto tipo de problema.

Los patrones de diseño pretenden:

- Proporcionar catálogos de elementos reusables en el diseño de sistemas de *software*.
- Evitar la reiteración en la búsqueda de soluciones a problemas ya conocidos y solucionados anteriormente.
- Formalizar un vocabulario común entre diseñadores.
- Facilitar el aprendizaje de las nuevas generaciones de diseñadores condensando conocimiento ya existente.

Para el desarrollo de la aplicación, mediante la utilización del *framework* Symfony, ya se establecen ciertos patrones que éste implementa por defecto. Se hará uso de los patrones GRASP (*General Responsability Asignment Software Patterns*) y GoF (*Gang of Four*).

GRASP

Alta Cohesión: Symfony permite la organización del trabajo en cuanto a la estructura del proyecto y la asignación de responsabilidades con una alta cohesión. Un ejemplo de ello es la clase *Actions*, que contiene varias funcionalidades que están estrechamente relacionadas, siendo la misma la responsable de definir las acciones para las plantillas y colaborar con otras para realizar diferentes operaciones.

Experto: El patrón experto plantea asignar una responsabilidad a la clase que tiene toda la información necesaria para cumplir con dicha responsabilidad. Es un principio usado continuamente en el diseño orientado a objetos. Dicho patrón se evidencia en la capa Modelo, donde existen dos tipos de clases, las de acceso a datos y las de abstracción de datos**.**

Creador: Mediante el patrón creador se identifica el responsable de crear una nueva instancia de alguna clase. En la clase *Actions* se encuentran las acciones definidas para el sistema y se ejecutan en cada una de ellas. En dichas acciones se crean los objetos de las clases que representan las entidades, lo que evidencia que la clase *Actions* es “creadora” de dichas entidades**.**

Bajo Acoplamiento: La clase *Actions* hereda únicamente de *sfActions* para alcanzar un bajo acoplamiento de clases. Las clases que implementan la lógica del negocio y de acceso a datos se encuentran en el modelo, las cuales no tienen asociaciones con las de la vista o el controlador, lo que proporciona pocas dependencias entre las mismas.

Controlador: El patrón controlador asigna la responsabilidad del manejo de los eventos del sistema a una clase. Todas las peticiones web son manejadas por un solo controlador, que es el único punto de entrada de toda la aplicación en un entorno determinado. Cuando el controlador recibe una petición, utiliza el sistema de enrutamiento para asociar el nombre de una acción y el nombre de un módulo con la URL entrada por el usuario. Con el uso de dicho patrón se puede obtener como beneficio el incremento del potencial de los elementos que pueden ser reutilizados [39]. El patrón controlador se evidencia en los "*actions*" del sistema (DefaultControler.php).

GoF

Patrón *Singleton* (Instancia única)**:** Garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia. En el controlador frontal hay una llamada al método getInstance. Donde se encarga de enrutar todas las peticiones que se hagan a la aplicación. El singleton sfRouting precisa otros métodos muy útiles para la gestión manual de las rutas: ClearRoutes (), hasRoutes (), getRoutesByName ().

Patrón *Decorator* (Decorador)**:** Pertenece a la clase abstracta sfView, padre de todas las vistas, que contienen un decorador para permitir agregar funcionalidades dinámicamente. El archivo nombrado frontent.php es el que contiene el *Layout* de la página. Dicho archivo, conocido también como plantilla global, guarda el código HTML que es usual en todas las páginas del sistema, para no tener que repetirlo en cada página. El contenido de la plantilla se integra en el *layout*, o si se mira desde el otro punto de vista, el *layout* decora la plantilla.

Patrón *Registry* (Registro)**:** Muy útil para los desarrolladores en la Programación Orientada a Objetos. Es un medio sencillo y eficiente de compartir datos y objetos en la aplicación sin la necesidad de preocuparse por conservar numerosos parámetros o hacer uso de variables globales. Dicho patrón se aplica en la clase sfConfig, que es la encargada de acumular todas las variables de uso global en el sistema. Symfony aplica además el patrón Front Controller (Controlador frontal), por lo que posee una estructura bien organizada de controladores, que comienza desde el index.php del ambiente y termina en los *Actions*. Cada clase de esta capa tiene su responsabilidad y es única. Hay controladores que se encargan de la seguridad del sistema trabajando con ficheros YML, y otros que se encargan de identificar mediante algunos datos las clases que deben realizar determinadas tareas (Patrón GoF Command, clase sfRouting) y las clases relacionadas con la configuración del sistema (sfConfig, y sfConfigHandler) [40].

2.8.3 Diagramas de secuencia

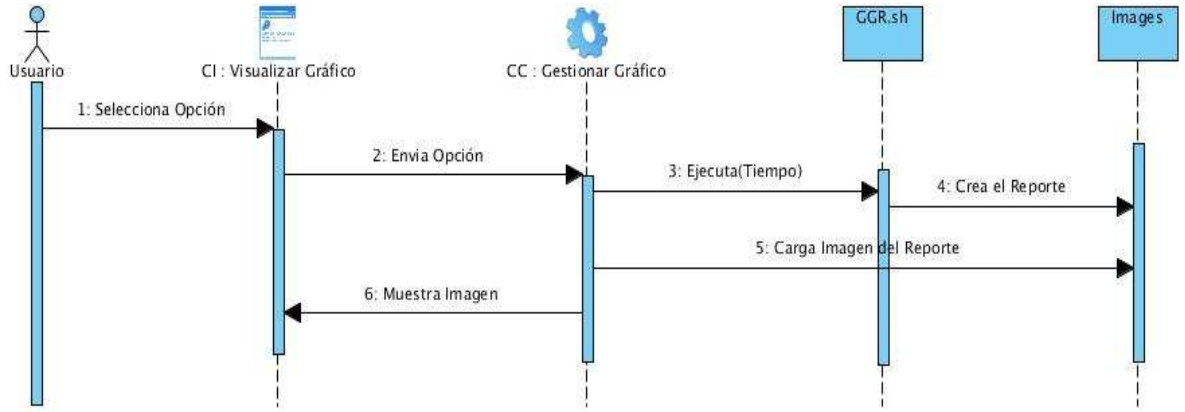


Ilustración 9: Mostrar Reportes Gráficos con Valores Históricos.

Cuando el usuario accede a uno de los reportes que está dentro del apartado “Reportes con Valores Históricos”, el sistema ejecuta un *script* que se encarga de generar un gráfico con los datos existentes en la BD RRD correspondiente. Luego esta imagen es cargada y mostrada al usuario.

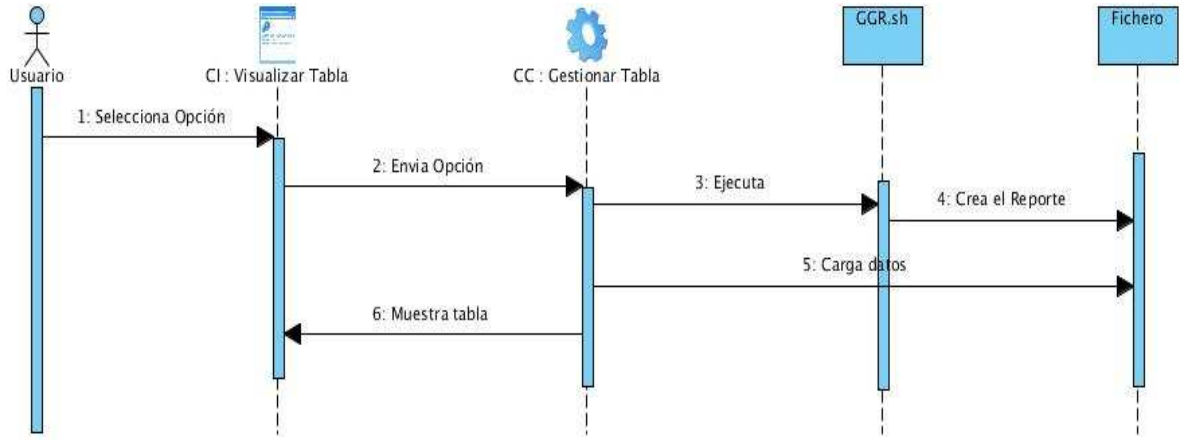


Ilustración 10: CU-Mostrar Gráficos en Tiempo Real.

Cuando el usuario accede a uno de los reportes que está dentro del apartado Reportes en Tiempo Real, el sistema ejecuta un *script* que mediante Squidclient realiza una consulta a Squid y los guarda en un fichero. Luego estos datos son cargados en tablas y mostrados al usuario.

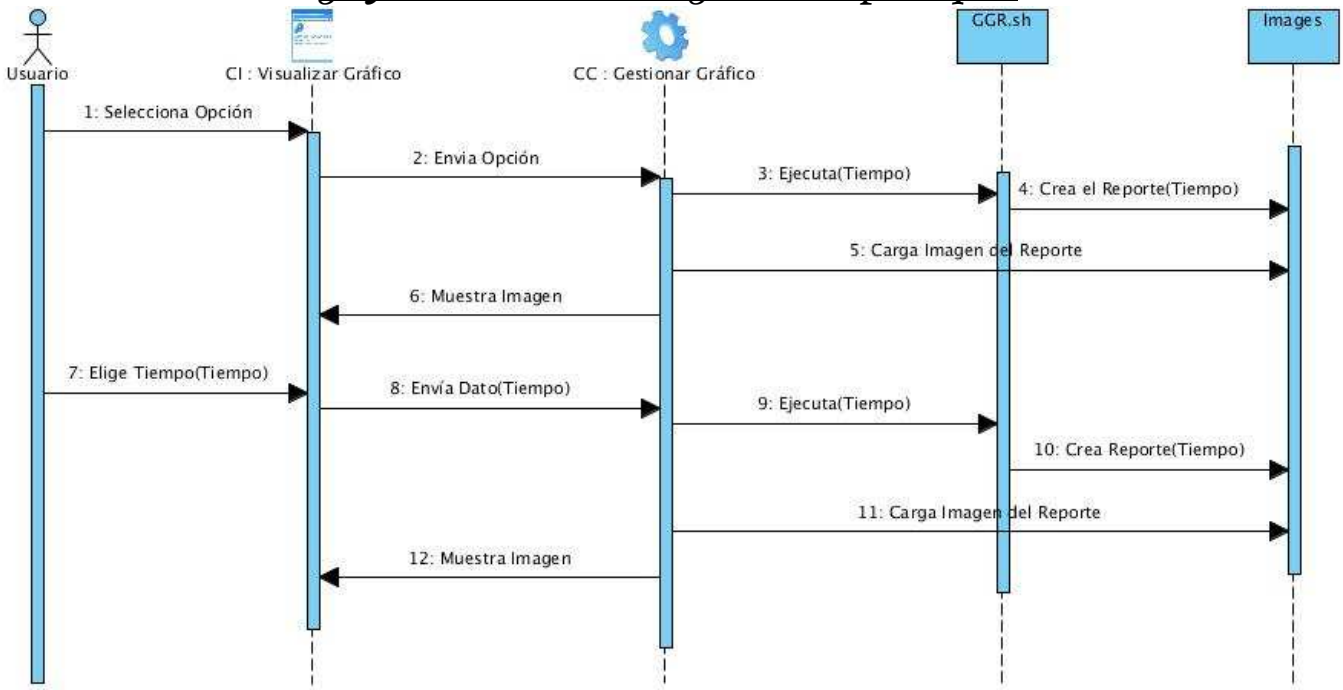


Ilustración 11: CU-Filtrar Gráficos por Tiempo.

El sistema posee la funcionalidad de filtrar los reportes gráficos en diferentes espacios de tiempo (última hora, último día, última semana o último mes). Una vez elegido el tiempo deseado, el sistema ejecuta un *script* que se encarga de generar el gráfico de ese reporte con el tiempo especificado por el usuario.

2.9 Modelo de Despliegue

El diagrama de despliegue define la arquitectura física del sistema por medio de nodos interconectados. Dichos nodos son elementos de *hardware* sobre los cuales pueden ejecutarse los elementos de *software* [41]. El diagrama de despliegue se utiliza para visualizar la distribución de los componentes de *software* en los nodos físicos. A continuación se muestra una descripción de los elementos que componen un diagrama de despliegue.

Nodos: elementos de procesamiento con al menos un procesador, una memoria y posiblemente otros dispositivos.

Dispositivos: nodos estereotipados sin capacidad de procesamiento en el nivel de abstracción que se modela.

Conectores: expresan el tipo de conector o protocolo utilizado entre el resto de los elementos del modelo.

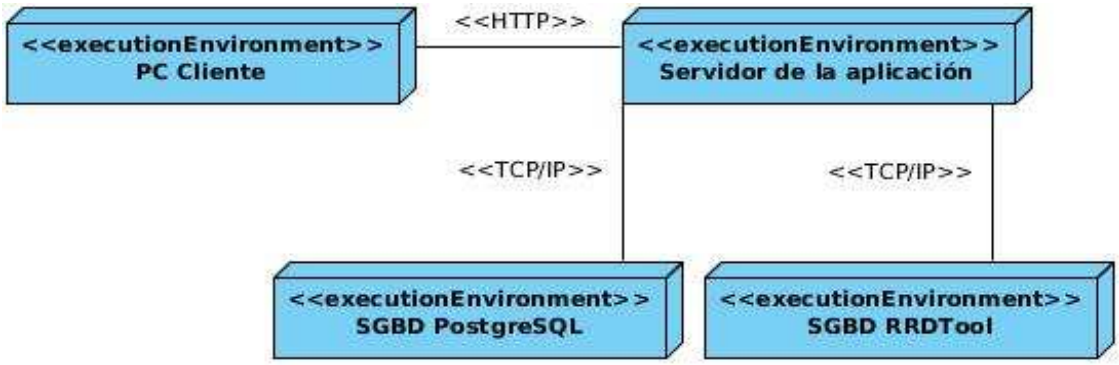


Ilustración 12: Modelo de despliegue.

Como se aprecia en la figura, el usuario (desde una PC cliente y utilizando un navegador) se comunica vía HTTP con el servidor de la aplicación, que a la vez se comunica, mediante la familia de protocolos TCP/IP con las bases de datos epecificadas.

2.10 Conclusiones parciales

- El análisis del flujo actual de los procesos permitió ganar en comprensión sobre el problema existente.
- La solución propuesta permitió establecer una guía para el desarrollo del sistema.
- La definición de los requerimientos del sistema esclareció las características que éste debe tener.
- El diagrama de clases del diseño y el diagrama de secuencia permitió representar la estructura estática y dinámica del sistema, ayudando a esclarecer estas.
- La definición de los patrones de diseño y de arquitectura dotaron al sistema de una arquitectura más sólida.
- El diagrama de despliegue permitió mostrar la distribución de los componentes de *software* en nodos físicos, así como los protocolos de comunicación entre dichos nodos.

3 Capítulo 3

Introducción

El desarrollo del *software* implica una serie de actividades de producción donde son comunes las posibilidades de que aparezcan errores del desarrollador. Los errores pueden ocurrir desde el primer momento del proceso en el que los objetivos pueden estar especificados de forma errónea.

Debido a la imposibilidad humana de trabajar y comunicarse de forma perfecta, el desarrollo del *software* ha de ir acompañado de una actividad que garantice la calidad. Las pruebas son una actividad en la cual un sistema o componente es ejecutado bajo unas condiciones específicas; los resultados son observados y registrados, y una evaluación es hecha respecto a algún aspecto del sistema o componente.

La prueba de *software* es un elemento crítico para la garantía de su calidad y representa una revisión final de las especificaciones del diseño y de la codificación. La técnica de prueba que se aplicó al sistema es la de caja negra, que se refiere a las pruebas que se llevan a cabo sobre la interfaz del *software*. 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 un resultado correcto, así como que se mantiene la integridad de la información externa [42].

3.1 Diagrama de componentes

Los diagramas de componentes describen los elementos físicos del sistema y sus relaciones. Los componentes representan todos los tipos de elementos de *software* que intervienen en la fabricación de aplicaciones informáticas. Pueden ser simples archivos, paquetes o bibliotecas que se cargan dinámicamente. A continuación se muestra el diagrama de componentes del sistema desarrollado:

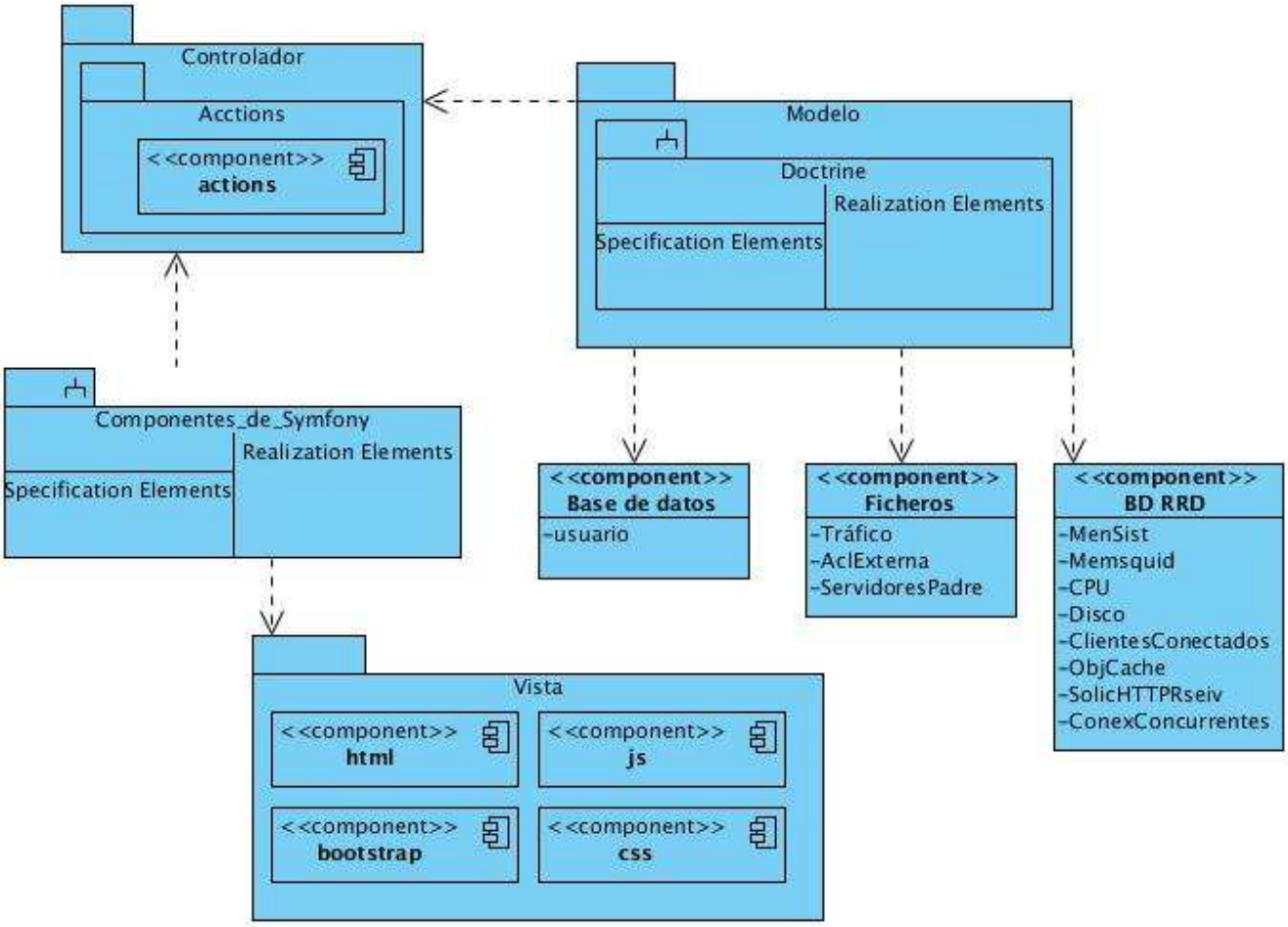


Ilustración 13: Diagrama de componentes.

Para una mayor comprensión sobre los componentes desarrollados, se especifican los ficheros más utilizados para la gestión de los reportes.

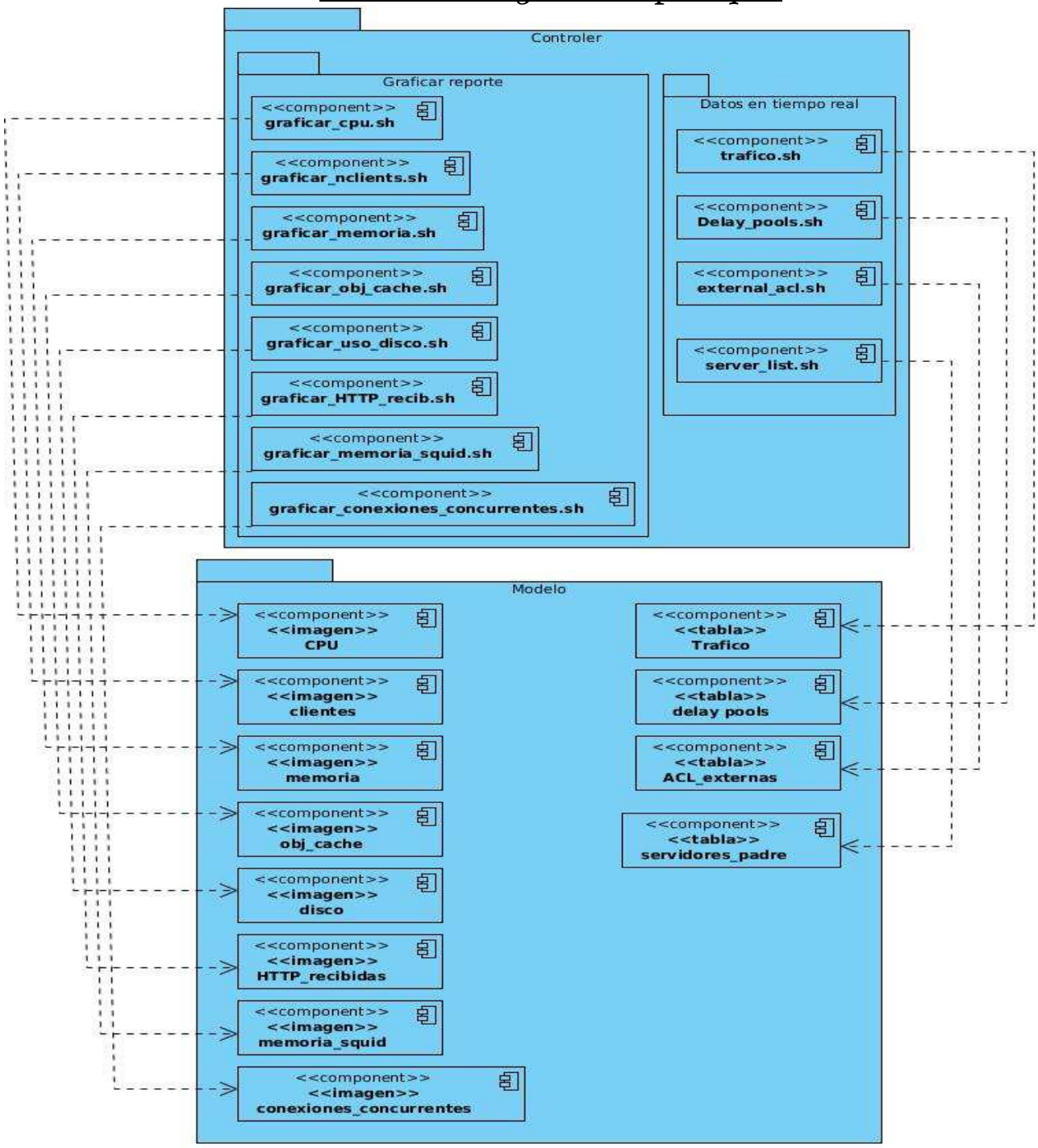


Ilustración 14: Componentes de la gestión de reportes.

3.2 Pantallas principales de la aplicación

La información en el sistema está estructurada de una manera fácil e intuitiva donde existe una barra superior que muestra acceso a un conjunto de acciones como la gestión de usuarios, la opción de cambiar contraseña, acceder a la página de ayuda, entre otras. Existe también un menú lateral a la izquierda el cual proporciona el acceso de los reportes existentes en el sistema. A la derecha de las páginas existe un panel que es donde se muestra la información de los reportes a los usuarios.

Las siguientes imágenes muestran las principales vistas del sistema desarrollado.

Ilustración 15: Vista página principal.

La imagen anterior muestra la vista de la página principal para los administradores con rol ROLE_ADMIN, esta muestra los datos generales del servidor Squid que se está analizando. La diferencia con la de los administradores con rol ROLE_USUARIO es que en esta se pueden visualizar las opciones de gestión de usuarios (parte superior izquierda) y modificar conexión (debajo de los datos de Squid).

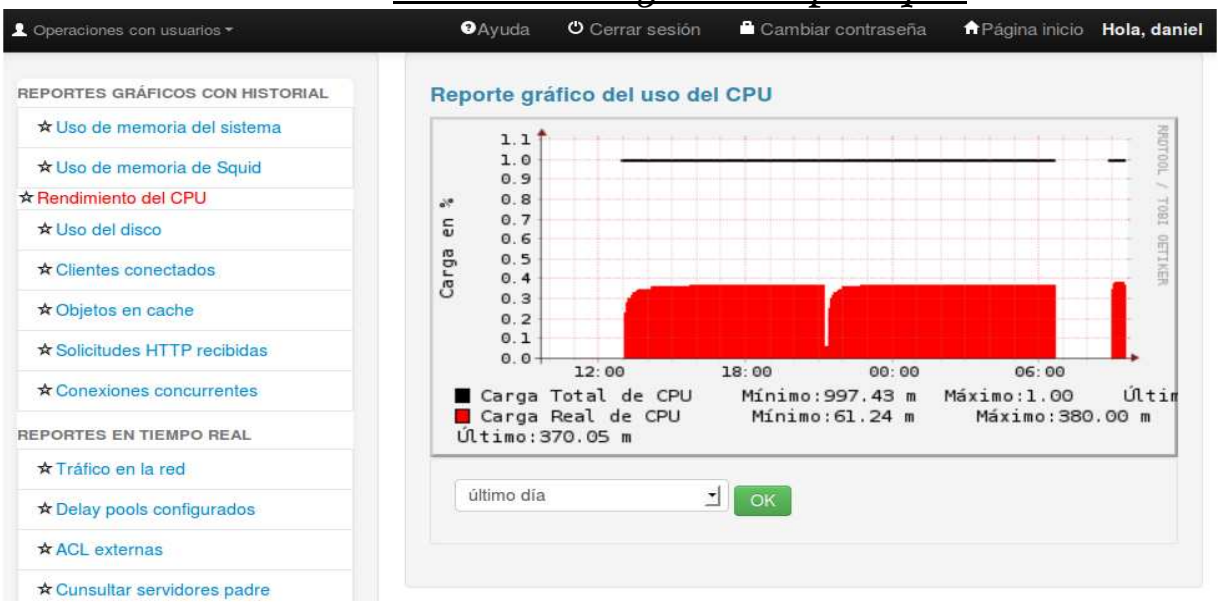


Ilustración 16: Vista del uso de CPU.

La vista de la imagen anterior muestra un reporte en forma de gráfico del consumo de CPU por Squid.

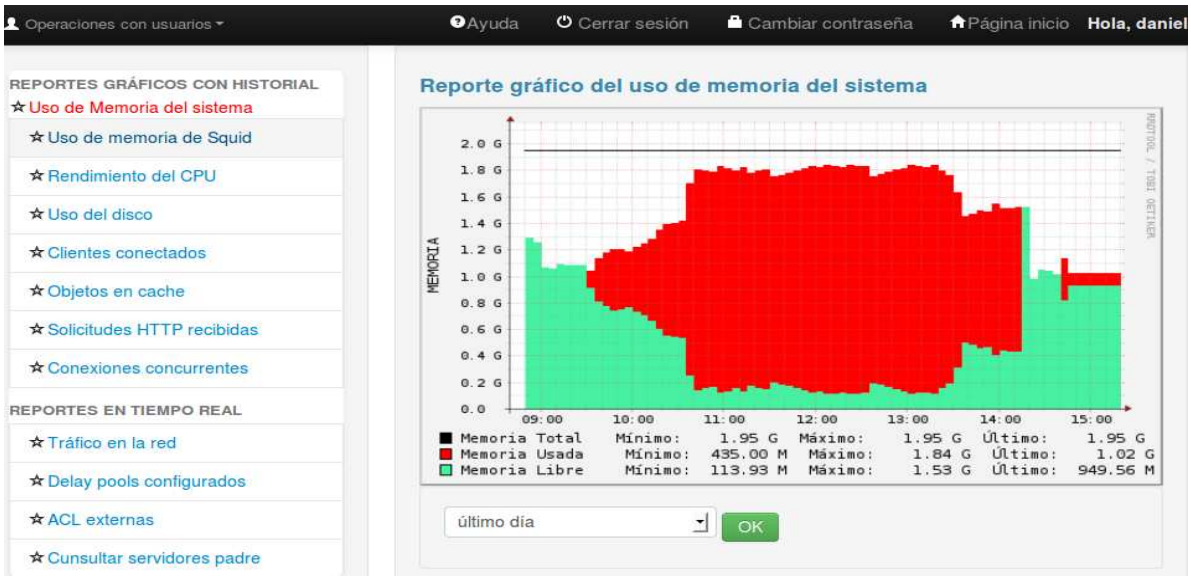


Ilustración 17: Reporte Uso de memoria por el sistema.

La anterior imagen muestra la utilización de memoria por el sistema de manera general.

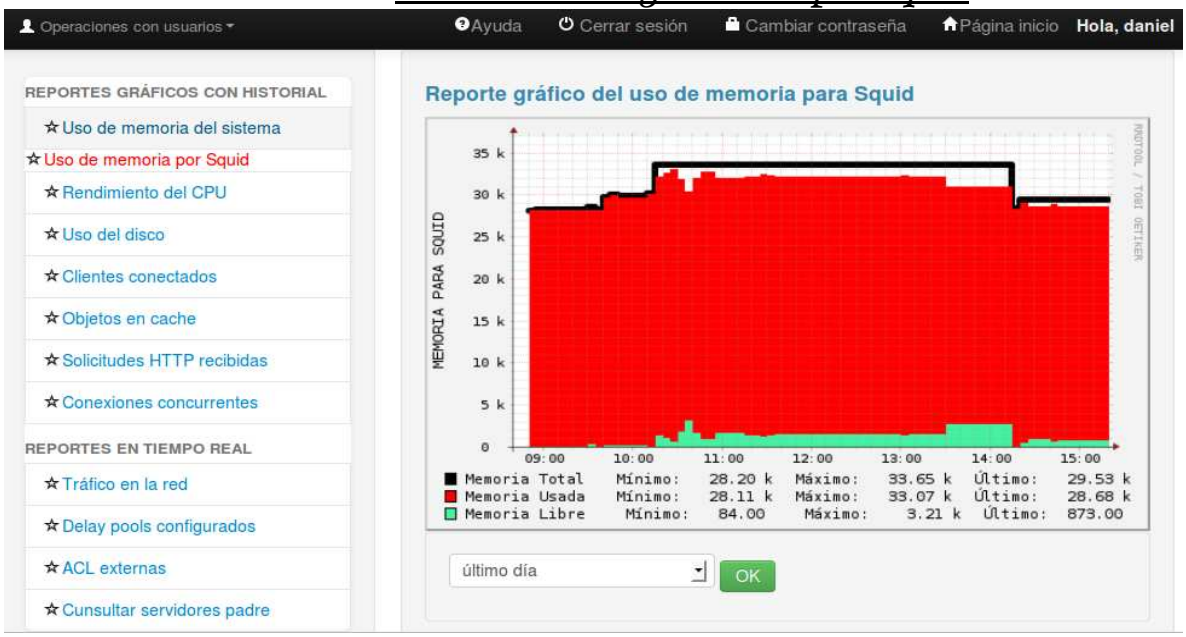


Ilustración 18: Vista del consumo de memoria por Squid.

La imagen anterior muestra el consumo de memoria por Squid.

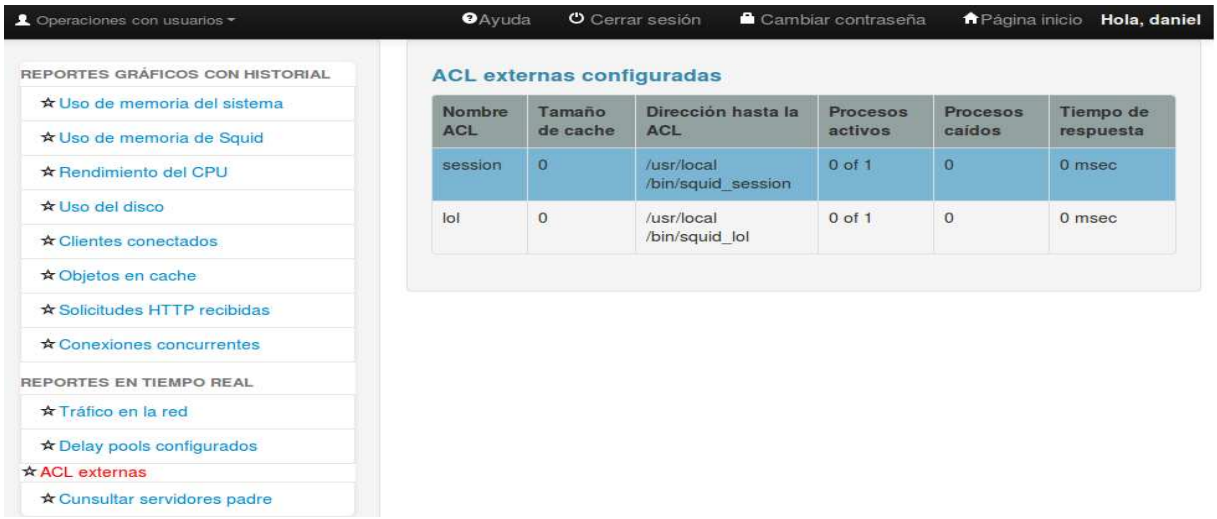


Ilustración 19: Vista de ACL externas configuradas.

La imagen anterior muestra una tabla con las principales características de las ACL externas configuradas en el Squid.

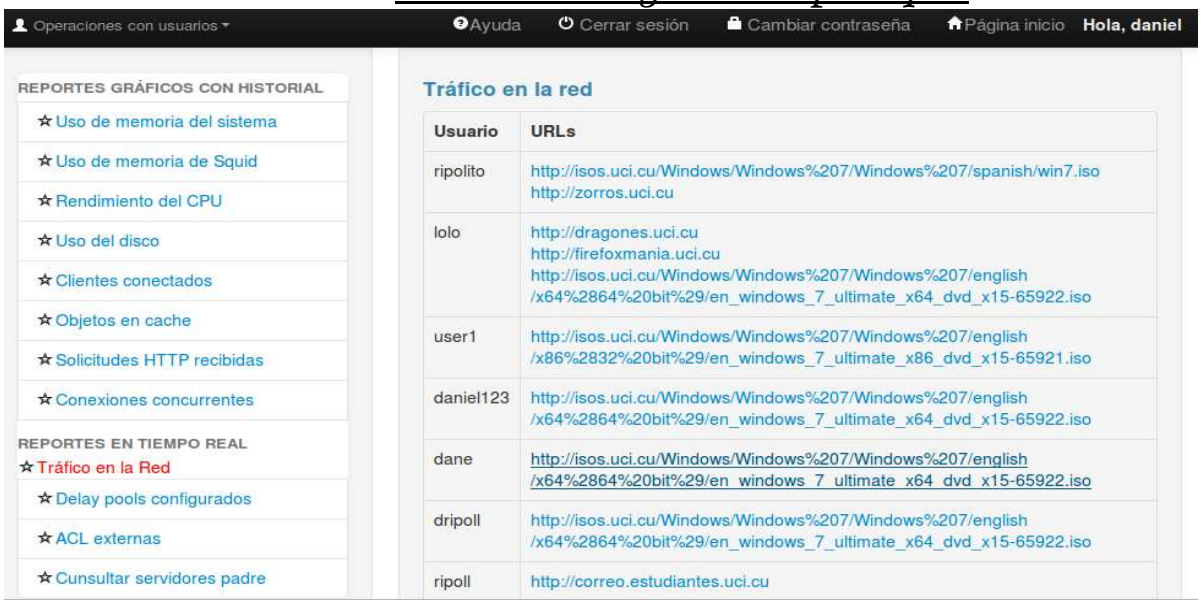


Ilustración 20: Vista del tráfico en la red por parte de los usuarios.

La imagen anterior muestra, en forma de tabla, los usuarios que están navegando en tiempo real así como la dirección de las páginas a las que acceden.

3.3 Técnica de prueba

El *software* debe ser probado con el objetivo de detectar y corregir el máximo de errores posibles antes de su entrega al cliente. Para esto es necesario diseñar los casos de pruebas para encontrar la mayor cantidad posible de errores. Existen dos técnicas de prueba: caja blanca y caja negra. La técnica de caja negra se centra en los RF del *software*, y es por ello que ha sido empleada en el diseño de los casos de prueba. También permiten al ingeniero del *software* obtener un conjunto de condiciones de entrada que prueben completamente todos los RF del sistema.

3.4 Tipos de pruebas aplicadas

Para comprobar la calidad del producto desarrollado se realizaron pruebas funcionales, las cuales consisten en evaluar las funcionalidades del sistema. Estas están centradas en asegurar que el producto satisface los RF establecidos. Para la realización de dichas pruebas se diseñaron casos de prueba basados en casos de uso. Además, fueron aplicadas pruebas de aceptación. Las pruebas de aceptación son realizadas por el usuario final y su objetivo es verificar que el *software* está listo para ser usado [44].

3.4.1 Pruebas de caja negra

Las pruebas de caja negra son las que se realizan en 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*.

A continuación se presentan los casos de pruebas para los casos de uso que necesitaban alguna validación especial, permitiendo comprobar que la aplicación funciona de forma correcta.

V: indica válido
I: indica inválido
NA: indica que no es necesario proporcionar un valor del dato en este caso, ya que es irrelevante.

CU: Cambiar Contraseña.

Escenario	Descripción	Variable 1	Variable 2	Variable 3	Respuesta del sistema	Flujo central
EC 1.1 Incetrando algún valor incorrecto	*El usuario introduce algún dato incorrectamente*	*I contraseña incorrecta*	*NA*	*NA*	*El sistema muestra un mensaje de error: Ha introducido valores no válidos.*	*El usuario accede a la opción Cambiar Contraseña de la barra superior de la página e introduce los valores de cambio de contraseña. En dependencia de los datos introducidos, el sistema devuelve los mensajes de error.*
		V contraseña correcta	*I hjdfhhdfh*	*NA*	*El sistema muestra un mensaje de error: La contraseña debe ser fuerte (ver en ayuda)*	
		V contraseña correcta	*V Daniel123**	*I diferente a la anterior*	*El sistema muestra un mensaje de error: Ha introducido valores no válidos.*	
EC 1.2 Incertando todos los valores correctamente.	*El usuario introduce los datos correctamente.*	*V contraseña correcta*	*V Daniel123**	*V Daniel123**	*El sistema muestra un mensaje comunicando que la acción se ha realizado correctamente.*	*El usuario accede a la opción Cambiar Contraseña de la barra superior de la página e introduce los valores de cambio de contraseña. El sistema devuelve un mensaje comunicándole al usuario que se ha realizado la acción correctamente.*

Tabla 8: Caso de prueba del CU-Cambiar contraseña.

CU: Autenticar Usuario

Escenario	Descripción	Variable 1	Variable 2	Respuesta del sistema	Flujo central
EC 1.1 Deja algún campo vacío.	El usuario introduce sus datos de acceso incorrectamente	I null	N/A	El sistema muestra un mensaje de error: Por favor, rellena este campo.	Para acceder al sistema el usuario debe loguearse y si este deja algún campo vacío el sistema muestra un mensaje de error.
		V usuario existente	I null		
EC 1.2 Incertando el usuario o la contraseña incorrectamente	El usuario introduce uno o los dos datos de acceso incorrectamente	I usuario	N/A	El sistema muestra un mensaje de error: Datos incorrectos.	Para acceder al sistema el usuario debe loguearse y si este introduce valores incorrectos el sistema muestra un mensaje de error.
		V usuario existente	I contraseña incorrecta		
EC 1.3 Incetrando usuario y contraseña correctamente	El usuario intruduce el usuario y contraseña correctamente.	V usuario correcto	V contraseña correcta	El sistema muestra la página principal en dependencia del rol del usuario.	Luego de loguearse correctamente, el usuario puede acceder a los recursos del sistema en dependencia del rol que desempeña.

Tabla 9: Caso de prueba del CU-Autenticar Usuario.

CU: Crear Nuevo Usuario

Escenario	Descripción	Variable 1	Variable 2	Variable 3	Respuesta del sistema	Flujo central
EC 1.1 Deja algún campo vacío.	El usuario deja algún campo vacío.	I null	N/A	N/A	El sistema muestra un mensaje de error: Por favor, rellena este campo.	El usuario accede a la opción Crear Nueo Usuario a la izquierda de la barra superior. Si este deja algún campo vacío el sistema muestra un mensaje de error.
		N/A	I null	N/A		
		N/A	N/A	I null		
EC 1.2 Incertando el usuario o la contraseña incorrectamente	Introduce algún valor que no concuerda con el formato especificado	I usuariono cumple con formato especificado	N/A	N/A	El sistema muestra un mensaje de error en dependencia de los datos introducidos: Para los primeros tres casos->La contraseña debe ser fuerte(ver ayuda).Para el último caso-> Las contraseñas debes coincidir.	Si el usuario introduce valores incorrectos el sistema muestra un mensaje de error en dependencia del error cometido.
		V dani78	I contraseña no cumple con formato especificado	N/A		
		V 3kid34	V Qwer!22k	I Repetir contraseña no cumple con formato especificado		
		V daniel	V Daniel12*	I Repetir contraseña no coincide con campo anterior (Daniel2*)		
EC 1.3 Incetrando usuario y contraseña correctamente	El usuario intruduce el usuario y contraseña correctamente.	V usuario correcto(daniel)	V contraseña correcta(Daniel123*@)	V contraseña correcta(Daniel123*@)	El sistema muestra un mensaje comunicándo que el nuevo usuario se ha creado correctamente.	Luego de introducir los datos correctamente, el sistema crea un nuevo usuario con los valores introducidos y muestra una vista con los dtos del nuevo usuario creado.

Tabla 10: Caso de prueba del CU-Crear Nuevo Usuario.

De manera general durante el proceso de pruebas del sistema propuesto fueron detectadas 10 no conformidades, las cuales fueron resueltas antes de la entrega del producto. A continuación se muestra dicho resultado más detalladamente haciendo uso de una gráfica.

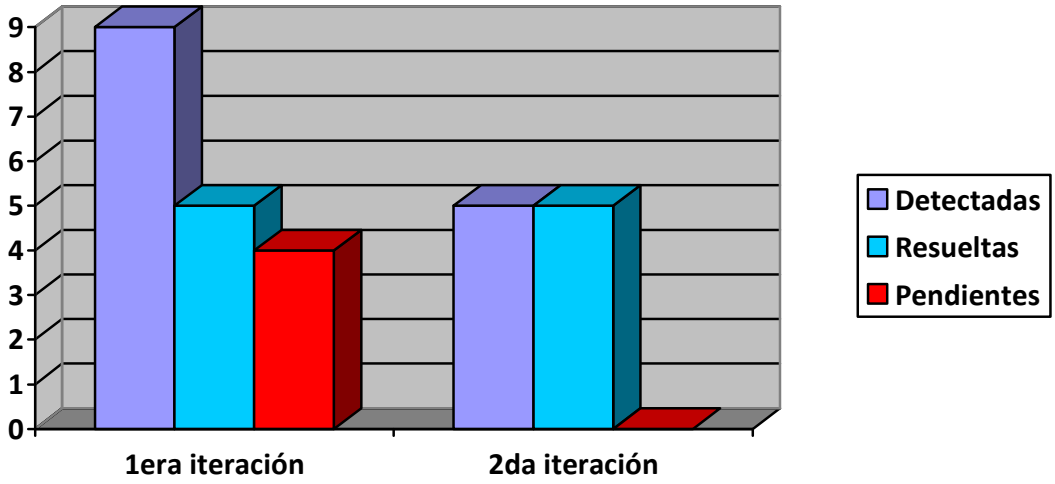


Ilustración 21: Gráfica de no conformidades.

Luego de realizadas estas pruebas, se demostró que el sistema funciona correctamente en cada una de sus iteraciones.

3.4.2 Carga y estrés

El rendimiento es una de las características más importantes de los sistemas informáticos modernos. El análisis de calidad desde el punto de vista de rendimiento incluye las siguientes pruebas.

- La **prueba de estrés** consiste en someter la aplicación a una carga elevada como valores numéricos complejos, elevado número de entradas, elevado número de peticiones etc. que comprueban el límite de carga que la aplicación puede soportar. La prueba de estrés intenta romper el sistema que está siendo sometido a dicha prueba desbordando sus recursos o reduciendo la cantidad de estos. Sus objetivos son encontrar los límites del sistema y asegurar que tras un fallo en el sistema, se recupera sin causar graves problemas.
- La **prueba de carga** generalmente se refiere a la práctica de probar el comportamiento de una aplicación mediante cargas o entradas pesadas para verificar si el sistema satisface los requisitos de rendimiento para situaciones críticas como, por ejemplo, la cantidad límite de usuarios accediendo de forma concurrente a los servicios del programa, documentos extremadamente grandes, cantidad de transacciones que se pueden procesar de forma concurrente cada minuto, tiempo de respuesta, etc. [45].

La propuesta de solución es un sistema que será utilizado por pocos usuarios pues solo será de interés para los administradores de Squid. En cualquier empresa los administradores de red no constituyen un número considerable. No obstante se le realizaron pruebas de carga y estrés a la aplicación web de la propuesta de solución simulando 2, 10 y 100 usuarios conectados a la vez con un intervalo de 1 segundo, arrojando los siguientes resultados.

Muestras	Media	Mediana	Línea de 90%	Mín.	Máx.	%Error	Rendimiento	Kb/seg
38	96	80	168	0	224	0,00%	15,8 seg	243,9

Tabla 11: Resultado de prueba de carga y estrés (primera iteración 2 usuarios).

Muestras	Media	Mediana	Línea de 90%	Mín.	Máx.	%Error	Rendimiento	Kb/seg
190	405	376	836	0	1051	0,00%	18,04 seg	284,2

Tabla 12: Resultado de prueba de carga y estrés (segunda iteración 10 usuarios).

Muestras	Media	Mediana	Línea de 90%	Mín.	Máx.	%Error	Rendimiento	Kb/seg
2100	5662	6102	8235	0	12968	0,00%	16,7 seg	535,8

Tabla 13: Resultado de prueba de carga y estrés (tercera iteración 100 usuarios).

Para visualizar cada reporte que soliciten los usuarios es necesario ejecutar al menos un fichero, lo que hace que el tiempo de respuesta del sistema sea mayor. Teniendo en cuanta lo planteado anteriormente, resultados obtenidos son buenos pues el porciento de error de páginas cargadas fue de 0.00% y el rendimiento (tiempo de respuesta del sistema) osciló entre 15 y 19 segundos.

Pruebas de seguridad

Como el sistema propuesto está desarrollado en Symfony, el cual está pensado para proteger el producto de inyecciones SQL y ataques XSS, no es necesario realizarle pruebas de este tipo. Otra acción que realiza dicho framework, y que favorece en su seguridad, es que comprueba los formularios sin necesidad de programar nada [46]. No obstante se le realizaron una serie de pruebas, arrojando las siguientes conclusiones:

- **Pruebas de Autorización:** Ningún usuario con rol ROLE_USUARIO puede acceder a los recursos a los que no tenga privilegios para hacerlo.
- **Pruebas de Gestión de Sesiones:** No se puede acceder a la aplicación copiando la URL después

de estar autenticado, cerrar la sesión y acceder a la dirección de la URL sin estar autenticado. Además un usuario que no esté autenticado no puede acceder al sistema.

- **Validación de Datos:** Se enmascaran datos confiables cuando se visualizan en la aplicación. Solamente se permiten contraseñas fuertes, que incluyan caracteres especiales, letras mayúsculas y minúsculas, números y que tengan 7 caracteres mínimos de longitud. La funcionalidad de cambio de contraseña únicamente se permite a usuarios autenticados validando la antigua contraseña, la nueva contraseña y el campo de repetir contraseña. El sistema no muestra mensajes indebidos al colocar en la barra de dirección o en campos de entrada los caracteres: (´´, &, +, -, /).
- **Comprobación del Sistema de Autenticación:** Los mensajes de error para distintas combinaciones de autenticación muestran la misma información. Los tiempos de respuestas usuario correcto - contraseña incorrecta y usuario - contraseña incorrecta son los mismos.

La validación de estas pruebas permitió recoger los puntos eficientes y los ineficientes que tienen los elementos chequeados, así como verificar que el grado de seguridad de la aplicación es adecuado para la protección de la información.

3.5 Conclusiones parciales

- El diagrama de componentes permitió obtener una visión más clara sobre la estructura del sistema desarrollado.
- Como resultado de la implementación se obtuvo un sistema funcional y operativo.
- La realización de pruebas al *software* implementado ha permitido validar la solución desarrollada y comprobar que se adapta a los requerimientos definidos.

4 Conclusiones

La realización del presente trabajo ha evidenciado el cumplimiento de los objetivos propuestos. En general se ha arribado a las siguientes conclusiones:

- A partir de la investigación realizada se concluye que aunque los sistemas homólogos no constituyen una solución al problema identificado, representaron un punto de partida y sentaron las bases para el desarrollo del presente trabajo.
- Como resultado de la investigación realizada se diseñaron los reportes del sistema propuestos que dan solución al problema de investigación planteado.
- Se logró desarrollar una aplicación web, encaminada a brindar una solución viable al problema identificado. Dicha aplicación permite a los administradores de red la observación y el seguimiento, de manera sencilla e intuitiva, del rendimiento y desempeño del *proxy* Squid, apoyando en la identificación de posibles fallos en el sistema. Además permite seguir el tráfico de los usuarios que navegan en la red. Además, se determinaron las herramientas y tecnologías para su desarrollo.
- Las pruebas aplicadas permitieron validar que el sistema desarrollado cumple con los requerimientos de *software* que fueron definidos.

5 Recomendaciones

Como resultado del sistema implementado se hacen las siguientes recomendaciones:

- Incorporar nuevas estadísticas al sistema implementado con el fin de poseer un mayor control sobre el funcionamiento de Squid y el tráfico de sus usuarios en Internet.
- Incorporar otros elementos visuales para que el control de estos datos se realice de manera más agradable e intuitiva.
- Incorporar notificaciones automáticas cuando algún parámetro se salga del rango admisible.
- Incorporar la opción de exportar los reportes a otros formatos.

6 Referencias Bibliográficas

1: Gumilla, 2001.

2: Sánchez & Arias, 2007.

3: Definición de Reporte [Sitio oficial: http://definicion.de/reporte/#ixzz2ENIiAwvK] [En línea] [Citado en: enero de 2013].

4: VEASUIP.COM [Sitio Oficial: http://www.veasuip.com/conceptos/concepto_proxy.html] [En línea] [Citado en: octubre de 2012].

5: Phk12.blogspot.com [Sitio Oficial: http://phk12.blogspot.com/2012/05/proxy-nat-ics.html] [En línea] [Citado en: octubre de 2012].

6: Squid-cache.org [Sitio Oficial: www.squid-cache.org] [En línea] [Citado en: octubre de 2012]

7: Barrios Dueñas, Joel. Configuración de Squid: Cachés en jerarquía. [Sitio Oficial: http://www.alcancelibre.org/staticpages/index.php/19-0-como-squid-caches-jerarquia] [En línea] [Citado en: octubre de 2012].

8: Calamaris [Sitio Oficial: http://calamaris.m.softonic.com/linux] [En línea] [Citado en: noviembre de 2012].

9: Sarg Squid Analysis Report Generator. [Sitio oficial: http://sarg.sourceforge.net/] [En línea]. [Citado en: octubre de 2012].

10: Centralizar los Logs de tus servidores [Sitio Oficial: http://carrero.es/centralizar-los-logs-de-tus-servidores/2417/] [En línea] [Citado en: noviembre de 2012].

11: OpenUP como alternativa metodológica para proyectos pequeños de software [Sitio Oficial: http://kasyles.blogspot.com/2008/09/openup-como-alternativa-metodolgica.html] [En línea] [Citado en: febrero de 2012].

12: Red Line Software [Sitio Oficial: http://www.redline-software.com/eng/] [En línea] [Citado en: noviembre de 2012].

13: Controlador de acceso a Internet [Sitio Oficial: http://www.freedownloadmanager.org/es/downloads/Regulador_de_Acceso_de_Internet_2007_51807_p/] [En línea] [Citado en: noviembre de 2012].

14: Munin [Sitio Oficial: http://munin.projects.linpro.no/] [En línea] [Citado en: noviembre de 2012].

15: Pandora FMS [Sitio Oficial: http://pandorafms.org/] [En línea] [Citado en: noviembre de 2012].

16: Cayuqueo, Sergio. Monitoria y análisis de Red con Nagios. Sergio Cayuqueo. [Disponible en:

Referencias Bibliográficas

http://cayu.com.ar/files/manual-nagios-2009.pdf, Sitio oficial: www.nagios.org] [En línea] [Citado en: enero de 2013].

17: The Centreon Software [Sitio oficial: http://www.centreon.com/Centreon/product-overview.html] [En línea] [Citado en: noviembre de 2012].

18: Metodologías de Desarrollo de Software [Sitio Oficial: http://www.ecured.cu/index.php/Metodolog%C3%ADas_de_desarrollo_de_software] [En línea] [Citado en: noviembre de 2012].

19: Metodologías de Desarrollo de Software [Sitio Oficial: http://www2.rhernando.net/modules/tutorials/doc/ing/met_soft.html] [En línea] [Citado en: noviembre de 2012].

20: OpenUP Copyright.||What is OpenUP?|| .2009. [Disponible en: http://epf.eclipse.org/wikis/openup/index.htm] [En línea] [Citado en: enero de 2013].

21: Ingeniería de Software I METODO OPENUP [Sitio Oficial: http://www.slideshare.net/StalinTuza/ingenieriasoftwareopenup] [En línea] [Citado en: noviembre de 2012].

22: PostGreSQL [2011] [Sitio oficial: http://www.postgressql-adsi.blogspot.com/2011/11/ampliamente-popular-ideal-para.html] [En línea] [Citado en: enero de 2013].

23: Base de conocimiento [Sitio Oficial: http://www2.linuxparatodos.net/web/comunidad/base-de-conocimiento] [En línea] [Citado en: diciembre de 2012].

24: POTENCIER, Fabien; ZANINOTTO, François. Symfony, la guía definitiva. Pp.26- . (Citado 20 enero 2013).

25: Libro oficial disponible en: http://symfony.com/doc/2.1/book.

26: Santos, Christian Van Der Henst y Herminio Heredia. ¿Qué es el PHP? 2001. [Disponible en: http://www.maestrosdelweb.com/editorial/phpintro/] [En línea] [Citado en: octubre de 2012].

27: Eguíluz Pérez, Javier. Introducción a JavaScript. [En línea] 2008. [Citado en: enero de 2013.]

28: Bash Reference Manual [Sitio oficial: http://es.wikipedia.org/wiki/Bash] [En línea] [Citado en: enero de 2013].

29: Netbeans [Sitio Oficial: http://netbeans.org/index_es.html] [En línea] [Citado en: diciembre de 2012].

30: Netbeans [Sitio Oficial: http://netbeans.org/community/releases/72/] [En línea] [Citado en: diciembre de 2012].

31: Servidores Web [Sitio Oficial: http://www.monografias.com/trabajos75/servidores-web/_servidores-web.shtml] [En línea] [Citado en: diciembre de 2012].

32: El servidor de web Apache: Introducción práctica [Sitio Oficial: http://www.acsblog.es/articulos/trunk/LinuxActual/Apache/html/x31.html] [En línea] [Citado en: noviembre de 2012].

Referencias Bibliográficas

33: RÍOS, S. S. (2007). Análisis y UML. Modelo conceptual. 2007, Disponible en: http://www.uvmsf.cl/~ssanchez/images/Metodologias/Unidad4_MAD.pdf [En línea] [Citado en: marzo de 2013].

34: Ingenieria De Requerimientos [Sitio oficial: http://www.buenastareas.com/ensayos/Ingenieria-De-Requerimientos/737153.html] [En línea] [Citado en: marzo de 2013].

35: Pressman, Roger S.; 2007. "Ingeniería de software. Un enfoque práctico". 6ta Edición. Parte I.Prólogo y Capítulos 1, 2 y 21. Páginas 1-18, 22-45 y 641-657.

36: SAWMILL: THE ONLY ANALYTICS SOLUTION YOU'LL NEED [Sitio oficial: http://www .sawmill.net] [En línea] [Citado en: enero de 2012].

37: Patrones de diseño y arquitectura [Sitio Oficial: http://www.ecured.cu/index.php/Patrones_de_dise%C3%B1o_y_arquitectura] [En línea] [Citado en: diciembre de 2012].

38: Burbeck, Steve. Application programming in Smalltalk-80: How to use Model-View-Controller (MVC). 1992.

39: Veloso Hernández, Pedro. Uso de patrones de arquitectura. 1996.

40: Patrones en Symfony [Sitio Oficial: http://www.ecured.cu/index.php/Patrones_en_Symfony] [En línea] [Citado en: abril de 2013].

41: Jacaboson, I., Booch, G., Rumbaugh J. El Proceso Unificado de Desarrollo de Software. Addison Wesley, 2000. Páginas 255-256, 282.

42: Flujo de pruebas de un software [2013] [Sitio oficial: http://www.ecured.cu/index.php/Flujo_de_pruebas_de_un_software] [En línea] [Citado en: abril de 2013].

43: Revisiones de código y estándares de codificación [2011] [Sitio oficial: http://msdn.microsoft.com/es-es/library/aa291591%28v=vs.71%29.aspx] [En línea] [Citado en: marzo de 2013].

44: Mario Barrientos, Yamila Nieves Hernández. Automatización de Pruebas Funcionales para Aplicaciones Web. Universidad de las Ciencias Informáticas. Habana, 2010. 106.

45: Testeo de estrés y carga [Sitio: http://www.iti.es/servicios/servicio/resource/7240/index.html] [En línea] [Citado en: abril de 2013].

46: Mi primer ¡Hola Mundo! con Symfony [Sitio: http://www.interadictos.es/2010/04/19/mi-primer-%C2%A1hola-mundo-con-symfony/] [En línea] [Citado en: abril de 2013].

Anexos

7 Anexo I

Ilustración 22: Diagrama de clases del Análisis CU Visualizar reporte de Cantidad de Objetos en Cache.

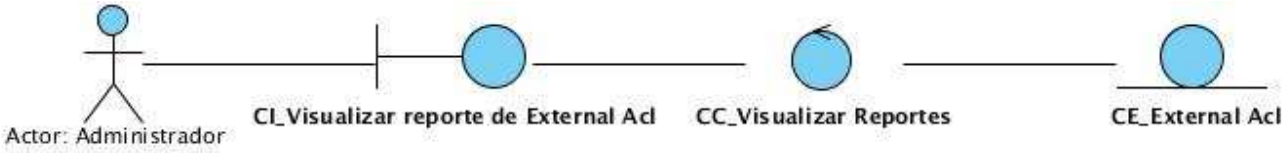


Ilustración 23: Diagrama de clases del Análisis CU Visualizar reportes de Cache External ACL.

Ilustración 24: Diagrama de clases del Análisis CU Visualizar reportes de conexiones concurrentes.

Ilustración 25: Diagrama de clases del Análisis CU Visualizar reportes de delay pools.

Ilustración 26: Diagrama de clases del Análisis CU Visualizar reportes de http recibidas.

Ilustración 27: Diagrama de clases del Análisis CU Visualizar reportes de Número de clientes Conectados.

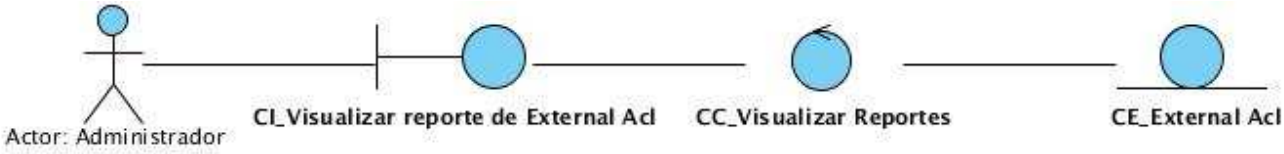


Ilustración 28: Diagrama de clases del Análisis CU Visualizar reportes de Cache External Acl.

Ilustración 29: Diagrama de clases del Análisis CU Visualizar reportes de tráfico en red.

Ilustración 30: Diagrama de clases del Análisis CU Visualizar reportes de uso del disco.

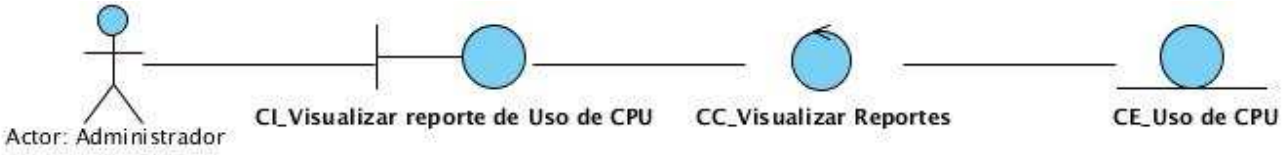


Ilustración 31: Diagrama de clases del Análisis CU Visualizar Uso de CPU.

8 Anexo II

Estándar de codificación

Un estándar completo de codificación comprende todos los aspectos de la generación de código. Si bien los programadores deben implementar un estándar de forma prudente, éste debe tender siempre a lo práctico. Un código fuente completo debe reflejar un estilo homogéneo, como si un único programador

hubiese escrito todo el código de una sola vez.

La legibilidad del código fuente repercute directamente en lo bien que un programador comprende un sistema de *software*. El mantenimiento del código es la facilidad con que el sistema de *software* puede modificarse para añadirle nuevas características, modificar las ya existentes, depurar errores, o mejorar el rendimiento [43].
Para facilitar el entendimiento del código y fijar un modelo a seguir se establecieron estándares de codificación los cuales se describen a continuación.

8.1 Identificadores

En el caso de los identificadores existen estilos definidos mundialmente como el lowerCamelCase y el UpperCamelCase. Cada palabra interna en identificadores compuestos comienza con mayúsculas para ambos estilos; además, ocurre que no se colocan caracteres de separación entre las palabras que conforman un identificador compuesto en ninguno de los dos casos. Para el primero, el identificador comienza con minúscula y para el segundo, el identificador comienza con mayúscula.

- ✓ Clase: para las clases se escogió el UpperCamelCase.
  Ejemplo: class DefaultController
- ✓ Función: para las funciones se escogió el lowerCamelCase.
  Ejemplo: public function indexAction ()
- ✓ Variable: para las variables se seleccionó el empleo del estándar lowerCamelCase.
  Ejemplo: $lineasAcl

8.2 Indentación

La indentación es una práctica de programación que consiste en comenzar a escribir cada línea de código a diferentes distancias desde el borde izquierdo del área de texto del editor. Esta distancia está determinada por la jerarquía que se forma al introducir sentencias dentro de bloques de estructuras. Brinda mayor legibilidad y entendimiento para el programador e igualmente depende del estilo propio de cada persona, del lenguaje y del tipo de programa que se implementa. El Netbeans, IDE utilizado en el desarrollo de la aplicación, se encarga de ajustar los espacios de manera automática.

Por otra parte se escribirá sólo una sentencia por línea de código y en el caso de cortar las líneas, se hará luego de una coma o antes de un operador. La sección de la derecha de la línea que se corte se ubicará

en la línea siguiente indentada al nivel de la expresión correspondiente en la línea superior.

El cuerpo de declaraciones compuestas (declaraciones como “if”, “while”, “for”, etc.) debe anidarse a una profundidad de 4 espacios.

8.3 Llaves

Existe diversidad de criterios en cuanto a la ubicación de las llaves que delimitan el cuerpo de los bloques de código. Algunos programadores prefieren ubicar la llave de apertura inmediatamente detrás de la línea cabecera del bloque, mientras otros apuestan por ubicarlas de forma solitaria en la línea siguiente a la línea cabecera. Existen además diferencias en cuanto al nivel de indentación de las mismas; algunos lo hacen al nivel de la línea cabecera y otros al nivel de las líneas del cuerpo del bloque. Para el trabajo las llaves de apertura de las funciones y las llaves de aperturas de las estructuras for, if, while, else, se colocarán inmediatamente detrás de la línea cabecera del bloque. Las llaves de cierre se colocarán solitarias en la línea que sigue a la última línea dentro del bloque e indentadas al nivel de la línea cabecera del bloque.

Ejemplo del uso de llaves:

```
public function trafico_en_redAction() {
    $command = "cd " . $this->getRequest()->server->get('DOCUMENT_ROOT') . $this->getRequ
    exec($command);
    $usuarios = file('/var/www/Ripoll/web/rrd/usuario1');
    $lineas = file('/var/www/Ripoll/web/rrd/trafico.txt');
    $ast = file('/var/www/Ripoll/web/rrd/usua');
    $nuevas = array();
    $mio = array();
    foreach ($lineas as $value) {
        if ($value != $ast[0]) {
            $nuevas[] = $value;
        }
        else {
            $mio[] = $nuevas;
            $nuevas = array();
        }
    }
    $limpiartra = fopen("/var/www/Ripoll/web/rrd/trafico.txt", "w+");
    fclose($limpiartra);
    return $this->render('ReportesBundle:Default:graficar_trafico_en_red.html.twig',
            array(
                'usuarios' => $usuarios,
                'mio' => $mio,
            ));
}
```

Ilustración 32: Ejemplo del uso de llaves.

8.4 Líneas y espacios en blanco

Para mejorar la legibilidad y organización del código muchas veces se utilizan líneas en blanco para separar segmentos de código que pueden corresponder a clases, funciones, declaraciones, implementaciones, comentarios, bloques o sencillamente secciones críticas que se deseen despejar. Así mismo sucede con los espacios en blanco cuando se utilizan para separar elementos dentro de las sentencias de código. En ocasiones se separan con espacios cada operador de su respectivo operando, paréntesis, identificadores, símbolos y algunos lenguajes exigen que se separen las palabras propias del vocabulario de las adyacentes para ser comprendidas por los compiladores. En el trabajo se ha definido emplear líneas en blanco:

- ✓ Entre funciones.

  Ejemplo de línea en blanco entre funciones:

```
public function indexAction() {...}
public function indexUsuarioAction() {...}
public function indexAdminAction() {...}
public function loginAction() {...}
public function ayudaAction() {...}
public function contrasenaAction() {...}
```

Ilustración 33: Ejemplo de línea en blanco entre funciones.

- ✓ Entre declaraciones de variables e implementaciones dentro del cuerpo de las funciones.

  Ejemplo de espacio entre declaraciones de variables e implementaciones:

```
$nuevas = array();
$mio = array();
foreach ($lineas as $value) {
    if ($value != $ast[0]) {
        $nuevas[] = $value;
    }
    else {
        $mio[] = $nuevas;
        $nuevas = array();
    }
}
```

Ilustración 34: Ejemplo de espacio entre declaraciones de variables.

Se colocarán espacios en blanco:

- ✓ Después de las comas en la lista de argumentos de las funciones.

  Ejemplo de espacios en blanco en las listas de argumentos de las funciones:

```
public function aux($arr, $pos){...}
```

Ilustración 35: Espacios en blanco en argumentos de las funciones

- ✓ Después de cada punto y coma (;) en las estructuras ***for***.
- ✓ Entre los signos lógicos y de operación, antes y después del mismo.
- ✓ No se debe utilizar:
  - Entre el nombre de un método y el paréntesis abierto.
  - Después del corchete abierto y antes del cerrado de un arreglo.

8.5 Declaraciones

- ✓ Clases: Las directivas de visibilidad deben estar indentadas con la definición de la clase. Estas directivas deben declararse en el orden siguiente: private, protected, public.
- ✓ Variables locales: Las variables deben ser declaradas justo antes de ser utilizadas.

  Cuando sea posible, la inicialización de las variables locales ocurrirá en el momento de su

declaración. Se debe evitar la declaración dentro del cuerpo de un ciclo.

- ✓ Condicionales: Las declaraciones “if” – “else” no deben poseer más de 4 niveles de profundidad, de tenerlos no deben estar en una línea sino repartidos en igual número de líneas como niveles tenga, esto se explica porque es más fácil leer una declaración “if” – “else” de arriba para abajo que de izquierda a derecha. La cláusula else siempre debe estar alineada con la cláusula “if”.

```
foreach ($lineas as $value) {
    if ($value != $ast[0]) {
        $nuevas[] = $value;
    }
    else {
        $mio[] = $nuevas;
        $nuevas = array();
    }
```

Ilustración 36: Ejemplo declaraciones if/else.

- ✓ “for”, “foreach”: El código de inicialización debe realizarse en la declaración del mismo, a menos que sean necesario declararlo fuera para darle más tiempo de vida a la variable de control.
- ✓ “do”... “while”: El código de la declaración está acotado por las palabras reservadas: “do”... “while”. Se utilizará cuando se necesite explícitamente que se cumpla al menos una vez el ciclo post condicional.