Universidad Central de Venezuela
Facultad de Ciencias
Escuela de Computación
Laboratorio de Comunicación y Redes

BIBLIOTECA PARA LA ADMINISTRACIÓN DE REDES BASADAS EN TCP/IP MEDIANTE LA UTILIZACIÓN DEL PROTOCOLO SNMP v1/v2c PARA EL LENGUAJE DE PROGRAMACIÓN PHP

Trabajo Especial de Grado
presentado ante la Ilustre
Universidad Central de Venezuela
por los Bachilleres:

Joselyn C. Camero.
C.I.: 17.438.786
E-mail: joselync19@gmail.com

Laura A. Uzcátegui J.
C.I.: 17.588.310
E-mail: laura608@gmail.com

para optar al título de Licenciado en Computación

Tutores:
Prof. Eric Gamess
Prof. Karima Velásquez

Caracas, Octubre 2008

Universidad Central de Venezuela
Facultad de Ciencias
Escuela de Computación
Laboratorio de Comunicación y Redes

ACTA DEL VEREDICTO

Quienes suscriben, Miembros del Jurado designados por el Consejo de Escuela de Computación, para examinar el Trabajo Especial de Grado, presentado por los Bachilleres Joselyn Camero 17.438.786 y Laura A. Uzcátegui J. C.I.: 17.588.310, con el título "**Biblioteca para la Administración de Redes basadas en TCP/IP mediante la utilización del Protocolo SNMP v1/v2c para el Lenguaje de Programación PHP**", a los fines de cumplir con el requisito legal para optar al título de Licenciado en Computación, dejan constancia de lo siguiente:

Leído como fue dicho trabajo por cada uno de los Miembros del Jurado, se fijó el día 31 de octubre de 2008, a las 01:30 PM, para que sus autores lo defiendan en forma pública, en el laboratorio de Internet2, mediante la exposición oral de su contenido, y luego de la cual respondieron satisfactoriamente a las preguntas que le fueron formuladas por el Jurado, todo ello conforme a lo dispuesto en la Ley de Universidades y demás normativas vigentes de la Universidad Central de Venezuela. Finalizada la defensa pública del Trabajo Especial de Grado, el jurado decidió aprobarlo.

En fe de lo cual se levanta la presente Acta, en Caracas a los 31 días del mes de octubre del año dos mil ocho, dejándose también constancia de que actuaron como Coordinadores del Jurado el Profesor Tutor Eric Gamess y la Prof. Karima Velásquez.

Prof. Eric Gamess (Tutor)	Prof. Karima Velásquez (Tutor)
Prof. Luis Manuel Hernández (Jurado Principal)	Prof. Ana Romero (Jurado Principal)

RESUMEN

TÍTULO:
Biblioteca para la Administración de Redes basadas en TCP/IP mediante la utilización del Protocolo SNMP v1/v2c para el Lenguaje de Programación PHP.

AUTORES:
Joselyn C. Camero.
Laura A. Uzcátegui J.

TUTORES:
Prof. Eric Gamess
Prof. Karima Velásquez

En este trabajo especial de grado se diseñó y desarrolló una biblioteca denominada Net_SNMP, para la administración de redes basadas en TCP/IP por medio del protocolo SNMP, que brinda soporte al lenguaje PHP. Net_SNMP puede utilizar IPv4 e IPv6 como protocolo de red.

El trabajo se inició con un estudio teórico práctico del protocolo SNMP obteniendo así la base necesaria para el desarrollo de la biblioteca. Una vez finalizado este estudio, se dio paso a la elección de una metodología de implementación que se adaptara al proceso de desarrollo.

La metodología seleccionada se denomina Programación Extrema y está basada en un proceso iterativo e incremental, que cuenta con las fases de Planificación, Diseño, Implementación y Pruebas. Esta metodología emplea una planificación abierta y flexible, permitiendo aumentar la habilidad de respuesta ante los cambios que surjan a lo largo del proyecto y disminuyendo así la complejidad del desarrollo.

Net_SNMP tiene soporte para SNMP en sus versiones 1 y 2c. Además de las primitivas básicas del protocolo que fueron desarrolladas, también fue implementada la operación *Walk* que permite recorrer un subárbol de la MIB. La biblioteca permite el envío de notificaciones, registro de eventos en un log y proporciona un mecanismo de manejo de errores a través de la definición de excepciones.

Además, este trabajo incluye 2 herramientas web denominadas NMSWeb y RMSWeb las cuales hacen uso de Net_SNMP y fueron desarrolladas con la finalidad de validar la biblioteca.

Palabras Clave: SNMP, Administración de Redes, PHP, IPv4, IPv6.

Tabla de Contenido

Índice de Figuras

Índice de Tablas

Introducción

En un principio, al momento de surgir las redes de computadores, su finalidad no era más que la comunicación entre entes gubernamentales. No obstante, este concepto ha ido cambiando a medida que el tiempo avanza y las tecnologías de comunicación evolucionan, dando paso a la gran interconexión que existe en la actualidad a nivel mundial por medio de las redes. Por este motivo, no se puede dejar de tomar en cuenta los aspectos de seguridad, mantenimiento y revisión en relación a los componentes que la conforman, con el objetivo de aumentar su disponibilidad.

La administración y monitoreo constante de los elementos de red, pueden llegar a jugar un papel fundamental en el rendimiento y calidad de éstas. En consecuencia, se definió un protocolo de comunicación que permitiera realizar de manera rápida y sencilla las actividades relacionadas con la gestión de distintos dispositivos en cualquier red. Este protocolo se denomina SNMP (*Simple Network Management Protocol – Protocolo Simple de Administración de Red*), y fue definido por los comités técnicos de Internet para supervisar, administrar y monitorear una red.

SNMP ha sido implementado en varios lenguajes de programación, a través de los cuales se brinda a los desarrolladores la facilidad de diseñar una gran cantidad de herramientas, para la administración de redes, que puedan dar respuestas a requerimientos particulares.

Por otra parte, se sabe que en la actualidad la mayoría de las aplicaciones están siendo migradas hacia el ambiente web. Uno de los lenguajes que posee mayor aceptación en el desarrollo de herramientas web es PHP[1]. De allí la importancia de que este lenguaje proporcione una biblioteca, lo suficientemente completa y portable, para dar soporte al protocolo SNMP.

Por esta razón, en este trabajo especial de grado, se diseñó e implementó una biblioteca en PHP, que facilite a los programadores la elaboración de aplicaciones para la gestión y administración de redes basadas en TCP/IP, a través del protocolo SNMP v1/v2c.

Adicionalmente, se desarrollaron dos aplicaciones web para la administración de redes, que hacen uso de la biblioteca, con el propósito de validar el correcto funcionamiento de los métodos implementados.

[1] http://www.desarrolloweb.com/articulos/436.php

El resto de este documento ha sido organizado como se especifica a continuación. En el capítulo 1 se presentan la descripción, los objetivos, la justificación y el alcance del trabajo especial de grado. En el capítulo 2 se describen las bases teóricas necesarias para el desarrollo del proyecto. El capítulo 3 aborda conceptos relacionados con la metodología seleccionada para la implementación de la biblioteca de clases y las aplicaciones web. En el capítulo 4 se explica el proceso llevado a cabo durante el desarrollo del proyecto. Finalmente, el capítulo 5 presenta las conclusiones del trabajo especial de grado.

1. El Problema

1.1 Introducción

La gestión de redes representa una labor importante dentro de las organizaciones en las que se requiera supervisión constante de los dispositivos que se encuentran interconectados. Esta tarea permite al administrador mantener control sobre la red y facilita la detección de cualquier tipo de inconveniente que pueda perjudicar la estabilidad de la misma.

Entre los mecanismos existentes que son utilizados para la administración de redes, se destaca el protocolo SNMP; que permite el envío de datagramas UDP para obtener información sobre el estado de algún dispositivo que se encuentre en la red.

En la actualidad, existen múltiples herramientas que hacen uso del protocolo SNMP. Sin embargo, un programador puede tener la necesidad o motivación de desarrollar una aplicación que cubra un conjunto de requerimientos específicos. Es por ello que los lenguajes de programación deben proporcionar características que faciliten la implementación de este tipo de aplicación.

1.2 Planteamiento del Problema

Hoy en día existen diversos software que brindan soporte para el protocolo SNMP, facilitando al administrador la tarea de consultar las variables de los agentes y monitorear la red. Asimismo, se presenta una fuerte tendencia de migrar la mayoría de las aplicaciones desarrolladas hacia un ambiente web, por factores como la alta penetración del Internet, y el aumento de la movilidad, entre otros. Las herramientas SNMP no escapan a esta situación.

Debido a esto, los lenguajes de programación orientados al desarrollo web están en la obligación de proporcionar mecanismos que permitan el uso de este protocolo, así como de otras funciones de red, simplificando la tarea de los programadores. Entre estos lenguajes destaca PHP por su gran popularidad y aceptación[1].

[1] http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html

Sin embargo, el lenguaje PHP no posee una biblioteca de funciones para el manejo del protocolo SNMP lo suficientemente completa y documentada. En su lugar, posee una extensión[2] que brinda soporte para dicho protocolo, en la cual se pueden observar deficiencias en aspectos tales como la portabilidad e instalación de la misma. Gran parte de las operaciones proporcionadas por esta extensión no funcionan en los sistemas operativos para los cuales fue desarrollada, es decir, algunas operaciones que se ejecutan sobre plataformas Unix no funcionan sobre plataformas Windows, y viceversa.

Por otra parte, la documentación que posee la extensión SNMP para PHP es escasa y no otorga los detalles necesarios que debería poseer el programador para familiarizarse con su manejo.

De acuerdo a lo mencionado anteriormente, surge la necesidad de diseñar e implementar una biblioteca de funciones con soporte para los protocolos SNMPv1 y SNMPv2c, que pueda ser utilizada bajo sistemas Windows y Unix, y que además posea una documentación lo suficientemente amplia y entendible para el usuario.

1.3 Justificación

Desde sus inicios, el lenguaje de programación PHP ha sido construido bajo la colaboración de múltiples usuarios y desarrolladores que han contribuido al crecimiento del mismo. Un ejemplo de esto es el repositorio de extensiones y aplicaciones PEAR, el cual alberga una amplia gama de módulos y clases de codificación estándar y de fácil instalación. A pesar de que en este repositorio existe soporte para diversos protocolos de red, el protocolo SNMP no se encuentra incluido.

Aunque en la actualidad hay diversas herramientas que facilitan la administración y supervisión de redes, debido a lo amplia y extensa que es esta actividad es posible que surjan requerimientos particulares, que obliguen al programador a realizar aplicaciones que se adapten a necesidades específicas. De allí la importancia de brindar operaciones que simplifiquen el manejo y uso del protocolo SNMP, por parte de los desarrolladores.

En vista de esta situación, se requiere proporcionar una biblioteca de funciones SNMP para el lenguaje PHP, que cumpla con los estándares

[2] http://ve.php.net/manual/es/book.snmp.php

establecidos en el repositorio PEAR, y que adicionalmente sea portable, fácil de instalar, con una amplia documentación, y de un alto nivel y calidad.

1.4 Objetivos

1.4.1 Objetivo General

Desarrollar una biblioteca de clases en PHP, destinada a la gestión y administración de redes basadas en TCP/IP, mediante la utilización del protocolo SNMP v1/v2c.

1.4.2 Objetivos Específicos

- Realizar un estudio correspondiente a las operaciones del protocolo SNMP en sus versiones 1 y 2c.
- Evaluar las funciones SNMP existentes en el lenguaje de programación PHP.
- Diseñar la biblioteca de funciones SNMP v1/v2c para el lenguaje PHP.
- Implementar la biblioteca de funciones SNMP v1/v2c, de acuerdo a los estándares de codificación establecidos.
- Crear una documentación que especifique las funciones desarrolladas en la biblioteca y su uso.
- Probar la biblioteca de funciones implementada, a través del desarrollo de una aplicación con interfaz web para la administración de dispositivos de red, orientada a usuarios con conocimientos del protocolo SNMP.
- Probar la biblioteca de funciones implementada, a través del desarrollo de una aplicación con interfaz web para la administración de un *router Cisco Systems* 2811.
- Publicar la biblioteca de funciones y las aplicaciones web desarrolladas, como un producto de software libre bajo la licencia GNU GPL.

1.5 Alcance

La biblioteca de clases diseñada deberá proporcionar operaciones capaces de gestionar y administrar redes basadas en TCP/IP a través de los protocolos SNMP v1/v2c. Esta biblioteca funcionará para el lenguaje PHP versión 5.0, bajo los sistemas operativos *Windows* y *Unix*.

2. Marco Teórico

2.1 Antecedentes de la Administración de Redes

Durante el auge de la red ARPANET (*Advanced Research Projects Agency Network*), creada por el Departamento de Defensa de los Estados Unidos para establecer comunicaciones entre los organismos de este país, en caso de notarse cierto retardo en alguna estación de la red tan sólo bastaba con hacer un *ping* para monitorear si existían fallas en alguno de los nodos, y poder tomar las acciones pertinentes [1].

Una vez que la red ARPANET se convierte en Internet, ya no fue suficiente hacer *ping* debido al número de computadores y a la complejidad de la red, por lo que surgieron nuevas herramientas o protocolos para la administración de las redes. El primero de ellos, denominado SGMP (*Simple Gateway Management Protocol*), fue definido en el RFC (*Request For Comment*) 1028. Este protocolo permite la recuperación o asignación de valores de variables en las cuales reside la información de estado del dispositivo, útil para la administración de *routers* [2].

SGMP fue reemplazado por el protocolo SNMP (*Simple Network Management Protocol*), definido inicialmente en el RFC 1067 [3]. En la actualidad existen tres versiones de SNMP (SNMPv1, SNMPv2c, SNMPv3).

2.2 SNMP (Simple Network Management Protocol)

SNMP (*Simple Network Management Protocol – Protocolo Simple de Administración de Red*), como sus siglas lo indican, es un protocolo de gestión de redes perteneciente a la capa de aplicación del modelo OSI, que facilita las tareas de administración relacionadas a una red mediante el intercambio de información entre los dispositivos que la conforman. Esta información es útil para:

- Supervisar el rendimiento de la red.
- Encontrar soluciones ante cualquier evento, falla o problema.
- Configurar dispositivos remotos.
- Auditar el uso de la red, con el fin de identificar el acceso de grupos y/o usuarios a los servicios.

De esta manera, es posible que los administradores de la red tengan la capacidad de consultar o cambiar el estado de algún dispositivo, a través del conjunto de operaciones proporcionadas por el protocolo SNMP. SNMP, a diferencia del protocolo SGMP, no sólo realiza la gestión e intercambio de información entre *routers*, sino que además lo hace en todos los dispositivos

que soporten el protocolo, que incluyen la mayoría de los dispositivos de red (*switches*, *routers*, impresoras, entre otros) y los sistemas operativos (*Unix, Windows, Mac OS,* entre otros).

Desde que este protocolo fue desarrollado en 1988 [4], se ha convertido en un estándar de *facto* en lo que a gestión de redes se refiere. Entre las razones para justificar esta estandarización se incluyen:

- Es una solución simple.
- Poco código a implementar.
- Es extensible, lo que permite a fabricantes agregar nuevas funciones de gestión a sus dispositivos.
- Es independiente de la arquitectura.

2.2.1 Arquitectura del protocolo

En general, el modelo arquitectónico para SNMP es una colección de estaciones de gestión de red denominadas NMS (*Network Management System*), y elementos de red. Los NMSs ejecutan aplicaciones de gestión con el fin de monitorear y controlar los elementos de red.

Estos elementos de red en realidad son dispositivos, tales como estaciones de trabajo, servidores, *switches*, *routers*, impresoras, entre otros. Cada uno de ellos posee un agente de gestión de red (*Agent*), responsable de ejecutar las funciones solicitadas por los NMSs.

Finalmente, el protocolo SNMP es el responsable de comunicar y transmitir la información entre los NMSs y los agentes.

La arquitectura descrita anteriormente tiene como objetivos:

- Minimizar la cantidad y complejidad de las funciones realizadas por el agente.
- Ser extensible, de tal forma que puedan adaptarse a posibles cambios en las funciones de red, relacionadas a la gestión de la misma.
- Ser en la medida de lo posible independiente de la arquitectura y de los mecanismos que utilicen los dispositivos.

Para ello, cuenta con los siguientes elementos que forman parte del modelo SNMP:

- Dispositivo a ser gestionado.
- Agente de gestión de red.
- Información de gestión.

- Sistema de gestión de red.

En la Figura 2.1 se puede observar el modelo descrito anteriormente.

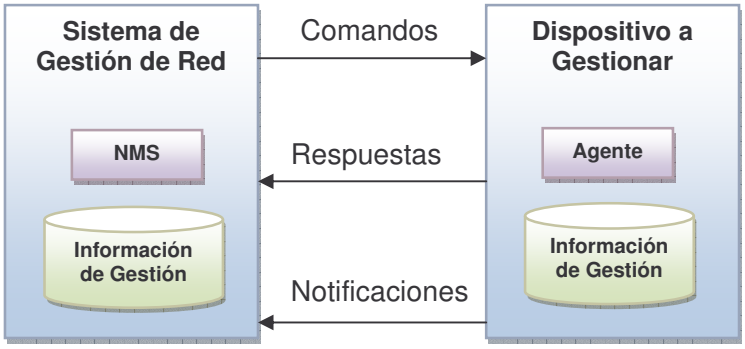

Figura 2.1: Modelo de arquitectura SNMP.

El NMS se encarga de proveer una interfaz entre el administrador de la red y el dispositivo a gestionar. Generalmente en este sistema se despliega alguna salida gráfica en la que se pueden observar datos de gestión, estadísticas de la red, entre otros. Este sistema, mediante un *software* especial que provee las funciones de gestión de red, se comunica con el agente ubicado en el dispositivo a gestionar enviando consultas y recibiendo respuestas del estado en el que se encuentra el dispositivo, para así tomar acciones cuando se considere necesario [5].

El dispositivo a gestionar mantiene un proceso activo denominado *agente*, el cual realiza operaciones de gestión de red tales como modificación de parámetros, estadísticas operacionales del dispositivo, entre otros. Adicionalmente el agente se comunica con el sistema de gestión de red, mediante un mecanismo denominado *Trap*, a través del cual es posible notificar que algún evento ha ocurrido en el dispositivo. En este caso, se envían mensajes de forma asíncrona y no como una respuesta a alguna consulta hecha por el administrador.

El dispositivo contiene objetos a gestionar que pueden ser parámetros de configuración, estadísticas de rendimiento, entre otros. Una colección de estos objetos es definida como la base de datos de información de gestión o MIB (*Management Information Base).* La MIB está asociada tanto con el agente como con el administrador, y tiene una estructura de almacenamiento y recuperación de los datos mediante una organización lógica, la cual se conoce como estructura de información de gestión o SMI (*Structure of Management Information*).

2.3 ASN.1 (Abstract Syntax Notation One)

La parte fundamental del modelo SNMP es el conjunto de objetos que manejan los agentes, los cuales pueden ser leídos y modificados por los NMSs. Sin embargo, es necesario que estos objetos sean definidos de manera estándar y lo más neutral posible, en cuanto a fabricantes de dispositivos se refiere. Por esta razón, se adaptó un subconjunto del lenguaje de definición de objetos estándar como las reglas de codificación conocido como ASN.1 (*Abstract Syntax Notation One*).

ASN.1 es la notación formal usada para describir las estructuras de datos transmitidos por protocolos de comunicaciones, por ejemplo SNMP, sin importar la implementación ni representación física de estos datos, o que tan compleja o simple sea la aplicación. Esta sintaxis ha sido adoptada por un gran conjunto de aplicaciones como: administración de redes, correo seguro, telefonía celular, control de tráfico aéreo, voz y video sobre Internet [6].

2.3.1 Descripción de tipos de datos y estructuras

- **Types:** Representa el conjunto de valores no vacío, que puede ser codificado y transmitido. Se pueden definir tipos complejos usando los tipos básicos. Cuando se define un tipo, el mismo debería tener un nombre que lo referencie.

  En la Tabla 2.1 se pueden observar los tipos de datos básicos definidos para ASN.1.

Nombre	Descripción
NULL	Es usado para colocar el valor a alguna alternativa del tipo elección (*CHOICE*).
INTEGER	Números tanto positivos como negativos.
ENUMERATED	Enumeración de identificadores.
BIT STRING	Usado para transmitir datos binarios de tamaño variable.
OCTECT STRING	Usado para transmitir datos binarios, pero la transmisión es en múltiplos de 8 bits.
OBJECT IDENTIFIER, RELATIVE OID	Identificación sin ambigüedad de una entidad registrada en la MIB.

Tabla 2.1: Tipo de datos básicos en ASN.1 para SNMP.

- **Values:** Son los valores que se referencian para tipos que han sido definidos por el usuario. Estos valores deben ser referenciados por un nombre que comience en minúscula y deben representar un tipo que comience por mayúscula.

- **OBJECT IDENTIFIER:** Son los nombres que se le dan a los objetos que pertenecen al árbol. Estos OIDs identifican a los objetos unívocamente. En la Figura 2.2 se observa un ejemplo en el que se tiene un árbol, y la identificación del objeto *payroll* a través del OID 1.2.0.0.6.2.

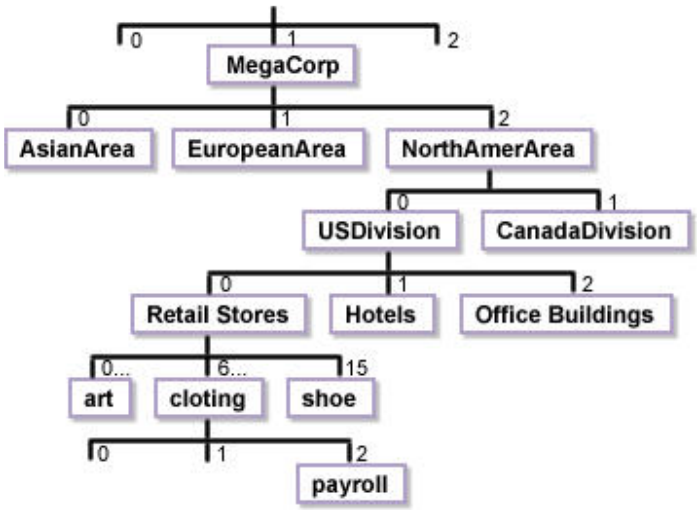

Figura 2.2: Ejemplo de OBJECT IDENTIFIER.

- **Tags:** (Los tags) o etiquetas, están asociadas con los tipos de un módulo de ASN.1. Estas se encuentran representadas por un par formado por una clase etiquetada y un número. Son útiles debido a que evitan la ambigüedad, de esta manera dos etiquetas son diferentes si pertenecen a distintas clases con números distintos.

  ASN.1 define 4 clases de etiquetas: *UNIVERSAL*, *CONTEXT-SPECIFIC*, *APPLICATION*, *PRIVATE*. La clase *CONTEXT-SPECIFIC* permite definir la codificación para las primitivas de las operaciones del protocolo SNMP, tales como *GetRequest*, *GetNextRequest*, *GetResponse*, *SetRequest* y *Trap*.

2.3.2 Reglas básicas de codificación (BER, Basic Encoding Rules)

ASN.1 tiene definido un estándar que proporciona reglas para codificar sus tipos de datos, así como sus estructuras. Este estándar se denomina BER (*Basic Encoding Rules)*.

Las BERs siguen un formato que se encuentra representado por una tripleta denominada *TLV* (*Type, Length, Value*) que está formada por los siguientes componentes:

- *Type.*
- *Length.*
- *Value.*

En la Figura 2.3 se tiene la representación gráfica de la TLV.

Figura 2.3: Tripleta TLV.

A su vez, una TLV puede contener dentro de su definición, específicamente en el campo *value*, otra u otras TLVs. Esto se debe a que este valor puede ser un tipo de dato complejo, como se observa en la Figura 2.4.

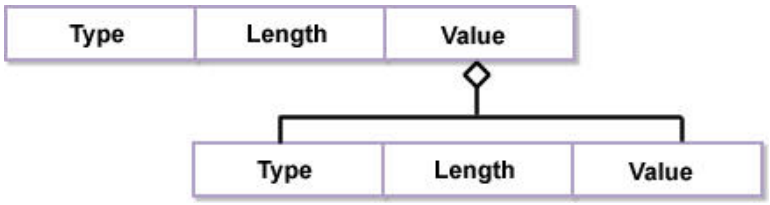

Figura 2.4: Tripleta TLV Recursiva.

A continuación se describen cada uno de los campos que conforman la TLV.

- ***Type*:** Este octeto tiene una etiqueta que identifica el tipo de dato que se está codificando, para ello hace uso de los bits del octeto como se observa en la Figura 2.5.

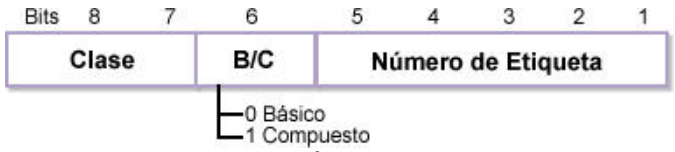

Figura 2.5: Representación del campo Type en la TLV.

- ***Length*:** Indica el número de octetos que contendrá el campo *Value*. Puede representarse de dos maneras: la *definida*, en la que se especifica la longitud del campo *Value* antes de realizarse la transmisión; y la *indefinida*, cuando no se conoce la cantidad de octetos del campo *Value* antes de la transmisión.

- ***Value*:** Para definir este campo es necesario tomar en cuenta el tipo de dato que se está representando.

Entre los tipos de datos más comunes utilizados por el protocolo SNMP, se encuentran:

- ***NULL*:** Se emplea para representar un valor que no contiene octetos. En la Figura 2.6 se tiene un ejemplo de su definición*.*

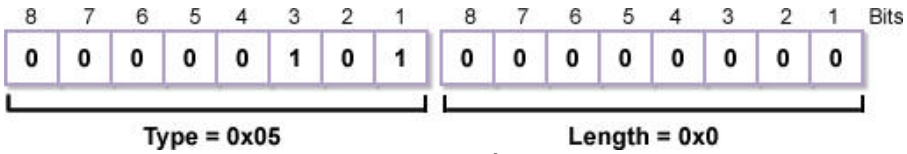

Figura 2.6: Representación del *Value NULL*.

- ***INTEGER*:** Representa un entero. Su valor es codificado en binario complemento-2, utilizando el número de octetos que sean necesarios. Si el campo contiene más de un octeto entonces los bits del primer octeto y el octavo bit del segundo octeto no podrán ser todos uno o no podrán ser todos cero, asegurando que el entero sea codificado en el menor número de octetos. En la Figura 2.7 se puede apreciar un ejemplo de su codificación para el entero 75.

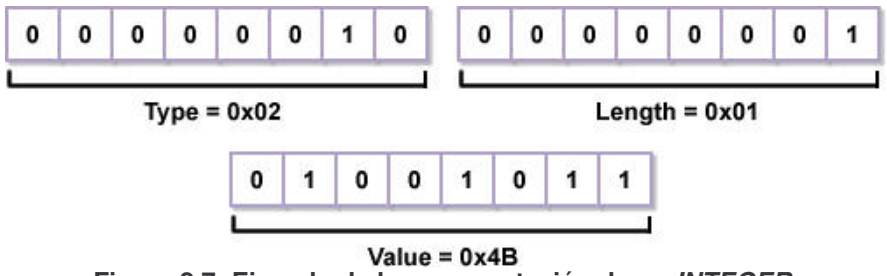

Figura 2.7: Ejemplo de la representación de un *INTEGER*.

- ***OCTET STRING*:** Es un tipo básico cuyo valor depende de la información codificada. En la Figura 2.8 se tiene un ejemplo de la codificación del *OCTET STRING* “BM” utilizando *ASCII* (*American Standard Code for Information Interchange*).

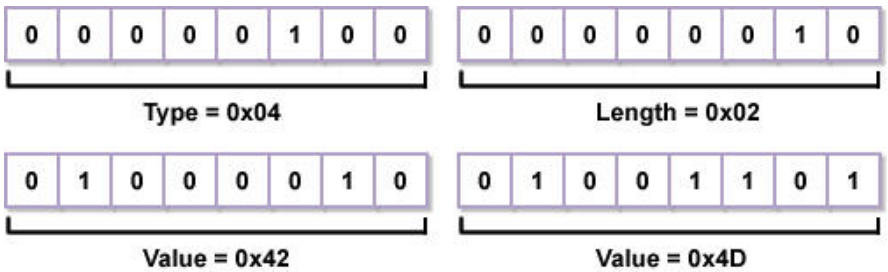

Figura 2.8: Ejemplo de la representación de un *OCTET STRING*.

- ***OBJECT IDENTIFIER*:** El protocolo SNMP emplea este valor para identificar los objetos administrados. En la codificación el campo *Value* contiene una lista de los subidentificadores que conforman el OID. Los dos

primeros subidentificadores se combinan en una sola cifra, a través de la expresión matemática que se observa en la Figura 2.9.

```
Sea X el primer subidentificador, y Y el segundo:

Primer Subidentificador a codificar = (X *40) + Y
```

Figura 2.9: Expresión matemática para codificar un *OBJECT IDENTIFIER*.

Luego cada subidentificador es codificado en octetos. El primer bit indica si el octeto es el último requerido para representar la cifra, si su valor es igual a 0 significa que se ha llegado al último octeto que forma parte del subidentificador.

En la Figura 2.10 se observa un ejemplo de la representación para el *OBJECT IDENTIFIER* 1.3.6.1.2.

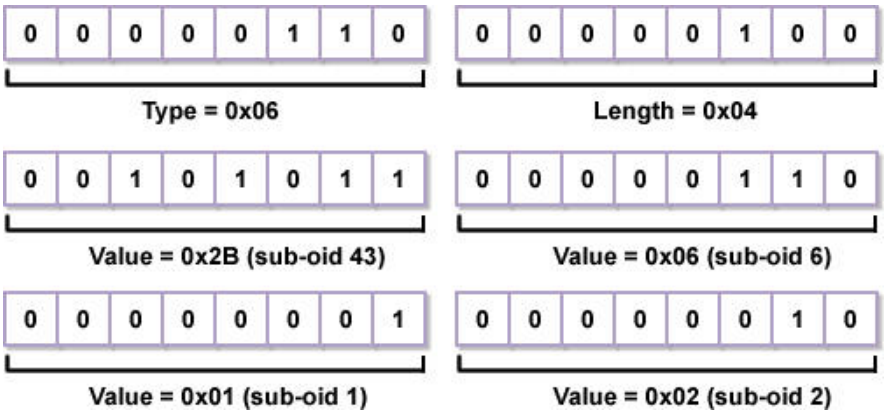

Figura 2.10: Ejemplo de la representación de un *OBJET IDENTIFIER*.

- ***SEQUENCE:*** La codificación para este valor se realiza de manera recursiva, cada componente es codificado como una tripleta TLV. El orden de las TLVs, será el mismo orden en el que se encuentra definida la estructura de datos.

- ***IpAddress:*** Se representa mediante el uso de 4 octetos en el campo *Value*. En la Figura 2.11 se tiene un ejemplo de la codificación para la Dirección IP 128.150.161.8.

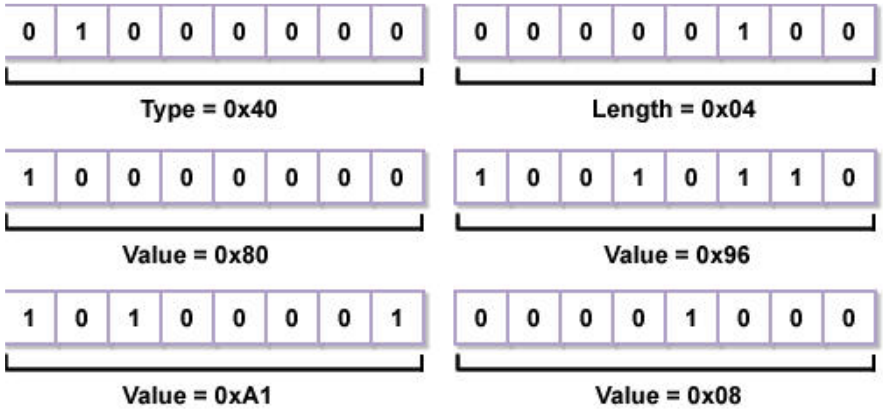

Figura 2.11: Ejemplo de la representación de un *IpAddress*.

- ***Counter:*** Permite representar un entero positivo. En la Figura 2.12 se puede observar como se realiza su codificación para el valor 203**.**

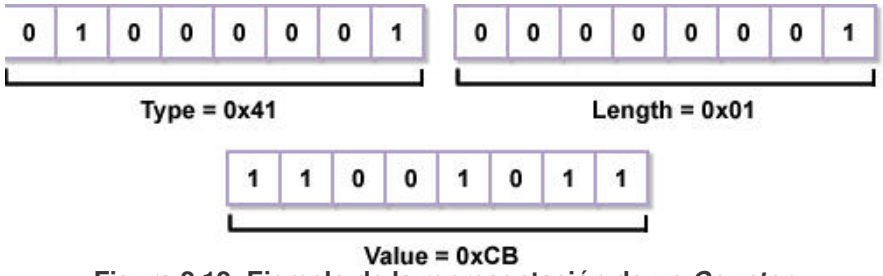

Figura 2.12: Ejemplo de la representación de un *Counter*.

- ***Gauge*:** Al igual que el *Counter*, representa un entero positivo. En la Figura 2.13 se puede apreciar la codificación para el valor 32.

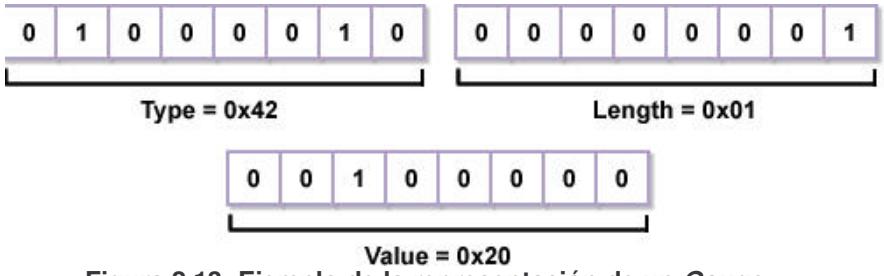

Figura 2.13: Ejemplo de la representación de un *Gauge*.

- ***TimeTicks*:** Por medio de este valor es posible representar un contador de tiempo, a través de un entero positivo. En la Figura 2.14 se observa un ejemplo su codificación para el valor 4023.

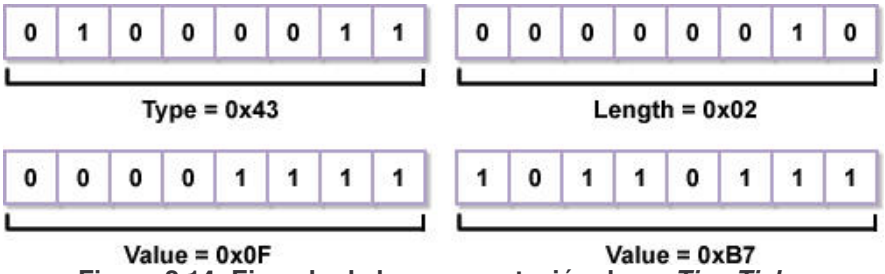

Figura 2.14: Ejemplo de la representación de un *TimeTicks*.

2.4 MIB (Management Information Base)

La MIB (*Management Information Base*), definida por la IETF (*Internet Engineering Task Force*), es una base de datos que contiene información relativa a la administración de una red de comunicaciones. De esta forma, describe la información que puede ser obtenida y/o modificada a través de un protocolo de gestión de red.

Esta base de datos está conformada por diferentes variables SNMP, que se encuentran definidas en un idioma estándar, independiente del sistema destino. Gracias a esta estandarización es posible controlar y administrar redes heterogéneas, cuyos componentes o dispositivos pertenezcan a diferentes fabricantes. Sin embargo, los fabricantes pueden definir y diseñar MIBs privadas con el objetivo de permitir la gestión de características específicas que no han sido incluidas en la MIB estándar.

La versión inicial de la MIB para su uso en protocolos de gestión de red se encuentra definida en el RFC 1156 [7]. Sin embargo, en la actualidad la versión soportada por la mayoría de los equipos SNMP corresponde a la MIB-II definida en el RFC 1213 [8].

2.4.1 Estructura de la MIB

La MIB posee una estructura jerárquica donde los datos se organizan conceptualmente en forma de árbol. En la parte superior de este árbol se encuentra disponible la información más general acerca de la red, mientras que, cada una de las ramas siguientes contiene detalles particulares sobre los dispositivos y servicios.

Por consiguiente, los niveles superiores de la MIB son fijos y se encuentran definidos según el RFC 1155 y el RFC 1213. No obstante, existen subárboles específicos que han sido definidos por la IETF y otras organizaciones.

De acuerdo al estándar, el nodo raíz no tiene etiqueta, pero posee por lo menos tres hijos directamente dependientes de él:

- Un nodo administrado por CCITT (*International Telegraph and Telephone Consultative Committee*), con la etiqueta ccitt(0).
- Un nodo administrado por ISO (*International Organization for Standardization*), cuya etiqueta es iso(1).
- Un nodo administrado conjuntamente por ISO y CCITT, identificado por la etiqueta joint-iso-ccitt(2) [8].

En la Figura 2.15 se observa un ejemplo de la estructura descrita anteriormente.

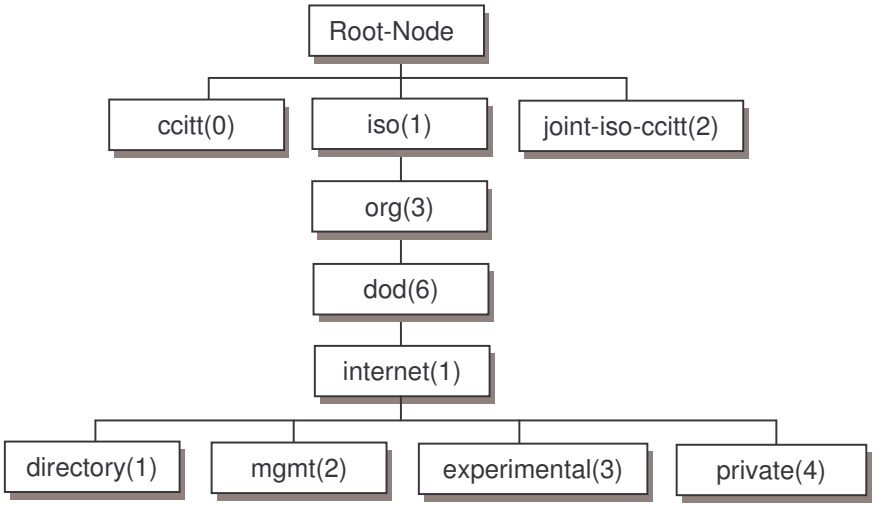

Figura 2.15: Subárbol de la MIB.

2.4.2 MIB versión I

La MIB I representa la versión inicial utilizada para la gestión de redes basadas en protocolos TCP/IP. En ella se encuentran definidas las variables necesarias para el monitoreo y control de los diversos componentes de Internet. Sin embargo, no todos los grupos de variables definidas son obligatorias para todos los dispositivos [7].

La lista de objetos de gestión definidos en esta versión se realizó incluyendo sólo los elementos esenciales. No obstante, este enfoque no es considerado restrictivo, debido a que se proporcionan mecanismos adicionales para extensión, tales como:

- Definición de nuevas versiones de la MIB.
- Definición de objetos privados.
- Definición de objetos no estandarizados.

Entre las descripciones incluidas en el RFC 1156 destacan las siguientes:

- **Objetos:** Los objetos almacenados en la MIB son definidos usando ASN.1. Cada tipo de objeto posee un nombre, una sintaxis y una codificación. El nombre se conoce como OID (*OBJECT IDENTIFIER*), y es utilizado para identificar los tipos de objetos que son gestionados.

  Por su parte, la sintaxis define la estructura de datos correspondiente a un tipo de objeto en particular, mientras que, la codificación se refiere a la forma en que el objeto está representado al momento de ser transmitido por la red.

- **Grupo de Objetos:** Los grupos de objetos son definidos con dos propósitos específicos:

  1. Proporcionar un mecanismo de asignación para los identificadores de objetos.
  2. Proporcionar un método para la implementación de agentes de gestión que conozcan cuales objetos se deben aplicar.

  Los objetos se encuentran organizados en los siguientes grupos: *System, Interfaces, Address Translation, IP, ICMP, TCP, UDP, EGP*.

- **Formato de Definiciones:** Los tipos de objetos son definidos empleando el conjunto de campos específicos que se describen en la Tabla 2.2.

Nombre	Descripción
Object	Está compuesto por un campo de texto denominado *OBJECT DESCRIPTOR* que representa el nombre de un tipo de objeto, y su OID correspondiente.
Syntax	Indica cual es la sintaxis del tipo de objeto correspondiente. Para ello, utiliza una instancia de ASN.1 tipo *ObjectSyntax*, definida en el SMI (*Structure Management Information*).
Definition	Proporciona una descripción semántica del tipo de objeto. Las implementaciones deben asegurarse de cumplir con esta definición.
Access	Este campo debe poseer uno de los siguientes valores posibles *read-only*, *read-write*, *write-only* o *not-accessible*.
Status	Contiene uno de los siguientes valores *mandatory*, *optional* u *obsolete*.

Tabla 2.2: Formato de definiciones de la MIB.

2.4.3 MIB versión II

La MIB II se encuentra definida en el RFC 1213 y actualmente representa la versión estándar incluida en todos los dispositivos que soportan al protocolo SNMP. Su objetivo principal consiste en proporcionar información general de gestión TCP/IP.

Entre las nuevas características añadidas a la versión destacan:

- Complementos para reflejar los nuevos requerimientos operacionales.
- Mejoras en el soporte de entidades multi-protocolos.
- Mejoras en la claridad y legibilidad.

Además, nuevos grupos de objetos se agregaron a esta versión. En la Figura 2.16 se encuentra un subárbol con los grupos definidos en la MIB II.

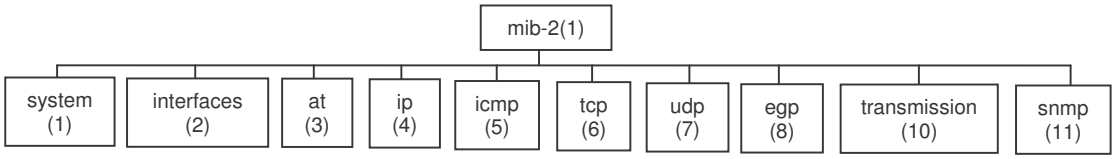

Figura 2.16: Subárbol de la MIB II.

En la Tabla 2.3 se observa una descripción de los grupos de objetos incluidos en la MIB II.

Nombre	OID	Descripción
system	1.3.6.1.2.1.1	Define un conjunto de objetos relacionados a la información del sistema.
interfaces	1.3.6.1.2.1.2	Permite realizar un monitoreo sobre el estado de cada interfaz perteneciente a una entidad gestionada.
at	1.3.6.1.2.1.3	Este grupo es considerado obsoleto; se proporciona sólo para mantener compatibilidad con versiones anteriores.
ip	1.3.6.1.2.1.4	Permite monitorear muchos de los aspectos referentes a IP, incluido el enrutamiento IP.
icmp	1.3.6.1.2.1.5	Proporciona información acerca de errores ICMP, paquetes descartados, etc.
tcp	1.3.6.1.2.1.6	Permite monitorear el estado de la conexión TCP (*closed, listen*, etc.), entre otros aspectos.
udp	1.3.6.1.2.1.7	Proporciona información para estadísticas UDP, número de datagramas recibidos, número de datagramas enviados, etc.
egp	1.3.6.1.2.1.8	Proporciona datos estadísticos relacionados al protocolo EGP (*Exterior Gateway Protocol*).
transmission	1.3.6.1.2.1.10	Actualmente no existen objetos definidos para este grupo.
snmp	1.3.6.1.2.1.11	Permite monitorear la ejecución de la aplicación SNMP en la entidad gestionada.

Tabla 2.3: Descripción de los grupos de la MIB II.

2.5 SMI (Structure of Management Information)

SMI es un estándar que describe la estructura común para definir una MIB. Entre sus especificaciones se encuentran:

- Las operaciones y tipo de datos permitidos.
- El modelo de organización y agrupación de la información de gestión.
- La representación y los nombres de los recursos dentro de la MIB.

Actualmente existen 2 versiones de este estándar: SMIv1 y SMIv2.

2.5.1 SMIv1

En la versión inicial de SMI se definen las estructuras comunes y los esquemas de identificación utilizados para representar los objetos a ser gestionados.

Entre las descripciones incluidas en el RFC 1155 [9] se encuentra las siguientes:

- **Nombres:** Como se mencionó anteriormente, los nombres se emplean para identificar los objetos administrados. A través de SMI es posible especificar nombres que son de naturaleza jerárquica, los cuales son denominados *OBJECT IDENTIFIER*.

  La estructura de árbol en la que se organizan los objetos gestionados es la base que hace posible el esquema de nombres. De esta manera, cada *OBJECT IDENTIFIER* está compuesto por una secuencia de enteros separados por puntos, cuyos valores corresponden a los nodos del árbol. Esta notación es la utilizada para representar los objetos internamente, dentro de los agentes SNMP.

  Sin embargo, cada objeto gestionado posee además un nombre asociado que puede ser empleado en sustitución de los números enteros.

  En la Figura 2.17 se puede observar un ejemplo del esquema de nombre utilizado para el subárbol Internet.

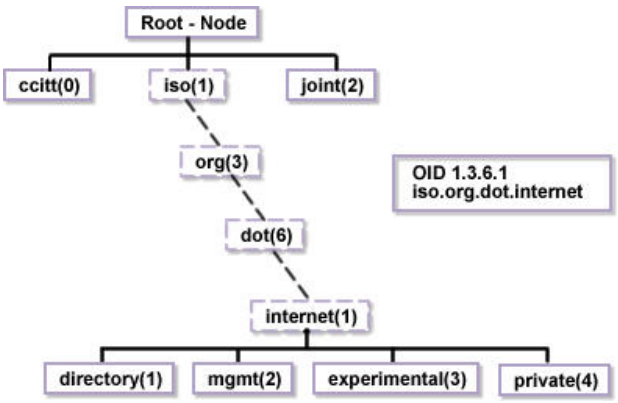

Figura 2.17: OID para subárbol internet.

SMIv1 determina las políticas bajo las cuales el subárbol anterior es administrado. Los nodos presentes en este subárbol se describen en la Tabla 2.4.

Nombre	Descripción
directory	Este subárbol se encuentra reservado para usos futuros.
mgmt	Es usado para identificar los objetos que son definidos en documentos aprobados por el IAB (*Internet Architecture Board*).
experimental	Este subárbol se reserva para objetos utilizados con fines de investigación y pruebas.
private	Los objetos pertenecientes a este subárbol son definidos por personas u organizaciones.

Tabla 2.4: Descripción de nodos pertenecientes al subárbol internet.

- **Tipos Definidos:** SMIv1 incluye la definición de nuevos tipos de datos necesarios para el protocolo SNMP, entre los que se encuentran:

  - *NetworkAddress*.
  - *IpAddress*.
  - *Counter*:
  - *Gauge*.
  - *TimeTicks*.
  - *Opaque*.

2.5.2 SMIv2

SMIv2 añade nuevas funcionalidades para dar soporte al protocolo SNMPv2. Entre ellas destaca la adición de nuevos objetos al subárbol Internet, la definición de nuevos tipos de datos, entre otros.

En la Tabla 2.5 se describen los nuevos tipos de datos definidos para SMIv2, en el RFC 2578 [10].

Tipo de Dato	Descripción
Integer32	Igual al tipo de dato *INTEGER*
Counter32	Igual al tipo de dato *Counter*
Gauge32	Igual al tipo de dato *Gauge*
Unsigned32	Representa valores decimales comprendidos entre 0 y $2^{32} - 1$
Counter64	Similar a *Counter32* pero en este caso el máximo valor es $2^{64} - 1$
BITS	Permite enumerar enteros no negativos llamados *bits*

Tabla 2.5: Nuevos tipos de datos en SMIv2.

Además, a través de SMIv2 se agregan nuevos campos opcionales al formato para la definición de los objetos a administrar. Esto con el fin de

proporcionar una mejor descripción de los objetos y un mayor control en el acceso a los mismos.

Los nuevos campos añadidos a la definición de objetos son:

- ***UnitParts***: Es la descripción textual de las unidades utilizadas para representar el objeto.

- ***MAX-ACCESS***: Se refiere al acceso permitido para un tipo de objeto en particular. Sus valores posibles son *read-only, read-write, read-create, not-accessible*, *accessible-for-notify*.

- ***STATUS***: Este campo ha sido extendido para permitir la incorporación de nuevos valores, tales como *current, obsolete, deprecated*.

- ***AUGMENTS***: Este campo permite extender una tabla mediante la adición de una o más columnas, representada por algún otro objeto.

2.6 Especificación del Protocolo SNMP

SNMP es un protocolo de la capa de aplicación del modelo OSI donde las variables de un agente SNMP pueden ser consultadas o modificadas [5].

La comunicación entre los NMSs y Agentes viene dada por el intercambio de mensajes, siendo cada uno de estos independiente y representado por un datagrama UDP siguiendo las reglas de codificación ASN.1.

Un mensaje contiene los campos siguientes:

- **Version**: Describe la versión del protocolo que se está utilizando.

- **Community**: Es la contraseña utilizada mediante la cual la entidad protocolo se autentica para obtener derechos y/o privilegios en la comunidad a la que pertenece.

- **Data**: Son los datos que se envían entre las entidades del protocolo, bien sea para hacer una solicitud, o para enviar una respuesta o notificación de un evento.

En la Figura 2.18 se puede observar la representación gráfica de un mensaje SNMP.

Figura 2.18: Representación gráfica de un mensaje SNMP.

Los mensajes son recibidos por medio del puerto UDP 161 y los Traps son recibidos por el puerto UDP 162. La implementación de esta versión del protocolo sólo acepta mensajes que no excedan de los 484 octetos.

Un mensaje SNMP se representa a través de una PDU. Cada PDU es definida utilizando ASN.1, como se observa en la Figura 2.19.

```
PDUs::= CHOICE { get-request GetRequest-PDU
                 get-next-request GetNextRequest-PDU
                 get-response GetResponse-PDU
                 set-request SetRequest-PDU
                 trap Trap-PDU }
```

Figura 2.19: Definición de la PDU.

2.6.1 Procedimiento que se realiza al ejecutar operaciones SNMP

Cuando se genera un mensaje por parte de alguna entidad protocolo, se construye la PDU como un objeto ASN.1 para representar la operación SNMP. Este objeto en adición con el *Community Name* y la dirección origen y destino es pasado al servicio que implementará el *Authentication Scheme*, retornando otro objeto ASN.1. Así, la entidad protocolo finalmente construye el mensaje con el objeto retornado junto con el *Community Name*. El resultado se codifica usando BER y se envía usando un servicio de transporte que se encuentra en la entidad protocolo [11].

Una vez que el mensaje es recibido en la entidad protocolo a la cual fue enviado, se realiza un análisis del datagrama a fin de construir un objeto ASN.1. En caso de que el análisis falle, el datagrama es descartado y no se realizan más acciones, de lo contrario se verifica la versión del mensaje SNMP para constatar que ambas entidades estén usando la misma versión del protocolo. Si la versión no es la misma para ambos, entonces el mensaje es descartado, si no se toma el *Community Name* junto con los datos provenientes en el mensaje, más la dirección origen y destino para pasarlos al servicio *Authentication Scheme*, el cual puede retornar un objeto ASN.1 o una señal debido a que ha fallado la autenticación, generando un Trap y descartando el datagrama.

Si el objeto ASN.1 es retornado, se realiza un análisis del mismo para construir un objeto ASN.1 que corresponda a la PDU que fue recibida, si el análisis falla es descartado y no se realizan más acciones. En caso contrario, se toma el *Community Name* y se procede a seleccionar el perfil apropiado para posteriormente procesar la PDU, por medio del cual se retorna el mensaje en respuesta a la solicitud que fue realizada.

2.6.2 Definición y especificación de las primitivas SNMP

Para que las PDUs que representan los mensajes SNMP puedan ser descritas, se definen los siguientes tipos de datos codificados en ASN.1:

- ***RequestID***: Es un entero que sirve para relacionar los *Response* entrantes con los *Request* que están pendientes. Además, es utilizado para identificar los mensajes que estén duplicados en la red. La definición de este tipo de dato se puede observar en la Figura 2.20.

```
RequestID::=INTEGER
```

Figura 2.20 Definición del tipo *RequestID.*

- ***ErrorStatus***: Es un entero que indica que una excepción ha ocurrido cuando se procesa un *Request*. La definición de este tipo de dato se puede observar en la Figura 2.21.

```
ErrorStatus::= INTEGER {
noError(0),tooBig(1),noSuchName(2),
badValue(3),readOnly(4),genErr(50)}
```

Figura 2.21 Definición del tipo *ErrorStatus.*

- ***ErrorIndex***: Es un entero utilizado para indicar cual variable ha causado el error. La definición de este tipo de dato se puede observar en la Figura 2.22.

```
ErrorIndex::=INTEGER
```

Figura 2.22: Definición del tipo *ErrorIndex*.

- ***VarBind***: Es un tipo de datos conformado por una tupla que contiene el nombre de la variable y su valor. La definición de este tipo de dato se puede observar en la Figura 2.23.

```
VarBind::= SEQUENCE {
name ObjectName, value ObjectSyntax  }
```

Figura 2.23: Definición del tipo *VarBind*.

- ***VarBindList***: Es una secuencia que contiene elementos del tipo *VarBind*, es decir, nombres de variables con sus valores correspondientes. La definición de este tipo de dato se puede observar en la Figura 2.24.

```
VarBindList::= SEQUENCE OF { VarBind }
```

Figura 2.24: Definición del tipo VarBindList.

Una vez descritos los tipos de datos usados en las PDUs, es posible hacer la descripción y especificación de cada una de éstas en detalle.

- ***GetRequest***: Esta primitiva SNMP es generada por un NMS cuando desea hacer una solicitud del valor de una variable a un agente.

- ***GetNextRequest***: Es generado por un NMS cuando requiere saber el valor de la próxima variable de un agente.

- ***GetResponse***: Es generada por el agente en respuesta ante alguna solicitud hecha por un NMS, tal como *GetRequest, GetNextRequest* o un *SetRequest*.

- ***SetRequest***: Es generado por un NMS para asignar valor a una variable presente en la MIB del agente.

- ***Trap***: Es generada desde el agente para notificar al NMS algún evento que haya ocurrido. En la Tabla 2.6 se puede apreciar una descripción de los traps genéricos definidos.

Traps Genéricos	
coldStart	La entidad protocolo se ha reiniciado indicando que tanto el agente como el NMS pueden ser alterados.
warmStart	La entidad protocolo ha sido reinicializada, pero ni el agente ni el NMS han sido alterados.
linkDowm	La entidad protocolo reconoce una falla en uno de los enlaces de comunicación del agente.
linkUp	La entidad protocolo reconoce que uno de los enlaces de comunicación presentes en el agente ha iniciado su servicio.
authenticationFailure	Significa que el mensaje que ha sido enviado no puede ser autenticado, ya que el nombre de la comunidad es incorrecto.
egpNeigborLoss	Indica que un vecino EGP no puede ser encontrado debido a que se encuentra inoperativo.
enterpriseSpecific	El agente reconoce que algún evento específico ha ocurrido.

Tabla 2.6: Traps genéricos.

2.6.3 Reglas aplicadas para generar respuestas a una solicitud

Al momento de generar alguna respuesta ante una solicitud, primero se deben cumplir con ciertas reglas.

Para las solicitudes de tipo *GetRequest* y *GetNextRequest* las reglas son:

- Si el nombre de alguna variable no coincide con el nombre de la variable en la vista SNMP, se envía como respuesta el mensaje recibido con la diferencia que *ErrorStatus* es "*noSuchName*" y *ErrorIndex* es inicializado con el índice de la primera variable que presenta el problema.

- Si la variable referenciada es de tipo "*Aggregate*", se responde el mismo mensaje con el *ErrorStatus* en "*noSuchName*".

- Si el tamaño del *GetResponse* generado excede el límite, el mensaje es devuelto igual como se recibió, con el *ErrorStatus* "*tooBig*".

- Si el valor de la variable no pudiese ser recuperado por alguna causa, se devuelve el mensaje con *ErrorStatus* en "*genErr*".

- Si por el contrario, ninguna de las cuatro situaciones anteriores ocurre, se responde ante la solicitud con el valor de la variable y *ErrorStatus* en "*noError*".

Para la solicitud de tipo SetRequest las siguientes reglas son aplicadas:

- Si el objeto no está disponible para la operación set en la vista de la MIB, el mensaje es devuelto de igual forma como se recibió, excepto que el valor de *ErrorStatus* es "*noSuchName*".

- Si el valor del objeto no concuerda con el lenguaje ASN.1 entonces se devuelve el mensaje recibido de igual manera, a excepción del valor de *ErrorStatus* es "*badValue*".

- Si el tamaño del *GetResponse* generado excede el límite, el mensaje es devuelto igual como se recibió, con el *ErrorStatus* "*tooBig*".

- Si el valor de la variable no pudiese ser modificado por alguna razón, se devuelve el mensaje con *ErrorStatus* en "*genErr*".

- Si por el contrario, ninguna de las cuatro situaciones anteriores ocurre, el valor es asignado a la variable, y se responde con el mismo mensaje excepto que el *ErrorStatus* es "*noError*".

2.7 SNMPv2

En marzo del año 1992, ante varias solicitudes de mejoras para la versión 1 del protocolo SNMP, la IETF inició investigaciones y esfuerzos para mejorar la seguridad en SNMP; sin embargo, el primer diseño de SNMPv2 no tuvo éxito ni una gran aceptación comercial [5]. Por otra parte, el grupo de trabajo de SNMPv2 se reunió nuevamente a finales de 1994, esta vez con la finalidad de plantear nuevas mejoras y propuestas a esta versión inicial del protocolo, sin embargo no llegaron a un consenso y se produjeron las siguientes acciones y/o mejoras:

- Modificaciones al modelo administrativo y de seguridad llamado SNMP versión 2 basado en las comunidades (SNMPv2c).
- Actividades sin finalizar con respecto a elementos como: la seguridad, el marco administrativo, la configuración remota de la MIB y la comunicación entre NMS.
- Nuevos tipos de datos, macros y convenciones textuales.
- Nuevas operaciones y/o primitivas, que facilitan transferencias de grandes cantidades de datos, enriquecimiento de los códigos de errores, soporte multiprotocolo a nivel de transporte.

Para realizar mejoras en relación al conjunto de operaciones del protocolo, fue necesario definir nuevas PDUs, entre estas se encuentran, *get-bulk-request*, *inform-request*, *snmpv2-trap* y *report*.

Las PDUs generadas para esta versión del protocolo, al igual que la versión descrita anteriormente, se conforman mediante un *CHOICE* entre los tipos de PDUs existentes, a diferencia que se incorporan las nuevas. Esto puede ser observado en la Figura 2.25.

```
PDU::= CHOICE { get-request GetRequest-PDU
                get-next-request GetNextRequest-PDU
                get-bulk-request GetBulkRequest-PDU
                response Response-PDU
                set-request SetRequest-PDU
                inform-request InformRequest-PDU
                snmpv2-trap SNMPv2-Trap-PDU
                report Report-PDU }
```

Figura 2.25: Definición de PDUs para SNMPv2.

2.7.1 Procesamiento de las PDUs para SNMPv2c

Si al momento de realizar una solicitud ante cualquier entidad protocolo los campos de la PDU no son referenciados por la solicitud, los mismos son

ignorados; sin embargo, todos los componentes o campos de la PDU deben estar correctamente codificados en ASN.1, sean o no ignorados [12].

Las PDUs para esta versión del protocolo son definidas como sigue:

- ***GetRequest-PDU***: Todos los campos del *response*-PDU tienen el mismo valor que reciben en el *request* a excepción de las siguientes condiciones:

  - Si el nombre de la variable concuerda con la variable que fue solicitada por el *request*, entonces se le asigna el valor correspondiente a la variable.
  - Si la variable no tiene un prefijo *OBJECT IDENTIFIER* que coincida con el *OBJECT IDENTIFIER* de la solicitud, entonces el valor del campo es asignado como "*noSuchObject*". En otro caso el valor del campo se asigna como "*noSuchInstance*".
  - Si el procesamiento de alguna variable falla, entonces el Response-PDU es reformateado con el valor de *ErrorStatus* en "*genErr*". En otro caso el valor de *ErrorStatus* en *Response-PDU* es "*noErr*".

- ***GetNextRequest-PDU***: Al igual que para el *GetRequest-PDU*, al momento de generar el *Response-PDU*, cada variable es procesada de la siguiente manera:

  - Si la variable en el orden lexicográfico es la primera en el sucesor de la solicitud que se ha realizado, entonces el valor de la variable es asignado en el *Response-PDU*.
  - Si la variable solicitada no precede ninguna variable, entonces el valor del *Response-PDU* es asignado como "*endOfMibView*".
  - Si el procesamiento de alguna variable falla por alguna razón, entonces el valor de *ErrorStatus* es "*genErr*". En otro caso el valor de *ErrorStatus* en el *Response-PDU* es "*noErr*".

- ***GetBulkRequest-PDU***: El propósito de esta PDU es transmitir grandes cantidades de datos, así como grandes tablas de manera rápida y eficiente. Si el procesamiento de las solicitudes es exitoso generará entonces cero o más valores de variables en el *Response-PDU*.

- ***Response-PDU***: Se genera en respuesta a solicitudes como, *GetRequest-PDU, GetNextRequest-PDU, GetBulkRequets-PDU, SetRequest-PDU, InformRequest-PDU*.

- ***SetRequest-PDU***: Modifica el valor de la variable especificada en el *Request-PDU*, esta asignación se realiza en 2 fases. En la primera fase se

verifica que la sintaxis y codificación de la variable sea correcta. En la segunda fase se modifica el valor de la variable.

- ***SNMPv2-Trap-PDU***: Esta PDU es generada y transmitida por un agente para notificar que algún evento ha ocurrido o que una condición en el mismo ha sido activada.

- ***InformRequest-PDU***: Es generada y transmitida desde un NMS a otro, para obtener información acerca de la vista MIB, así como de un evento que haya ocurrido.

2.8 Lenguaje de Programación PHP

El lenguaje PHP[1] tuvo su inicio en el año 1994 gracias a Rasmus Lerdorf, quién lo creó para controlar el número de personas que visitaban su curriculum vitae en la web. Estas primeras versiones no fueron distribuidas al público [13].

En los meses siguientes a su creación, un grupo de programadores comenzaron a prestar su colaboración en la verificación del código y sus revisiones. Además, Lerdorf desarrolló funciones en lenguaje C para añadir nuevas operaciones, tales como conexión a bases de datos. De esta manera se hace disponible a principios del año 1995 la primera versión para el público, conocida como *Personal Home Pages Tools*. Esta primera versión estaba compuesta por un analizador sintáctico muy simple, y un conjunto de utilidades comunes para las páginas web de la época (libro de visitas, contador, entre otras).

Posteriormente, a mediados del año 1995 aparece una nueva versión pública denominada PHP/FI 2.0. Para esta versión el analizador sintáctico fue reescrito, y se agregaron herramientas para el procesamiento de datos desde un formulario, además de soporte para conexiones con un gestor de bases de datos conocido como mSQL (*Mini Structured Query Language).*

En 1997, Zeev Suraski en compañía de Andi Gutmans, deciden crear una nueva versión de PHP/FI [13]. De este modo, el desarrollo del lenguaje dejó de ser un proyecto personal, y se convirtió en un proyecto de grupo con mayor organización que trae como consecuencia el nacimiento de una nueva versión llamada PHP 3.0, que proporciona varias novedades tales como

[1] http://www.php.net

protocolos, una API ampliada, y el soporte de conectividad con varios gestores de bases de datos. Esta versión apareció oficialmente en junio de 1998.

A continuación, en el año 1999 se realiza la primera versión del motor *Zend*, con el propósito de aportar modularidad, claridad, y herramientas de optimización. Este motor es el responsable de analizar el código PHP, y definir la sintaxis del lenguaje de programación. Es por ello que es incorporado en una nueva versión denominada PHP 4.0, la cual se hizo pública en mayo del año 2000 [13].

Por último se creó el motor *Zend* 2.0, trayendo como consecuencia el desarrollo de la versión más reciente del lenguaje conocida como PHP 5.0. Esta versión incorpora un soporte para la programación orientada a objetos, debido a que en las versiones anteriores no era posible la encapsulación. Además, añade mejoras para el manejo de errores.

En la actualidad se estima que PHP, conocido como *Hypertext Pre-processor*, es utilizado por 1,224,183 servidores alrededor del mundo[2]. Esta cifra evidencia el éxito que ha tenido el lenguaje debido a la combinación que brinda entre sencillez y potencia, además de otras características que lo convierten en uno de los lenguajes de mayor popularidad y aceptación.

2.8.1 Características de PHP

El lenguaje PHP proporciona al programador las herramientas necesarias para realizar el trabajo de una manera rápida y eficiente. Entre las numerosas características que han contribuido a la popularidad y éxito del lenguaje, destacan las siguientes:

- **Familiaridad**: La sintaxis del lenguaje PHP es similar a la de otros lenguajes tales como C y Perl, es por ello que los programadores se adaptan fácilmente y con rapidez a su uso.

- **Simplicidad**: La sencillez del lenguaje permite que los programadores puedan generar código en el menor tiempo posible.

- **Eficiencia**: En un ambiente multiusuario, tal como el web, la eficiencia representa una característica muy importante. Por este motivo, PHP incorpora su propio sistema de administración de memoria, además de

[2] http://www.netcraft.com/survey

características para el manejo de sesiones y eliminación de asignaciones de memorias innecesarias, con el fin de conformar un sistema robusto, estable, y de mayor eficiencia.

- **Seguridad**: El lenguaje PHP proporciona a los desarrolladores y a los administradores un conjunto eficiente de garantías de seguridad. Estas garantías pueden ser divididas en dos grupos: nivel de sistema y nivel de aplicaciones.

- **Flexibilidad**: Debido a que PHP es un lenguaje embebido, es extremadamente flexible en el cumplimiento de las necesidades de los desarrolladores. Aunque la mayoría de las veces el lenguaje es utilizado en conjunción con HTML, puede ser combinado con lenguajes tales como JavaScript, WML, XML, entre otros.

  El lenguaje PHP es totalmente independiente del navegador, ya que los scripts se compilan completamente del lado del servidor antes de ser enviados al usuario. De hecho, los scripts PHP pueden ser enviados a cualquier tipo de dispositivo que contenga un navegador, como por ejemplo teléfonos móviles, PDA (Personal Digital Assistant), entre otros. Además, los desarrolladores que requieran crear aplicaciones basadas en Shell también podrán ejecutar PHP desde la línea de comandos.

  Cabe destacar que PHP es en gran medida independiente de la plataforma, debido a que actualmente puede ser ejecutado en más de 25 plataformas diferentes, tales como Unix (Solaris, AIX, HP-UX, FreeBSD, Linux, etc.), Windows (95, 98, NT, etc.) entre otras [13].

- **Velocidad**: PHP está escrito totalmente en lenguaje C, por lo que se ejecuta rápidamente utilizando poca memoria. Además, es posible utilizarlo como un módulo de Apache proporcionándole mayor rapidez.

- **Open Source**: El lenguaje PHP es gratuito y de código abierto, por lo que usuarios de todo el mundo han contribuido en el avance de los proyectos.

2.8.2 PEAR (PHP Extension and Application Repository)

PEAR[3] es un repositorio de extensiones y aplicaciones para PHP [14]. Además, funciona como un sistema de distribución para componentes PHP

[3] http://pear.php.net

reutilizables de alta calidad, disponibles en forma de paquetes. Su principal objetivo consiste en proporcionar:

- Una biblioteca de código *Open Source* para los usuarios de PHP.
- Un sistema de distribución de código y mantenimiento de paquetes.
- Un estilo de codificación estándar.

La distribución oficial de PEAR posee ciertos estándares de codificación, que definen básicamente el modo en que un script PHP debe ser realizado. Estas normas tienen como objetivo mantener el código compatible, para que de esta forma sea fácilmente legible tanto para los desarrolladores del proyecto PEAR, como para los programadores en lenguaje PHP.

3. Marco Metodológico

Para llevar a cabo este trabajo especial de grado, se seleccionó una metodología basada en la velocidad y eficiencia del desarrollo que se adapta a la naturaleza del proyecto, permitiendo así cumplir exitosamente con cada uno de los objetivos propuestos.

A continuación se describen los aspectos más importantes de la metodología utilizada.

3.1 Metodología de Desarrollo

Las metodologías ágiles son estrategias de desarrollo de software que están enfocadas hacia una planificación adaptativa en lugar de predictiva, con el fin de aumentar las posibilidades de éxito en un proyecto. A diferencia de las metodologías tradicionales, los métodos ágiles no emplean una planificación estricta sino abierta y flexible para aumentar la habilidad de respuesta a los diversos cambios que puedan surgir a lo largo del proyecto.

Entre las metodologías ágiles más conocidas se encuentra la Programación Extrema (*Extreme Programming* o XP). Este método se basa en un proceso iterativo e incremental, con el propósito de disminuir la complejidad del desarrollo y aumentar la productividad [15].

Las principales características de la Programación Extrema son las siguientes:

- Se basa en la simplicidad, comunicación, reutilización y mejora continua del código.
- Está dirigido a grupos pequeños de programadores.
- Promueve el trabajo en equipo.
- Comprende el uso de un conjunto de prácticas de programación, con el fin de disminuir costos asociados con cambios en los requerimientos durante el desarrollo del software.
- Evalúa cuatro variables que afectan el resultado final, las cuales son el costo, el tiempo, la calidad y el alcance del proyecto.

3.1.1 Fases de la Programación Extrema

La Programación Extrema divide el proceso de desarrollo en cuatro fases: Planificación, Diseño, Implementación y Pruebas.

- **Planificación:** La etapa de planificación comprende una serie de actividades que permiten identificar el alcance y los requerimientos de la aplicación. Como primer paso se redactan las historias de usuario, las cuales se utilizan para capturar las funcionalidades que se desean desarrollar. Cada historia de usuario comprende una descripción corta y escrita en lenguaje natural de las necesidades del sistema, sin reflejar los detalles.

  El siguiente paso consiste en realizar el plan de entregas. Para ello, los desarrolladores deben hacer una estimación de cuánto tiempo se necesitará para implementar cada historia de usuario, tomando en cuenta dos parámetros: el tiempo de desarrollo ideal, y cuáles son los requerimientos más importantes o de mayor prioridad para el cliente.

  Una vez finalizado el plan de entregas, se elabora el plan de iteraciones. Cada iteración agrupa una serie de historias de usuario que deben ser desarrolladas en un periodo comprendido entre una y tres semanas. La selección de este grupo de historias es realizada por el cliente al comienzo de cada entrega, basándose en el plan de entregas, e incluyendo además aquellas historias de usuario que no pasaron la prueba de aceptación en la iteración anterior.

  Adicionalmente, durante la fase de planificación el equipo de desarrollo selecciona y se familiariza con las herramientas, tecnologías y prácticas que se utilizarán a lo largo del proyecto.

- **Diseño:** Esta fase persigue como meta la simplicidad, debido a que es menor el tiempo empleado en implementar un diseño simple que uno complejo. Además en un diseño simple cualquier cambio o modificación será mucho más sencillo y rápido de realizar.

  Durante la etapa de diseño se elaboran las tarjetas CRC (*Class, Responsibilities and Collaboration*). Cada tarjeta representa una clase junto con sus responsabilidades, y referencia a aquellas clases con las que trabaja en conjunto para llevar a cabo sus tareas. La principal ventaja de esta herramienta es que permite al desarrollador desprenderse del trabajo basado en procedimientos y trabajar con una metodología basada en objetos.

  En la Figura 3.1 se puede observar la estructura de las tarjetas CRC.

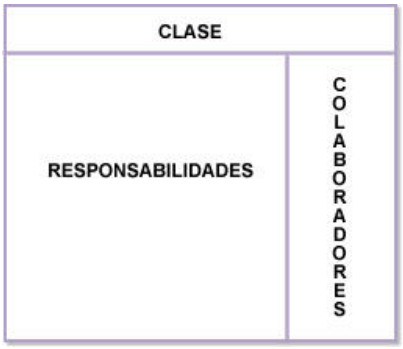

Figura 3.1: Estructura de Tarjeta CRC.

Por último se recomienda que en la etapa de diseño se practique siempre que sea posible la refactorización. De esta manera se eliminan redundancias, funcionalidades inútiles, se ahorra tiempo y se incrementa la calidad del proyecto en desarrollo.

- **Implementación:** Esta fase incluye varias iteraciones sobre el sistema antes de ser entregado. Luego de seleccionar las historias de usuario a implementar en la iteración actual, se pide al cliente que especifique detalladamente lo que debe ser implementado.

  Durante esta etapa el código debe ser desarrollado utilizando estándares de codificación, con el propósito de facilitar la lectura y modificación por cualquier miembro del equipo de desarrollo. Además es necesario que se haga una integración continua del código, evitando y detectando problemas de compatibilidad lo más pronto posible.

- **Pruebas:** Las pruebas unitarias constituyen uno de los aspectos más importantes de la Programación Extrema, por ello es indispensable crear o descargar un *framework* que permita efectuar pruebas unitarias automatizadas a lo largo del desarrollo.

  Se recomienda que todas las clases del sistema sean evaluadas a través de las pruebas unitarias, omitiendo los métodos triviales. Después de cada modificación se deben emplear nuevamente las pruebas, para verificar que cambios en la estructura no ocasionaron cambios en las funcionalidades. Sin embargo, si se añaden nuevos métodos al código las pruebas unitarias deberán ser rediseñadas para adaptarse a las nuevas funcionalidades.

  Asimismo, durante el desarrollo del proyecto se deben aplicar pruebas de aceptación también conocidas como pruebas funcionales, las cuales permiten garantizar que los requerimientos exigidos han sido alcanzados y que el sistema es aceptable.

La fase de pruebas incrementa la calidad del producto final, lo cual representa una característica esencial del proceso de Programación Extrema.

3.1.2 Prácticas de la Programación Extrema

El proceso descrito anteriormente se fundamenta en las siguientes prácticas con el propósito de obtener resultados de alta calidad en un tiempo razonable:

- **Planificación del proyecto:** Debe utilizarse una planificación incremental y flexible, que varíe en función de los cambios en los requerimientos y se adapte fácilmente a estos.

- **Pruebas del sistema:** A lo largo del desarrollo deben realizarse constantemente pruebas unitarias y pruebas de aceptación al proyecto.

- **Programación en parejas:** Se recomienda que los desarrolladores trabajen en pareja, de esta manera se garantiza que el código siempre está siendo revisado por otra persona. Con esta práctica se ha demostrado que dos programadores producen código fuente de mayor calidad que trabajando por separado.

- **Refactorización:** Permite mantener el código en buen estado, modificándolo para que conserve claridad y sencillez.

- **Diseño simple:** El diseño debe ser lo más sencillo posible, evitando añadir funcionalidades que no formen parte de los requerimientos, con el fin de acelerar el proceso de desarrollo.

- **Propiedad colectiva del código:** Todos los desarrolladores pueden realizar cambios sobre el código fuente y deben estar familiarizados con los diversos módulos del sistema.

- **Integración continua:** Establece que la integración de código debe realizarse frecuentemente, evitando así que se acumulen errores de incompatibilidad.

- **Cliente en el equipo:** Esta práctica indica que el cliente debe incluirse en el proceso de planificación y desarrollo del proyecto, verificando que las funcionalidades son correctas y van acorde con sus requerimientos.

- **Entregas pequeñas:** Recomienda la entrega de nuevas versiones del proyecto frecuentemente, para que el cliente evalúe y retroalimente el proceso de desarrollo.

- **Cuarenta horas semanales:** Se sugiere que cada programador dedique al proyecto ocho horas diarias por cinco días a la semana, con el propósito de obtener un mejor rendimiento que se refleje en la calidad del producto final.

- **Estándares de código:** Se deben establecer estándares de código para facilitar la legibilidad y manipulación del proyecto por parte de los desarrolladores.

- **Uso de metáforas:** Es necesario definir una metáfora que describa claramente el funcionamiento y propósito del proyecto.

4. Marco Aplicativo

En este capítulo se describe el proceso que se llevó a cabo a lo largo de la elaboración de la Biblioteca de funciones SNMP, y las interfaces web para la prueba de su correcto funcionamiento. Este proceso se basa en una adaptación de la metodología de desarrollo ágil conocida como Programación Extrema, la cual se explicó en detalle en el capítulo 3.

4.1 Fase de Planificación

Como primer paso durante la etapa de planificación, se determinan los requerimientos a los que se le dará repuesta. Para ello se emplean las historias de usuario, que representan la técnica utilizada en Programación Extrema para describir las funcionalidades que se pretenden implementar a través del desarrollo de la aplicación.

4.1.1 Historias de usuario

Las historias de usuario se dividieron en 3 grupos: Biblioteca de funciones SNMP, NMSWeb (*Network Management System Web*), y RMSWeb (*Router Management System Web*).

- **Biblioteca de funciones SNMP:** En la Tabla 4.1 se describen los requerimientos que deben cumplirse.

Nombre de la Historia de Usuario	Descripción
Enviar GetRequest	Envía una solicitud GetRequest permitiendo consultar el valor de uno o más objetos, a través del protocolo SNMP en sus versiones v1 o v2c.
Enviar SetRequest	Envía una solicitud SetRequest para modificar el valor de uno o más objetos, a través del protocolo SNMP v1 o v2c.
Enviar GetNextRequest	Envía una solicitud GetNextRequest consultando el valor de uno o más objetos, a través del protocolo SNMP v1 o v2c.
Enviar GetBulkRequest	Envía una solicitud para recuperar el valor de varios objetos, utilizando el protocolo SNMPv2c.
Enviar Trap	Envía una notificación Trap, a través del protocolo SNMPv1.
Enviar InformRequest	Envía una notificación InformRequest.
Enviar Trap SNMPv2c	Envía una notificación Trap, utilizando el protocolo SNMPv2c.
Ejecutar Walk	Permite consultar un subárbol de la MIB, a través de solicitudes GetNextRequest.
Registrar eventos	Almacena en un archivo los eventos más importantes ocurridos durante el uso de las funciones de la biblioteca.
Validar datos	Valida que todos los datos sean correctos, y utiliza excepciones en caso de que ocurra algún error.

Tabla 4.1: Historias de usuario para la Biblioteca de funciones SNMP.

- **NMSWeb:** Comprende una interfaz realizada haciendo uso de la Biblioteca de funciones SNMP. En la Tabla 4.2 se mencionan los requerimientos establecidos para este grupo.

Nombre de la Historia de Usuario	Descripción
Solicitudes SNMP	Proporciona una interfaz para el envío de solicitudes SNMPv1/v2c y la recuperación de los valores consultados.
Envío de Notificaciones	Proporciona una interfaz para el envío de notificaciones: Trap, InformRequest, y Trapv2.
Cambio de Idioma	Permite cambiar el idioma de español a inglés o viceversa.

Tabla 4.2: Historias de usuario para NMSWeb.

- **RMSWeb:** Es una interfaz orientada a la administración de dispositivos *router Cisco Systems* específicamente para el modelo 2811, a través de la Biblioteca de funciones SNMP. Las historias de usuario establecidas para este grupo se reflejan en la Tabla 4.3.

Nombre de la Historia de Usuario	Descripción
Herramienta Ping	Ejecuta Ping a un *hostname* dado, a través de SNMP.
Herramienta Cisco IOS	Permite cargar hacia el *router*, o descargar desde el *router* el archivo correspondiente al IOS (*Internetwork Operating System*).
Herramienta ConfigFile	Permite cargar hacia el *router*, o descargar desde el *router* el archivo correspondiente a la configuración actual.
Detalles Generales	Muestra datos generales acerca del *router*.
Detalles Chasis	Muestra información relacionada al chasis del dispositivo.
Detalles Flash	Muestra detalles de la memoria Flash usada por el dispositivo.
Detalles IOS	Muestra información correspondiente al IOS del *router*.
Detalles IP Routes	Muestra información de las IP Routes que se encuentran configuradas en el dispositivo.
Detalles Interfaces	Proporciona información de cada una de las interfaces del *router* y su estado actual.
Detalles VoIP	Muestra información relacionada con el protocolo de VoIP configurado en el dispositivo.
Detalles CDP	Muestra información del protocolo CDP (*Cisco Discovery Protocol*) configurado en el *router*.
Detalles VPDN	Muestra información de la configuración del protocolo VPDN (*Virtual Private Dialup Network*) existente en el dispositivo.
Monitor CPU	Grafica el consumo de CPU en el dispositivo.
Monitor Memoria	Grafica el consumo de memoria en el dispositivo.
Monitor Interfaces	Grafica el uso de cada una de las interfaces del *router*.
Configuración de Eventos	Permita crear y configurar eventos utilizando el protocolo RMON (*Remote Network Monitoring*).

Nombre de la Historia de Usuario	Descripción
Configuración de Alarmas	Permite crear y configurar alarmas a través de RMON.
Configuración NetFlow	Permite configurar características relacionadas con el protocolo Cisco NetFlow.
Configuración CDP	Permite configurar características relacionadas con el protocolo CDP en el *router*.
Cambio de Idioma	Permite cambiar el idioma de español a inglés o viceversa.

Tabla 4.3: Historias de usuario para RMSWeb.

4.1.2 Plan de iteraciones

Una vez identificadas las historias de usuario, se procede a dividir el proyecto en iteraciones cuya duración debe oscilar entre una y tres semanas. A continuación en la Tabla 4.4 se observa el plan de iteraciones establecido.

Iteración	Grupo de Historias	Historias de Usuario
1	Biblioteca de funciones SNMP	Enviar GetRequest
		Enviar SetRequest
		Enviar GetNextRequest
		Enviar GetBulkRequest
		Enviar Trap
		Enviar InformRequest
		Enviar Trap SNMPv2c
		Ejecutar Walk
2	Biblioteca de funciones SNMP	Registrar eventos
		Validar datos
3	NMSWeb	Solicitudes SNMP
4	NMSWeb	Envío de Notificaciones
5	NMSWeb	Cambio de Idioma
6	RMSWeb	Herramienta Ping
		Herramienta Cisco IOS
		Herramienta ConfigFile
7	RMSWeb	Detalles Generales
		Detalles Chasis
		Detalles Flash
		Detalles IOS
		Detalles IP Routes
		Detalles Interfaces
		Detalles VoIP

Iteración	Grupo de Historias	Historias de Usuario
		Detalles CDP
		Detalles VPDN
8	RMSWeb	Monitor CPU
		Monitor Memoria
		Monitor Interfaces
9	RMSWeb	Configuración de Eventos
		Configuración de Alarmas
		Configuración NetFlow
		Configuración CDP
10	RMSWeb	Cambio de Idioma

Tabla 4.4: Plan de iteraciones.

4.1.3 Herramientas y tecnologías

En la siguiente sección se describirán las herramientas y tecnologías utilizadas en el desarrollo del proyecto.

- ***Emacs*:** GNU *Emacs*[1] es un editor de texto extensible y configurable. Entre sus características se encuentran:

  - Se ejecuta en una gran variedad de sistemas operativos, funcionando tanto en entornos de terminal de texto como en entornos de interfaz gráfica de usuario.
  - Posee una documentación completa y detallada.
  - Proporciona soporte para varios idiomas.
  - Tiene un gran número de extensiones para añadir diversas funcionalidades.
  - Adapta su comportamiento al tipo de texto que está editando, a través de modos de edición.
  - Se encuentra disponible bajo GNU GPL (*GNU General Public License*), por lo que su código fuente puede ser libremente modificado y redistribuido.

  GNU *Emacs* se seleccionó para el desarrollo de la biblioteca, debido a que facilita la utilización de los estándares de codificación establecidos por PEAR. Por ejemplo, como se observa en la Figura 4.1 permite almacenar

[1] http://www.gnu.org/software/emacs

los archivos bajo la codificación *ISO-8859-1* que deben tener todos los paquetes que forman parte del repositorio PEAR.

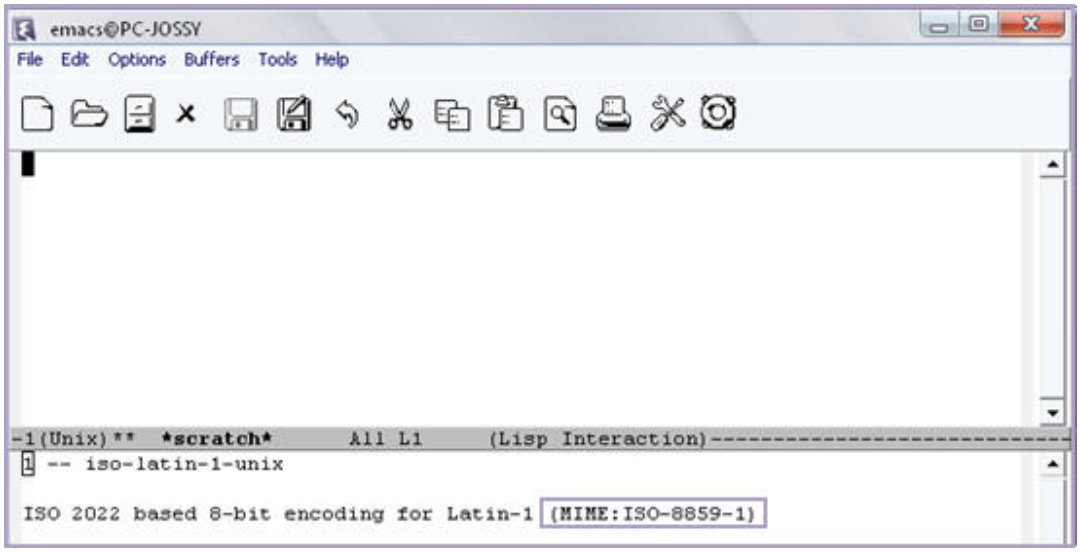

Figura 4.1: Editor GNU *Emacs*.

Cabe destacar que para forzar al editor a que sean aplicadas las reglas de codificación PEAR a todos los archivos con extensión .php, se debe modificar el archivo de configuración *php-mode.el* que se incluye en la instalación. En la Figura 4.2 se observa la línea que debe ser modificada.

```
(defcustom php-mode-force-pear t
  "Normally PEAR coding rules are enforced only when the
filename contains \"PEAR\"\ Turning this on will force
PEAR rules on all PHP files."
  :type 'boolean
  :group 'php)
```

Figura 4.2: Configuración del archivo *php-mode.el*.

- ***Xajax*:** *Xajax*[2] es una biblioteca *Open Source* compuesta por un conjunto de clases PHP y *javascript*, para la construcción de aplicaciones web interactivas con tecnología *Ajax*.

  Las aplicaciones desarrolladas con *Xajax* pueden comunicarse de manera asíncrona con funciones PHP que se encuentran del lado del servidor, y de esta manera actualizar el contenido de una página sin tener que

[2] http://xajaxproject.org

recargarla. Esta biblioteca es compatible con la mayoría de los navegadores utilizados en la actualidad, tales como: *Mozilla Firefox, Internet Explorer, Safari* y *Opera*.

- ***AutoSuggest*:** *AutoSuggest*[3] es una clase *Open Source* de *javascript,* que permite añadir un menú desplegable de valores sugeridos a un campo de texto. De esta manera el usuario puede seleccionar algún valor de la lista, como se observa en el ejemplo de la Figura 4.3.

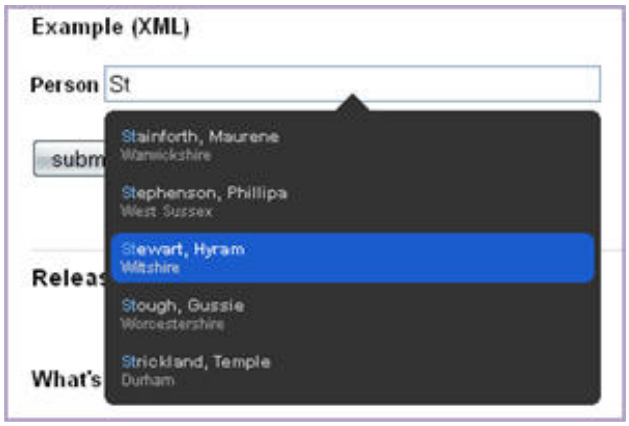

Figura 4.3: Ejemplo de *AutoSuggest.*

La clase *AutoSuggest* recibe los valores que serán listados de archivos *XML* (*eXtensible Markup Language*) o *JSON* (*JavaScript Object Notation*).

- ***Open Flash Chart*:** *Open Flash Chart*[4] es un proyecto *Open Source* distribuido bajo licencia *GNU GPL,* que permite generar diversos tipos de gráficos en aplicaciones web.

  Para trabajar con este componente es necesario incluir una referencia a *Open Flash Chart* en el archivo *HTML*, y además proporcionar el archivo de datos que se utilizará para la construcción de la gráfica. Este archivo puede estar escrito en lenguaje *PHP*, *Perl*, *Phyton*, *Java*, entre otros.

- ***XAMPP*:** *XAMPP*[5] es una distribución del servidor web *Apache*, que se caracteriza por ser de fácil instalación. Actualmente se encuentra disponible en versiones para los sistemas operativos Linux, Windows, Mac

[3] http://gadgetopia.com/autosuggest
[4] http:// teethgrinder.co.uk/open-flash-chart
[5] http://www. apachefriends.org/en/xampp.html

OS X, y Solaris. Además, se distribuye gratuita y libremente, bajo los términos de la licencia *GNU GPL*.

XAMPP fue construido con el propósito de utilizarse como herramienta de desarrollo y prueba de aplicaciones con tecnología web, por este motivo incluye en su instalación *MySQL*, PHP y *Perl*.

- ***VMware Workstation*:** *VMware Workstation*[6] es una aplicación que permite ejecutar múltiples sistemas operativos simultáneamente, en un único computador.

  Esta herramienta proporciona diversas ventajas a los desarrolladores de software, permitiendo:

  - Configurar y probar equipos y servidores en máquinas virtuales, antes de colocarlos en producción.
  - Desarrollar y probar aplicaciones con soporte para múltiples sistemas operativos, en un único computador.
  - Conectar múltiples máquinas virtuales para realizar pruebas sobre entornos de redes heterogéneas.

- ***Wireshark*:** *Wireshark*[7] es una herramienta utilizada para la captura de tráfico y análisis de protocolos de redes. Entre sus características más importantes, se encuentran:

  - Brinda soporte para el análisis de múltiples protocolos.
  - Está disponible para sistemas operativos Windows, Linux, Mac OS X, Solaris, FreeBSD, NetBSD, entre otros.
  - Es distribuido bajo licencia *GNU GPL*.

  Wireshark es una aplicación muy útil para detectar los problemas en una red, examinar fallas en la seguridad y estudiar la implementación de los protocolos.

4.2 Fase de Diseño

En la fase de diseño se elaboraron las tarjetas CRC, con el propósito de identificar los objetos involucrados en la Biblioteca de funciones SNMP.

[6] http://www.*vmware.com/products/ws*
[7] http://www.*wireshark.org*

Asimismo, se crearon los prototipos de interfaz para el NMSWeb y el RMSWeb.

4.2.1 Tarjetas CRC

A través de este artefacto, es posible tener una primera visión de las que serán las clases implementadas, sus métodos y relaciones. Cada tarjeta CRC representa una clase, con sus responsabilidades y el nombre de las clases colaboradoras.

Seguidamente se describen cada una de las tarjetas que fueron identificadas durante la etapa de diseño.

- **ASN1:** En la Tabla 4.5 se observa la tarjeta CRC que representa esta clase.

ASN1	
Definir constantes que serán utilizadas por la biblioteca. Construir la representación correcta de un tipo de dato determinado para su envío, según las reglas básicas de codificación. Decodificar valores recibidos en formato TLV, para obtener su representación natural.	Exception

Tabla 4.5: Tarjeta CRC ASN1.

- **Transport:** Esta clase será utilizada para trasmitir y recibir los datos del protocolo. En la Tabla 4.6 se observa su descripción.

Transport	
Enviar un paquete a una dirección y puerto dado. Recibir un paquete desde una dirección y puerto dado.	Exception

Tabla 4.6: Tarjeta CRC Transport.

- **PDU:** En la Tabla 4.7 se tiene la tarjeta CRC correspondiente a esta clase.

PDU	
Representar objetos del tipo PDU para las solicitudes, notificaciones y respuestas del protocolo. Construir e inicializar los campos de la PDU correspondiente a una solicitud o notificación en específico. Codificar objetos del tipo PDU utilizando las reglas básicas de codificación.	ASN1 Exception

Tabla 4.7: Tarjeta CRC PDU.

- **Exception:** Esta clase proporciona las excepciones que serán arrojadas por la biblioteca cuando ocurra algún comportamiento no válido. En la Tabla 4.8 se observa la tarjeta CRC utilizada para representar la clase Exception.

Exception	
Definir los objetos necesarios para describir los diferentes tipos de excepciones posibles que pueden ocurrir en la biblioteca.	No posee colaboradores.

Tabla 4.8: Tarjeta CRC Exception.

- **Log:** En la Tabla 4.9 se observa la descripción de esta clase.

Log	
Realizar operaciones de creación y escritura de archivos, para mantener un registro de los eventos ocurridos.	Exception

Tabla 4.9: Tarjeta CRC Log.

- **VarBind:** Como se observa en la Tabla 4.10, esta clase permite almacenar los identificadores de los objetos consultados y sus valores.

VarBind	
Representar objetos del tipo VarBind. Almacenar el tipo de dato del valor correspondiente al VarBind.	No posee colaboradores.

Tabla 4.10: Tarjeta CRC VarBind.

- **VarBindList:** En la Tabla 4.11 se observa la tarjeta CRC que representa a la clase VarBindList.

VarBindList	
Representar una lista de objetos del tipo VarBind. Proporcionar métodos para agregar o remover objetos de la lista.	VarBind

Tabla 4.11: Tarjeta CRC VarBindList.

- **Message:** La clase Message representa los mensajes SNMP. En la Tabla 4.12 se describen sus responsabilidades y clases colaboradoras.

Message	
Representar los mensajes de solicitudes SNMP, compuestos por la versión, la comunidad, y la PDU. Codificar mensajes SNMP para su envío. Decodificar mensajes de respuesta SNMP, y almacenarlos en objetos PDU.	PDU ASN1 VarBind VarBindList Exception

Tabla 4.12: Tarjeta CRC Message.

- **SNMP:** Representa la clase principal de la biblioteca de funciones. En la Tabla 4.13 se observa su descripción.

SNMP	
Enviar solicitud GetRequest. Enviar solicitud SetRequest. Enviar solicitud GetNextRequest. Enviar solicitud GetBulkRequest. Enviar notificación Trap. Enviar notificación InformRequest. Ejecutar operación Walk.	Log Message PDU VarBind VarBindList Transport Exception

Tabla 4.13: Tarjeta CRC SNMP.

4.2.2 Prototipo de interfaz

En esta sección se presentan los prototipos de interfaz construidos para el NMSWeb y el RMSWeb. Estas aplicaciones se desarrollaron con el propósito de validar el funcionamiento de la Biblioteca SNMP.

- **NMSWeb:** A través de las historias de usuario recabadas durante la fase de planificación, se establecieron las áreas en las que debe dividirse esta herramienta. En la Figura 4.4 se presenta un esquema que describe las áreas identificadas durante el diseño.

Figura 4.4: Áreas de NMSWeb.

Asimismo, el prototipo de interfaz comprende las siguientes áreas:

- Logo: Contiene información que identifica la aplicación, tal como el nombre y el logo.

- Formulario de datos: Esta área está destinada a la recolección de los datos necesarios para poder ejecutar las operaciones. Para ello, tendrá un formulario con diversos campos de texto donde se podrá seleccionar la operación a realizar e ingresar la información requerida.

- Área de resultados: En este recuadro se mostrarán mensajes con los resultados arrojados por la operación ejecutada.

- Pie de página: Contiene información acerca de la aplicación y sus desarrolladores.

- Barra superior: Esta barra posee enlaces a cada una de las secciones de la aplicación. Adicionalmente, proporciona la opción de cambio de idioma como se observa en la Figura 4.5.

Figura 4.5: Barra superior NMSWeb.

- Barra lateral: Al igual que la barra superior, proporciona enlaces a las secciones de la aplicación (Solicitudes SNMP, Envío de notificaciones, Información).

Un ejemplo del prototipo construido, se observa en la Figura 4.6.

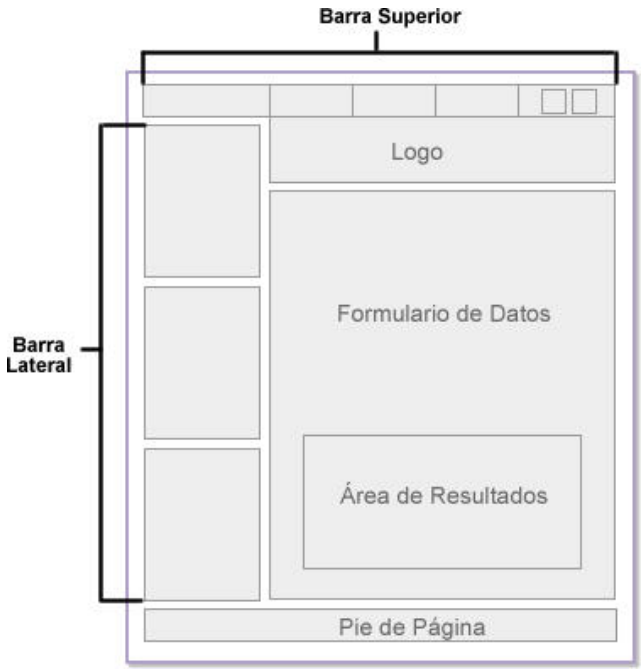

Figura 4.6: Prototipo de interfaz NMSWeb.

- **RMSWeb:** De acuerdo a las historias de usuario identificadas durante la planificación, se organizaron las secciones de la aplicación en cuatro áreas principales, tal y como se aprecia en la Figura 4.7.

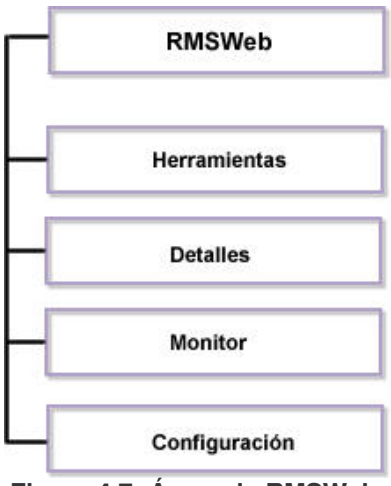

Figura 4.7: Áreas de RMSWeb.

Cada una de las áreas principales, se divide en varias subáreas. Esta distribución se observa en la Tabla 4.14.

Área	Subáreas
Herramientas	Ping SNMP
	Cisco IOS
	Config File
Detalles	General
	Chasis
	Flash
	IOS
	IP Routes
	Interfaces
	VoIP
	CDP
	VPN
Monitor	CPU
	Memoria
	Interfaces
Configuración	Eventos RMON
	Alarmas RMON
	NetFlow
	CDP

Tabla 4.14: Subáreas de RMSWeb.

En la Figura 4.8 se tiene el prototipo de interfaz diseñado para el RMSWeb.

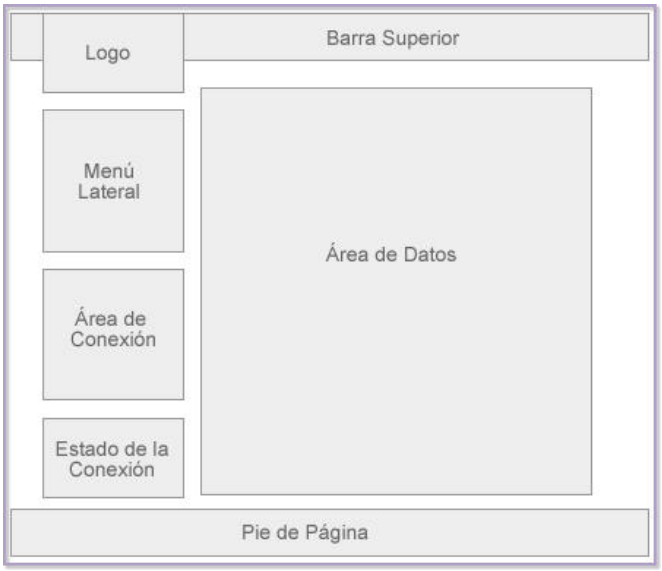

Figura 4.8: Prototipo de interfaz RMSWeb.

A continuación se describirán cada una de las secciones que conforman el prototipo.

- Logo: Contiene el nombre y el logo que identifica a la aplicación.

- Barra superior: Proporcionará un menú de navegación para acceder a las áreas principales del RMSWeb, y permitir el cambio de idioma. En la Figura 4.9 se observan las opciones que formarán parte del menú.

Figura 4.9: Barra superior RMSWeb.

- Menú lateral: Este menú mostrará las subáreas correspondientes al área que haya sido seleccionada en la barra superior.

- Área de conexión: En esta área se tendrá un formulario para que el usuario ingrese los datos correspondientes al *router* (dirección IP, nombre de la comunidad de lectura, nombre de la comunidad de escritura), con el fin de iniciar la conexión con el dispositivo.

- Estado de la conexión: En esta sección se mostrará información acerca del estado actual de la conexión con el dispositivo.

- Área de datos: Tendrá la información correspondiente a la opción de menú seleccionada y la operación realizada.

- Pie de página: Contiene información acerca de la aplicación y sus desarrolladores. Además, proporciona enlaces hacia las cuatro áreas principales del RMSWeb.

4.3 Fase de Implementación

El proceso de implementación se lleva a cabo, siguiendo el plan de iteraciones establecido durante la fase de planificación. En esta etapa se codifican los módulos que dan soluciones a las historias de usuario agrupadas en cada una de las iteraciones.

Para cada iteración se describen los aspectos más importantes tomados en cuenta durante la implementación, así como el producto final obtenido, luego de aplicar las refactorizaciones necesarias al modelo inicial, construido durante la etapa de diseño.

4.3.1 Iteración 1

Durante la primera iteración se codificaron las clases y métodos necesarios, para implementar las operaciones del protocolo SNMP v1/v2c.

Cabe destacar que a lo largo del proceso de codificación se cumplieron con los estándares establecidos por la distribución oficial de PEAR, con el objetivo de mantener el código legible y compatible con las extensiones que forman parte de este repositorio.

Seguidamente se describirán cada una de las clases implementadas en la iteración inicial.

- **Net_SNMP_ASN1:** De acuerdo a lo establecido en las tarjetas CRC durante la fase de diseño, el propósito de esta clase se resume en construir la representación correcta de diversos tipos de datos, utilizando las reglas básicas de codificación. Asimismo, permite decodificar un dato contenido en una tripleta TLV a fin de obtener su representación natural.

  En la Figura 4.10 se observa el diagrama con la implementación definitiva de la clase Net_SNMP_ASN1.

Figura 4.10: Diagrama de la clase Net_SNMP_ASN1.

De manera general, los métodos "*prepare*" reciben como parámetro una variable con el dato a codificar, y retornan una cadena binaria con la representación TLV correspondiente al dato. Por su parte, los métodos "*process*" reciben como parámetro un arreglo que almacena los octetos de la TLV, y retornan una variable con la representación natural del campo *Value* extraído de la tripleta.

Un ejemplo de las operaciones "*prepare*" se observa en la Figura 4.11, con el segmento de código correspondiente al método "*prepareIpAddress*".

```
function prepareIpAddress($ip)
{

    if (!($ip == long2ip(ip2long($ip)))) {
        throw new Net_SNMP_TypeException('Dirección IP inválida');
    }
    $dir   = explode('.', $ip);
    $value = pack('C4', $dir[0], $dir[1], $dir[2], $dir[3]);
    $tlv   = $this->prepareTypeLength(IPADDRESS, $value);
    return $tlv;
}
```

Figura 4.11: Código del método *prepareIpAddress.*

- **Transport:** En la fase de diseño se estableció como propósito de este objeto permitir el envío y la recepción de los mensajes SNMP. Estas operaciones pueden ser ejecutadas a través del protocolo de transporte UDP/IPv4 o UDP/IPv6, según la elección del usuario.

  Por esta razón, la implementación final comprende la codificación de tres clases:

  - Net_SNMP_TransportFactory: Es una clase abstracta, con una operación que permite centralizar la creación de objetos, dependiendo del protocolo de transporte que será utilizado.

  - Net_SNMP_TransportUDP: Esta clase posee los atributos y las operaciones indispensables para transmitir y recibir los datos del protocolo.

  - Net_SNMP_TransportUDP6: Hereda los atributos y las operaciones de la clase Net_SNMP_TransportUDP. Además, redefine los métodos necesarios para adaptar la trasmisión y recepción de datos al protocolo de transporte UDP/IPV6. Para ello, hace uso del paquete Net_IPv6 de PEAR.

  La descripción anterior, se observa con más detalle en la Figura 4.12.

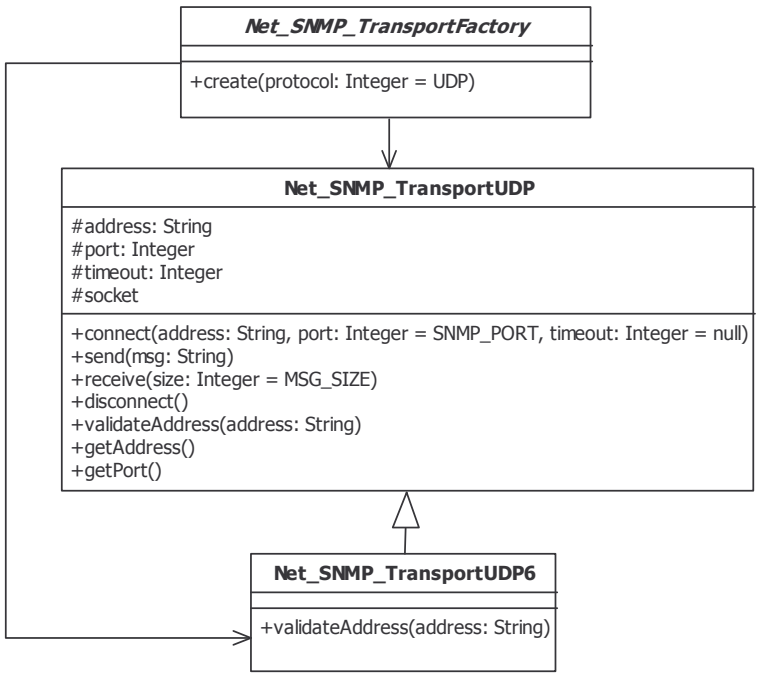

Figura 4.12: Diagrama de las clases Net_SNMP_Transport.

- **Net_SNMP_PDU:** La implementación de la clase Net_SNMP_PDU se llevó a cabo, con el objetivo de proporcionar una estructura adecuada para representar objetos del tipo PDU.

  Adicionalmente, se construyeron métodos para permitir codificar estos objetos, utilizando las reglas básicas de codificación, y a través de la colaboración de la clase Net_SNMP_ASN1.

  En la Figura 4.13 se describen los atributos y métodos definidos durante la implementación de la clase Net_SNMP_PDU.

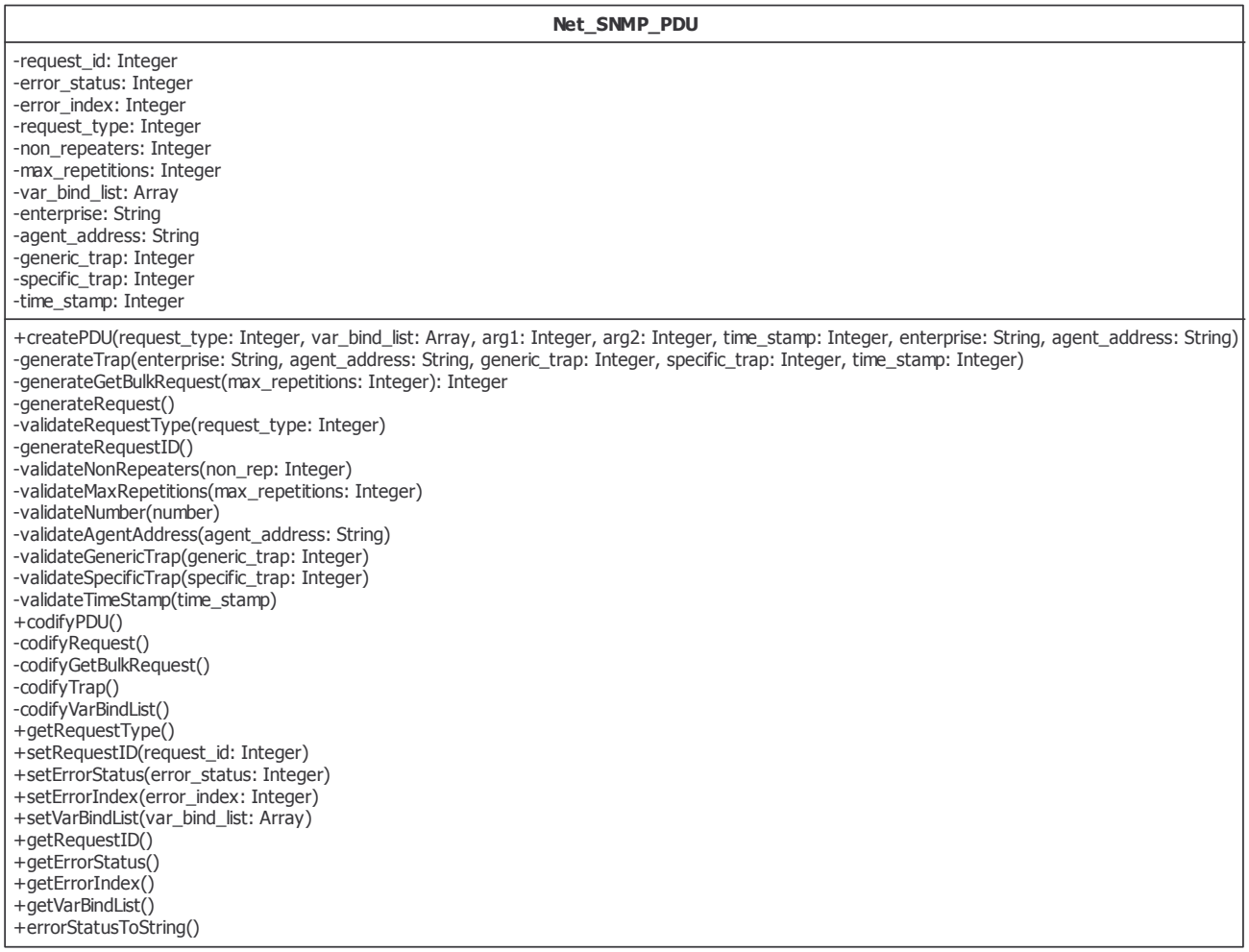

Figura 4.13: Diagrama de la clase Net_SNMP_PDU.

La representación de los objetos PDU se construye de acuerdo al tipo de solicitud SNMP que contienen, como se observa en la función "*codifyPDU*" descrita en la Figura 4.14.

```
function codifyPDU()
{
    switch ($this->request_type) {

    case GET_REQUEST:
    case GET_NEXT_REQUEST:
    case SET_REQUEST:
    case INFORM_REQUEST:
    case SNMPV2_TRAP:
        return $this->codifyRequest();
    case GET_BULK_REQUEST:
        return $this->codifyGetBulkRequest();
    case TRAP:
        return $this->codifyTrap();
    }
}
```

Figura 4.14: Código del método *codifyPDU*.

- **Net_SNMP_VarBind:** A través de esta estructura es posible almacenar los datos correspondientes a los objetos consultados, por medio del protocolo SNMP. Es por ello que durante la implementación se definieron los siguientes atributos:

  - object_identifier: Es una cadena utilizada para almacenar el identificador de los objetos que serán consultados o modificados, haciendo uso de las operaciones del protocolo.

  - value: Es un atributo que se utiliza para almacenar el valor de los objetos.

  - type: Es un atributo entero, que representa el tipo de dato perteneciente al valor almacenado en el objeto VarBind.

  El diagrama de clases correspondiente a Net_SNMP_VarBind se aprecia en la Figura 4.15.

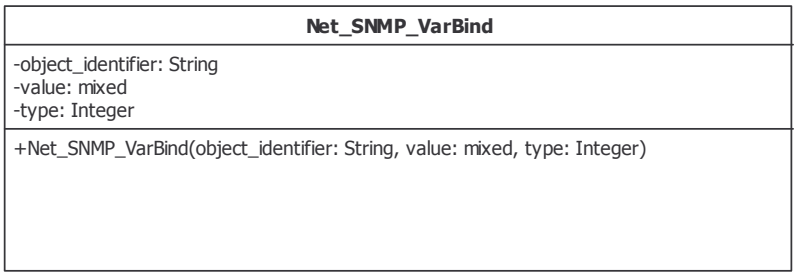

Figura 4.15: Diagrama de la clase Net_SNMP_VarBind.

- **Net_SNMP_VarBindList:** La implementación de esta clase comprendió la creación de una estructura de lista, para almacenar objetos pertenecientes a la clase Net_SNMP_VarBind descrita anteriormente.

Por esta razón, se codificaron métodos necesarios para poder manipular la estructura de la lista, permitiendo agregar o eliminar objetos. En la Figura 4.16 se observa la descripción de los atributos y métodos que se definieron.

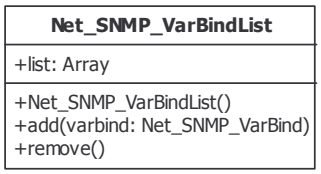

Figura 4.16: Diagrama de la clase Net_SNMP_VarBindList.

- **Net_SNMP_Message:** La clase Net_SNMP_Message se implementó con el fin de crear una estructura idónea, para la representación de los mensajes que serán intercambiados a través del protocolo. En consecuencia, se definieron los siguientes atributos:

  - version: Es una variable de tipo entero, que almacena el valor correspondiente a la versión del protocolo soportada por el respectivo mensaje.

  - community: Es una cadena que almacena la comunidad utilizada por el protocolo para la autenticación con el agente.

  - pdu: Es un objeto de la clase Net_SNMP_PDU que permite almacenar la PDU correspondiente al mensaje SNMP.

  En la Figura 4.17 se aprecia el diagrama correspondiente a la implementación definitiva de la clase Net_SNMP_Message.

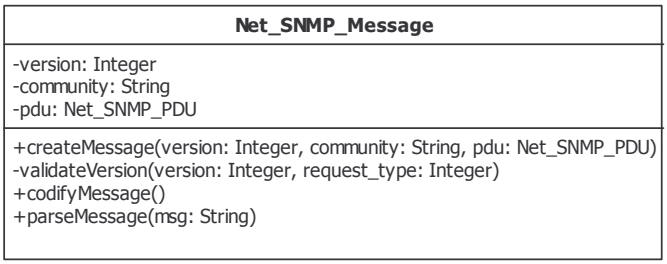

Figura 4.17: Diagrama de la clase Net_SNMP_Message.

Cabe destacar que además de los atributos descritos anteriormente, la clase Net_SNMP_Message proporciona un método llamado "*codifyMessage*" que permite obtener la representación correcta del mensaje para su envío, utilizando las reglas básicas de codificación. En la Figura 4.18 se puede observar el segmento correspondiente a este método.

```
function codifyMessage()
{
    $asn1      = new Net_SNMP_ASN1();
    $version   = $asn1->prepareInteger($this->version);
    $community = $asn1->prepareOctetString($this->community);
    $data      = $this->pdu->codifyPDU();

    return $asn1->prepareSequence($version.$community.$data);

}
```

Figura 4.18: Código del método *codifyMessage*.

Asimismo, se definió una operación denominada "*parseMessage*" que se encarga de decodificar un mensaje *response* obtenido como respuesta a alguna solicitud del protocolo SNMP, y lo retorna en un objeto del tipo Net_SNMP_PDU.

- **Net_SNMP:** Para finalizar con la primera iteración se codificó la clase Net_SNMP, que implementa las operaciones del protocolo SNMP v1/v2c.

  Net_SNMP representa la clase principal de la biblioteca de funciones, que será utilizada por los desarrolladores. Por este motivo, proporciona los métodos necesarios para el envío de las solicitudes SNMP.

  En la Tabla 4.15 se describen las operaciones proporcionadas por esta clase al usuario final:

Nombre	Parámetros		Descripción
getRequest	oids	Objeto de la clase Net_SNMP_VarBindList.	Permite leer el valor de uno o más objetos SNMP.
setRequest	oids	Objeto de la clase Net_SNMP_VarBindList.	Permite establecer o modificar el valor de uno o más objetos SNMP.
getNextRequest	oids	Objeto de la clase Net_SNMP_VarBindList.	Permite consultar el valor de uno o más objetos SNMP.
getBulkRequest	oids	Objeto de la clase Net_SNMP_VarBindList.	Permite enviar una solicitud del tipo GetBulkRequest para consultar el valor de varios objetos SNMP.
	nonrep	Valor *non repeaters*.	
	maxrep	Valor *max repetitions*.	
trap	enterprise	Cadena con el valor *enterprise*.	Envía una notificación trap empleando SNMP v1.
	agent_address	Cadena con la dirección IP del agente.	
	generic_trap	Entero que indica el tipo de trap genérico.	
	especific_trap	Entero que indica el tipo de trap específico.	
	time_stamp	Entero con el tiempo transcurrido entre la	

Nombre	Parámetros		Descripción
		última reinicialización del agente y la generación del trap.	
	oids	Objeto de la clase Net_SNMP_VarBindList.	
informRequest	oids	Objeto de la clase Net_SNMP_VarBindList.	Envía una notificación de tipo informRequest.
snmpv2Trap	oids	Objeto de la clase Net_SNMP_VarBindList.	Envía una notificación trap empleando SNMP v2c.
snmpwalk	root	Cadena con el OID a partir del cual se realizará la consulta.	Permite consultar un subárbol de la MIB a partir de un OID dado como parámetro.
	objects	Representa el número máximo de objetos a solicitar en la consulta.	

Tabla 4.15: Descripción de los métodos de la clase Net_SNMP.

En la Figura 4.19 se observa el diagrama con la implementación definitiva de la clase Net_SNMP.

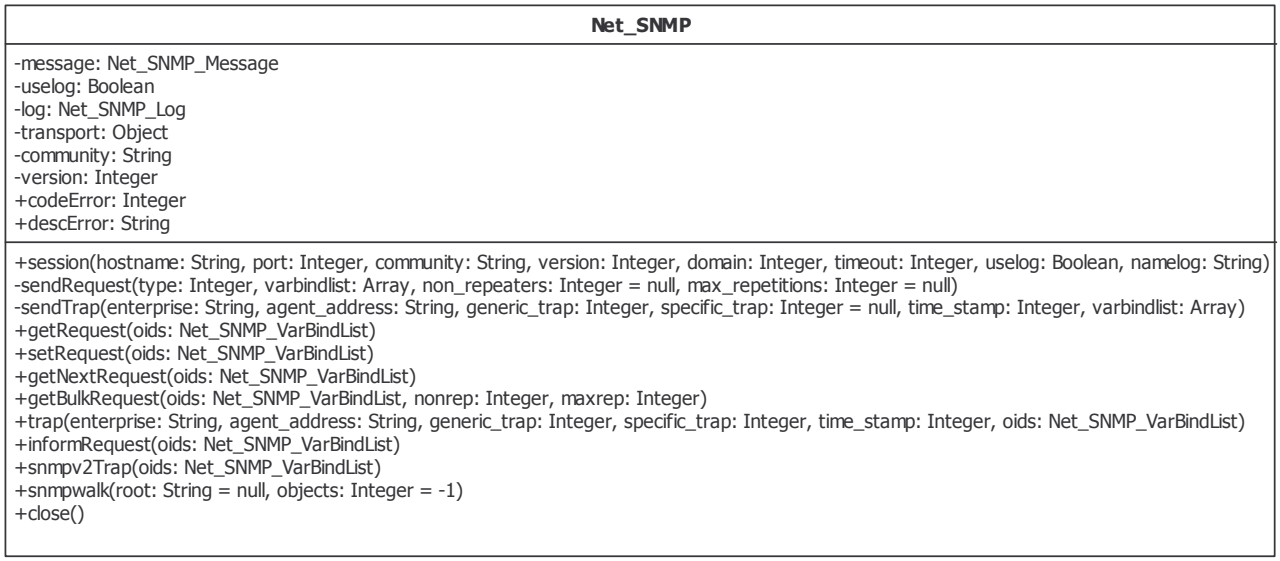

Figura 4.19: Diagrama de la clase Net_SNMP.

4.3.2 Iteración 2

De acuerdo con el plan de iteraciones, en esta fase se codificaron las siguientes historias de usuario:

- Registrar eventos.
- Validar datos.

Para ello, se añadieron a la biblioteca los módulos que se describen a continuación.

- **Net_SNMP_Log:** Esta clase proporciona operaciones para la creación y escritura de archivos. Esto con el fin de permitir integrar un registro o log de eventos a las funciones de la biblioteca.

  De esta manera se aporta a los desarrolladores la habilidad de monitorear las acciones realizadas por la aplicación, revisando las salidas registradas en los archivos de log.

  En la Figura 4.20 se aprecia el diagrama correspondiente a este módulo.

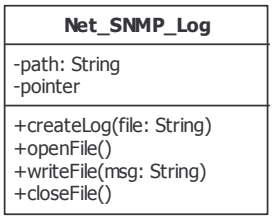

Figura 4.20: Diagrama de la clase Net_SNMP_Log.

- **Net_SNMP_Exception:** Según el estándar de codificación PEAR, a partir de la versión PHP 5.0 es necesario utilizar excepciones como mecanismo de manejo de errores. Las excepciones deberán ser arrojadas siempre que alguna condición de error se cumpla.

  Por otro lado, todas las excepciones que formen parte de los paquetes PEAR deben ser descendientes de la clase PEAR_Exception. Adicionalmente, cada paquete PEAR debe proporcionar una excepción de nivel superior llamada: <Package_Name>_Exception. De esta forma, se garantiza que el paquete nunca arrojará excepciones que no sean descendientes de la excepción de nivel superior [16].

  Para la biblioteca de funciones SNMP se estableció una clase denominada Net_SNMP_Exception, de la que heredan los diversos tipos de excepciones que pueden ocurrir.

  En la Tabla 4.16 se describe el grupo de excepciones que fueron definidas como mecanismo de manejo de errores en los métodos proporcionados por la biblioteca.

Clase	Descripción
Net_SNMP_Exception	Representa la excepción de nivel superior, de la que descienden todas las posibles excepciones arrojadas por la biblioteca.
Net_SNMP_TypeException	Excepción arrojada cuando se recibe un tipo de dato que no corresponde al esperado.
Net_SNMP_LengthException	Excepción arrojada cuando la longitud de algún tipo de dato no es válida.
Net_SNMP_TypeRequestException	Excepción arrojada cuando el tipo de solicitud de un mensaje es inválido.
Net_SNMP_BadVersionException	Excepción arrojada cuando la versión del mensaje no es válida, o no se encuentra soportada por la biblioteca.
Net_SNMP_BadTypeException	Excepción arrojada cuando el campo *Type* de una tripleta TLV es inválido, o no soportado por la biblioteca.
Net_SNMP_ConnectException	Excepción arrojada cuando ocurre alguna falla que impide que la conexión con el dominio de transporte sea establecida.
Net_SNMP_SendException	Excepción arrojada cuando ocurre algún error durante el envío de un mensaje.
Net_SNMP_ReceiveException	Excepción arrojada cuando ocurre algún error durante la recepción de un mensaje.
Net_SNMP_DisconnetException	Excepción arrojada al intentar finalizar una conexión que no ha sido establecida, o no se encuentra activa.
Net_SNMP_FileException	Excepción arrojada cuando ocurren errores en las operaciones relacionadas con el manejo del log de eventos.
Net_SNMP_BadFieldPDUException	Excepción arrojada cuando algún campo de una PDU no cumple con las restricciones necesarias.

Tabla 4.16: Descripción de las excepciones de la Biblioteca SNMP.

Una vez construidos los módulos descritos anteriormente, fue necesario realizar modificaciones sobre los productos obtenidos en la primera iteración, con el fin de poder integrar las excepciones como mecanismo de manejo de errores.

De igual forma, se realizaron modificaciones en la clase Net_SNMP, para brindar la posibilidad de que el usuario final pueda definir un archivo como log de eventos, y así poder registrar todas las actividades realizadas por las distintas operaciones de la biblioteca de funciones SNMP.

En la Figura 4.21 se aprecia el diagrama de clases simplificado de la implementación definitiva de la biblioteca.

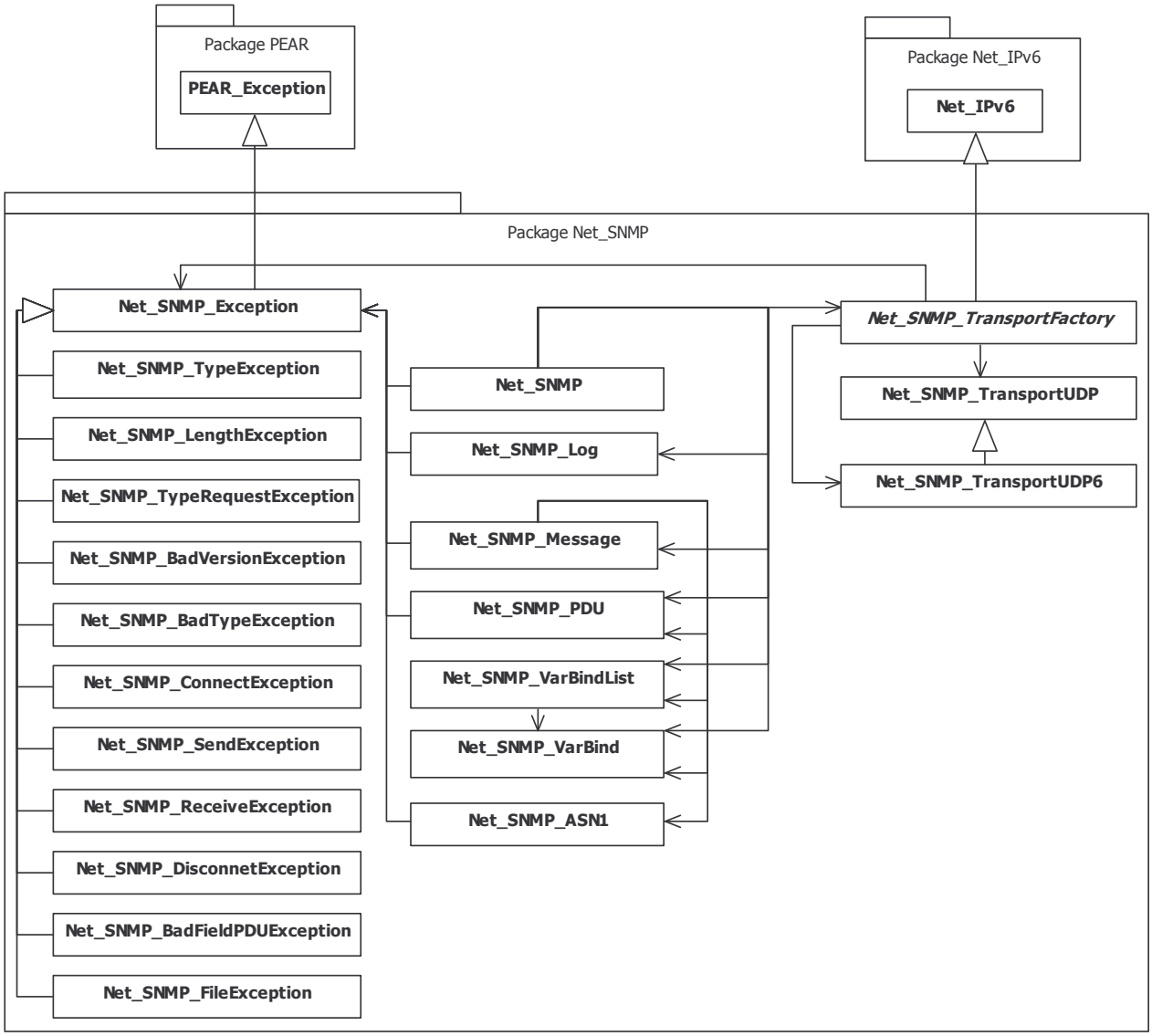

Figura 4.21: Diagrama de clases simplificado de la Biblioteca SNMP.

4.3.3 Iteración 3

A partir de esta iteración se dio inicio a la implementación de la interfaz web para el envío de solicitudes y notificaciones SNMP, denominada NMSWeb.

Para ello, se utilizó la biblioteca de funciones desarrollada durante las iteraciones anteriores, permitiendo así comprobar su correcto funcionamiento y facilidad de uso.

En esta etapa se codificó el área destinada a las solicitudes SNMP. Como primer paso de la implementación, se adaptó y modificó la interfaz web, de acuerdo con el prototipo construido durante la fase de diseño.

En la Figura 4.22 se puede observar la interfaz definitiva, con los campos de entrada necesarios para realizar las operaciones de esta área.

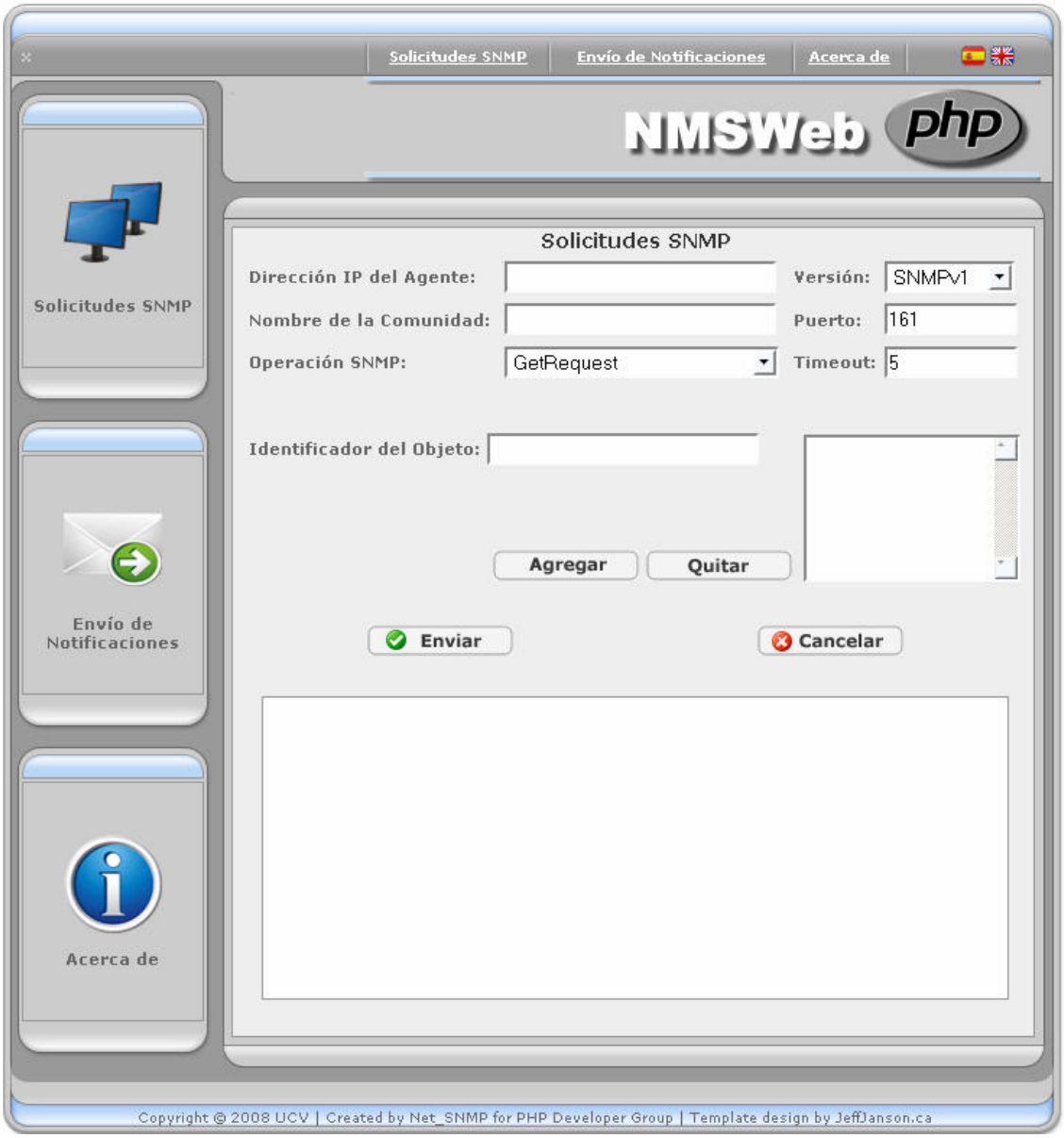

Figura 4.22: Interfaz de Solicitudes NMSWeb.

Seguidamente, se codificaron funciones *JavaScript* para validar los datos de entrada y modificar los campos mostrados, de acuerdo a la operación SNMP seleccionada.

Para el campo de entrada correspondiente al identificador del objeto, se utilizó la clase *Open Source Autosuggest*, con el propósito de mostrar sugerencias al momento de ingresar los datos. Estas sugerencias forman parte de algunos identificadores de los objetos definidos en la MIB II, como se observa en la Figura 4.23.

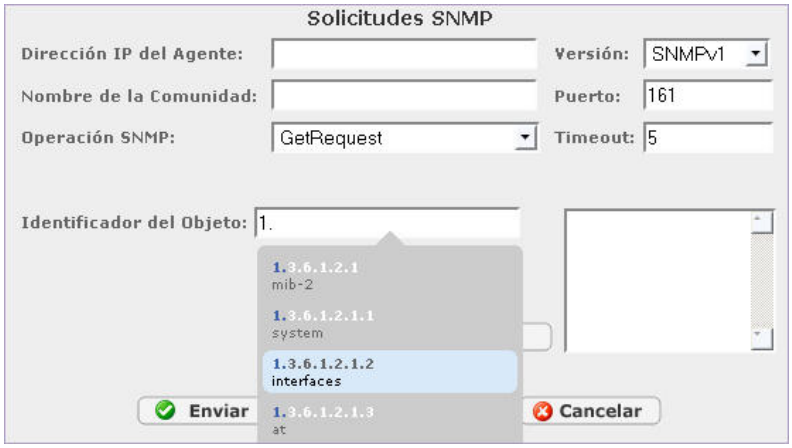

Figura 4.23: Campo de identificador del objeto en NMSWeb.

Por otra parte, para mostrar los resultados de las solicitudes realizadas se utilizó la clase *xajax*. De esta forma, los datos del formulario son procesados a través de funciones PHP, y las respuestas arrojadas por las operaciones son mostradas sin necesidad de recargar la página.

Entre las operaciones de solicitud SNMP que fueron implementadas desde la interfaz web se encuentran:

- GetRequest.
- GetNextRequest.
- SetRequest.
- GetBulkRequest.
- Walk.

En la Figura 4.24 se aprecia un segmento de código de la función "*sendRequest*" que ejecuta la solicitud SNMP respectiva, de acuerdo con la elección del usuario en la interfaz.

```
switch ($operation) {
    case 1:
        $name     = 'GetRequest';
        $response = $snmp->getRequest($varbindlist);
        break;
    case 2:
        $name     = 'GetNextRequest';
        $response = $snmp->getNextRequest($varbindlist);
        break;
    case 3:
        $name     = 'SetRequest';
        $response = $snmp->setRequest($varbindlist);
        break;
    case 4:
        $name     = 'Walk';
        $response = $snmp->snmpwalk($oid);
        break;
    case 5:
        $name     = 'GetBulkRequest';
        $nonrep   = (int)$formSNMP["nonrep"];
        $maxrep   = (int)$formSNMP["maxrep"];
        $response = $snmp->getBulkRequest($varbindlist, $nonrep, $maxrep);

}
```

Figura 4.24: Segmento de código de la función *sendRequest*.

4.3.4 Iteración 4

Durante esta iteración se desarrolló el módulo del NMSWeb correspondiente al envío de notificaciones SNMP.

Por ello, se codificaron nuevas funciones *JavaScript* que, en conjunto con las desarrolladas en la iteración 3, permiten la validación de los campos de entrada definidos para esta área.

Al igual que en la sección anterior, en esta implementación fue necesario utilizar la clase *Open Source Autosuggest* para mostrar sugerencias en el campo de entrada correspondiente al identificador del objeto. Asimismo, se utilizó la clase *xajax* para indicar al usuario los resultados de la notificación enviada, sin ser necesaria la actualización de la página.

De esta manera se proporcionó soporte para el envío de las siguientes notificaciones SNMP, desde la interfaz del NMSWeb:

- Trap.
- Trapv2.
- InformRequest.

En la Figura 4.25 se observa la interfaz definitiva construida para esta área.

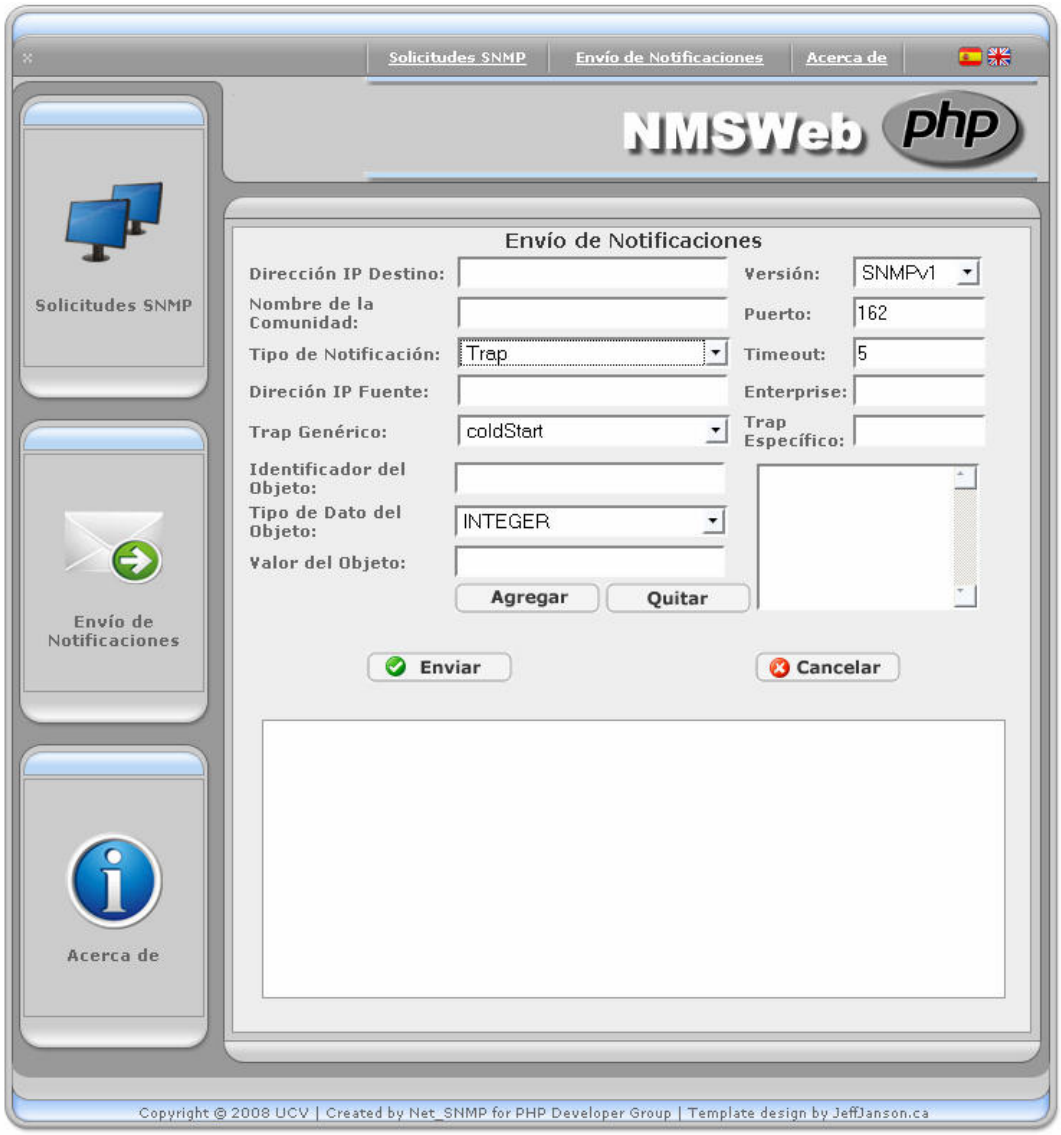

Figura 4.25: Interfaz de Notificaciones NMSWeb.

Para hacer posible el envío de las notificaciones, se creo una función PHP llamada "*sendNotify*", que se encarga de tomar los valores introducidos en el formulario y realizar el envío de la notificación seleccionada por el usuario, utilizando la biblioteca de funciones SNMP. Además, permite obtener y mostrar el resultado de la operación, mediante un objeto de la clase *xajax*.

En la Figura 4.26 se aprecia un segmento de código perteneciente a la función "*sendNotify*", en el que se ejecuta la operación para el envío de la notificación, de acuerdo a la elección realizada por el usuario en la interfaz web.

```
switch ($operation) {
    case 1:
        $name      = 'Trap';
        $enterprise = $formSNMP["enterprise"];
        $address = $formSNMP["IPFuente"];
        $generic = (int)$formSNMP["trapGeneric"];
        $specific = (int)$formSNMP["trapSpecific"];
        $timestamp = time() - $time_start;
        $response = $snmp->trap($enterprise, $address, $generic, $specific, $timestamp, $varbindlist);
        break;
    case 2:
        $name      = 'Trapv2';
        $response = $snmp->snmpv2Trap($varbindlist);
        break;
    case 3:
        $name      = 'InformRequest';
        $response = $snmp->informRequest($varbindlist);
        break;
}
```

Figura 4.26: Segmento de código de la función *sendNotify*.

4.3.5 Iteración 5

El propósito de esta iteración es proporcionar al NMSWeb la opción de cambio de idioma. De esta forma, es posible mostrar la información de la aplicación en idioma inglés o español, según la elección del usuario.

Para poder implementar esta funcionalidad, se construyeron dos archivos denominados *en.inc.php* y *es.inc.php*, con las definiciones de las etiquetas utilizadas para el idioma inglés y español respectivamente. Luego en cada archivo correspondiente a los módulos de la aplicación, se evalúa cual es el idioma que se está utilizando y de acuerdo a éste se incluye el archivo adecuado.

En la Figura 4.27 se observa un segmento del código utilizado para la inclusión de los archivos de idioma.

```
session_start();
if($_GET['lang']!="") {
    $lang=$_GET['lang'];
} else if ($_SESSION['lang']!="") {
    $lang=$_SESSION['lang'];
} else {
    $lang='es';
}
if ($lang=='es') {
    require_once('language/es.inc.php');
    $_SESSION['lang']='es';
} else if ($lang=='en') {
    require_once('language/en.inc.php');
    $_SESSION['lang']='en';
}
```

Figura 4.27: Segmento de código para la elección del idioma.

Finalmente, en cada uno de los símbolos empleados para el cambio de idioma, se proporciona un enlace a la página actual, pasando como argumento la cadena que identifica el lenguaje seleccionado por el usuario. La Figura 4.28 describe esta operación.

Figura 4.28: Ejemplo de enlaces para cambio de idioma.

4.3.6 Iteración 6

En esta iteración se inicia la implementación de la interfaz web para la administración de *routers Cisco Systems*, denominada RMSWeb. Cabe destacar que todas las áreas diseñadas para esta aplicación emplean el protocolo SNMP, a través de la biblioteca de funciones desarrollada durante las iteraciones iníciales.

Como primer paso, se diseñaron y codificaron las secciones que comprenden el área de herramientas. A continuación se describirán los módulos implementados.

- **Ping SNMP:** Para el desarrollo de esta área se utilizó una MIB privada, denominada CISCO-PING-MIB [17] que permite crear, ejecutar y recuperar mensajes ICMP (*Internet Control Message Protocol)*, entre dispositivos remotos, desde una estación de administración.

  En la Tabla 4.17 se describen los objetos SNMP manipulados para esta operación.

Objeto	Descripción
ciscoPingEntryStatus	Representa el estado de una secuencia ICMP en particular. Una vez que al objeto se le ha asignado el valor de activo, no podrá modificarse hasta que la secuencia se haya completado.
ciscoPingEntryOwner	Representa el nombre de la entidad que ha configurado una secuencia ICMP en particular.
ciscoPingProtocol	Identifica cual será el protocolo a utilizar.
ciscoPingAddress	Dirección IP del *host* al que se enviará la secuencia de mensajes ICMP.
ciscoPingPacketCount	Especifica el número de paquetes ICMP que

Objeto	Descripción
	serán enviados.
ciscoPingPacketSize	Representa el tamaño de los paquetes ICMP que serán enviados.
ciscoPingPacketTimeout	Indica la cantidad de tiempo máximo para esperar respuesta a una solicitud ICMP enviada.
ciscoPingDelay	Representa la cantidad de tiempo mínimo que se esperará antes de enviar la siguiente solicitud ICMP, después de recibir una respuesta.
ciscoPingCompleted	Permite determinar si la ejecución de la secuencia de solicitudes ICMP ha finalizado.
ciscoPingSentPackets	Número de solicitudes ICMP que han sido enviadas al *host* destino.
ciscoPingReceivedPackets	Número de respuestas a las solicitudes ICMP que fueron recibidas.
ciscoPingMinRtt	Tiempo mínimo de respuesta de todas las solicitudes enviadas durante la secuencia.
ciscoPingAvgRtt	Tiempo promedio de respuesta de todas las solicitudes enviadas durante la secuencia.
ciscoPingMaxRtt	Tiempo máximo de respuesta de todas las solicitudes enviadas durante la secuencia.

Tabla 4.17: Descripción de objetos de CISCO-PING-MIB.

Luego de determinar cuáles serían los campos de entrada requeridos por este módulo, se modificó el prototipo de interfaz obtenido durante la fase de diseño, como se observa en la Figura 4.29.

Figura 4.29: Interfaz de Ping SNMP RMSWeb.

Por último, para hacer posible la ejecución de la solicitud, se creó una función llamada "*sendPing*", que se encarga de tomar los parámetros de entrada proporcionados por el usuario y realizar las operaciones SNMP necesarias para el envío de los mensajes ICMP. Adicionalmente, esta función permite mostrar los resultados de la operación, haciendo uso de la clase *Open Source xajax*.

En la Figura 4.30 se aprecia un segmento de código perteneciente a la función "*sendPing*", en el que se llevan a cabo algunas de las solicitudes SNMP requeridas.

```
$varbindlist = new Net_SNMP_VarBindList();
$varbindlist->add(new Net_SNMP_VarBind(ciscoPingEntryStatus.'.'.$index, 5, INTEGER));
$varbindlist->add(new Net_SNMP_VarBind(ciscoPingEntryOwner.'.'.$index, $sysname, OCTET_STRING));
$varbindlist->add(new Net_SNMP_VarBind(ciscoPingProtocol.'.'.$index, 1, INTEGER));
$varbindlist->add(new Net_SNMP_VarBind(ciscoPingAddress.'.'.$index, $DirIP, OCTET_STRING));
$varbindlist->add(new Net_SNMP_VarBind(ciscoPingPacketCount.'.'.$index, $rep, INTEGER));
$varbindlist->add(new Net_SNMP_VarBind(ciscoPingPacketSize.'.'.$index, $size, INTEGER));
$varbindlist->add(new Net_SNMP_VarBind(ciscoPingPacketTimeout.'.'.$index, $time, INTEGER));
$varbindlist->add(new Net_SNMP_VarBind(ciscoPingDelay.'.'.$index, $delay, INTEGER));

$PDU = $snmp->setRequest($varbindlist);
```

Figura 4.30: Segmento de código de la función *sendPing*.

- **CISCO IOS:** Esta herramienta se desarrolló con el propósito de que el usuario pueda manipular el archivo del IOS alojado en el *router*. De esta manera, es posible realizar las siguientes acciones:

  - Copiar la imagen del IOS desde el *router*, hacia un servidor TFTP (*Trivial File Transfer Protocol*).
  - Cargar una imagen del IOS alojada en un servidor TFTP, hacia el *router*.

  Para implementar las funcionalidades mencionadas anteriormente, se utilizaron los objetos de la MIB privada OLD-CISCO-FLASH-MIB [18], que se describen en la Tabla 4.18.

Objeto	Descripción
flashToNet	Permite copiar el archivo del IOS al servidor TFTP. El valor del objeto debe ser el nombre del archivo a copiar y la instancia creada debe ser la dirección IP del servidor TFTP.
netToFlash	Permite copiar un archivo IOS alojado en un servidor TFTP, hacia el *router*. El valor del objeto debe ser el nombre del archivo a copiar y la instancia creada debe ser la dirección IP del servidor TFTP.

Tabla 4.18: Descripción de objetos de OLD-CISCO-FLASH-MIB.

Con el fin de poder manipular estos objetos, se codificaron dos métodos:

- "*downloadIOS*": A través de este método se envía la solicitud SNMP necesaria, para copiar el archivo desde el *router* hacia el servidor TFTP especificado por el usuario. En la Figura 4.31 se aprecia un segmento de código perteneciente a esta función.

```
$server     = $formSNMP["TFTPServer"];
$filename   = $formSNMP["Filename"];

$varbindlist = new Net_SNMP_VarBindList();
$varbindlist->add(new Net_SNMP_VarBind(flashToNet.'.'.$server, $filename, OCTET_STRING));

$PDU        = $snmp->setRequest($varbindlist);
```

Figura 4.31: Segmento de código de la función *downloadIOS*.

- "*uploadIOS*": Desde esta función es posible modificar el objeto netToFlash, permitiendo así cargar el archivo alojado en un servidor TFTP hacia el *router*, tal y como se observa en la Figura 4.32.

```
$server     = $formSNMP["TFTPServer"];
$filename   = $formSNMP["Filename"];

$varbindlist = new Net_SNMP_VarBindList();
$varbindlist->add(new Net_SNMP_VarBind(netToFlash.'.'.$server, $filename, OCTET_STRING));

$PDU        = $snmp->setRequest($varbindlist);
```

Figura 4.32: Segmento de código de la función *uploadIOS*.

- **Config File:** Durante la implementación de esta herramienta se utilizó una MIB privada, con la cual es posible copiar la configuración de un *router Cisco.*

  Los objetos de la MIB CISCO-CONFIG-COPY-MIB [19], que fue necesario manipular para el desarrollo de esta área, se describen en la Tabla 4.19.

Objeto	Descripción
ccCopyProtocol	Indica cual será el protocolo utilizado para realizar la copia del archivo de configuración.
ccCopySourceFileType	Especifica el tipo de archivo a copiar desde el *router*.
ccCopyDestFileType	Especifica el tipo de archivo a copiar hacia el *router*.
ccCopyServerAddress	Representa la dirección IP del servidor TFTP desde o hacia el cual se copiará el archivo de configuración.
ccCopyFileName	Indica el nombre del archivo a copiar.
ccCopyEntryRowStatus	Representa el estado de la solicitud creada en la tabla.

Tabla 4.19: Descripción de objetos de CISCO-CONFIG-MIB.

Para poder realizar las operaciones SNMP relacionadas con los datos de entrada proporcionados por el usuario, se construyeron las siguientes funciones:

- "*uploadConfig*": Permite cargar un archivo de configuración alojado en un servidor TFTP hacia el *router*. En la Figura 4.33 se aprecia un ejemplo de las operaciones implementadas desde esta función.

```
$DirIP     = $formSNMP["TFTPServer"];
$file      = $formSNMP["Filename"];
$index     = rand(1,65535);

$varbindlist = new Net_SNMP_VarBindList();
$varbindlist->add(new Net_SNMP_VarBind(ccCopyProtocol.'.'.$index, 1, INTEGER));
$varbindlist->add(new Net_SNMP_VarBind(ccCopySourceFileType.'.'.$index, 1, INTEGER));
$varbindlist->add(new Net_SNMP_VarBind(ccCopyDestFileType.'.'.$index, 3, INTEGER));
$varbindlist->add(new Net_SNMP_VarBind(ccCopyServerAddress.'.'.$index, $DirIP, IPADDRESS));
$varbindlist->add(new Net_SNMP_VarBind(ccCopyFileName.'.'.$index, $file, OCTET_STRING));
$varbindlist->add(new Net_SNMP_VarBind(ccCopyEntryRowStatus.'.'.$index, 4, INTEGER));
$PDU       = $snmp->setRequest($varbindlist);
```

Figura 4.33: Segmento de código de la función *uploadConfig*.

- "*downloadConfig*": A través de esta operación es posible realizar las solicitudes SNMP requeridas para copiar el archivo de configuración de un *router Cisco* hacia un servidor TFTP. En la Figura 4.34 se observa un segmento de código perteneciente a esta función.

```
$DirIP     = $formSNMP["TFTPServer"];
$file      = $formSNMP["Filename"];
$index     = rand(1,65535);

$varbindlist = new Net_SNMP_VarBindList();
$varbindlist->add(new Net_SNMP_VarBind(ccCopyProtocol.'.'.$index, 1, INTEGER));
$varbindlist->add(new Net_SNMP_VarBind(ccCopySourceFileType.'.'.$index, 4, INTEGER));
$varbindlist->add(new Net_SNMP_VarBind(ccCopyDestFileType.'.'.$index, 1, INTEGER));
$varbindlist->add(new Net_SNMP_VarBind(ccCopyServerAddress.'.'.$index, $DirIP, IPADDRESS));
$varbindlist->add(new Net_SNMP_VarBind(ccCopyFileName.'.'.$index, $file, OCTET_STRING));
$varbindlist->add(new Net_SNMP_VarBind(ccCopyEntryRowStatus.'.'.$index, 4, INTEGER));
$PDU       = $snmp->setRequest($varbindlist);
```

Figura 4.34: Segmento de código de la función *downloadConfig*.

4.3.7 Iteración 7

Durante esta iteración se diseñaron y codificaron las secciones que forman parte del área de Detalles del RMSWeb. En esta área se muestra información relativa al *router*, recuperada a través de consultas a diferentes objetos SNMP.

A continuación se describirán cada una de las secciones implementadas.

- **General:** Desde esta sección, es posible apreciar los datos generales del *router*, como se observa en la Figura 4.35.

Nombre del Sistema	RouterDeLasChicas.ciens.ucv.ve
Descripción	Cisco IOS Software, 2800 Software (C2800NM-ADVIPSERVICESK9-M), Version 12.4(13a), RELEASE SOFTWARE (fc1) Technical Support: http://www.cisco.com/techsupport Copyright (c) 1986-2007 by Cisco Systems, Inc. Compiled Tue 06-Mar-07 17:01 by prod_rel_team
Dirección IP	190.169.253.19
Nombre DNS	190.169.253.19
Localización	internet2
Contacto	Eric Gamess
Tiempo Activo	1 dia, 17:39:45.1

Figura 4.35: Área de datos generales en RMSWeb.

Estos datos se recuperan a través de consultas a objetos pertenecientes a la MIB II, definida en el RFC 1213 [8]. En la Tabla 4.20 se aprecia la descripción de alguno de los objetos consultados.

Objeto	Descripción
sysName	Nombre del sistema.
sysDescr	Descripción del sistema.
sysLocation	Localización del dispositivo.
sysContact	Persona responsable por el dispositivo.
sysUpTime	Tiempo que lleva activo el dispositivo.

Tabla 4.20: Descripción de objetos del grupo *system* de la MIB II.

Para realizar las solicitudes SNMP respectivas, se construyó una función denominada "*getGeneral*", que se encarga de consultar los objetos SNMP y proporcionar los resultados al usuario, por medio de la clase *xajax*. En la Figura 4.36 se aprecia un segmento de código perteneciente a esta función.

```
$session = $snmp->session($ip, SNMP_PORT, $readCommunity, SNMP_V2C);
if (!$session) {
return clearGeneral();
} else {
    $varbindlist = new Net_SNMP_VarBindList();
    $varbindlist->add(new Net_SNMP_VarBind(sysDescr));
    $varbindlist->add(new Net_SNMP_VarBind(sysLocation));
    $varbindlist->add(new Net_SNMP_VarBind(sysContact));
    $varbindlist->add(new Net_SNMP_VarBind(sysUpTime));
    $PDU = $snmp->getRequest($varbindlist);
```

Figura 4.36: Segmento de código de la función *getGeneral*.

- **Chasis:** Esta área muestra información relacionada al chasis del *router*, como se aprecia en la Figura 4.37.

Modelo	413
Revisión-Hardware	3.0
ID del Chasis	FTX0937A1BG
Versión-ROM	System Bootstrap, Version 12.3(8r)T7, RELEASE SOFTWARE (fc1) Technical Support: http://www.cisco.com/techsupport Copyright (c) 2004 by cisco Systems, Inc.
RAM	242 MB
RAM no volátil	0.23 MB Total 0.23 MB Usados 0 MB Libres
Registro de Configuración	8450

Figura 4.37: Área de datos de Chasis en RMSWeb.

Los datos son obtenidos a través de consultas SNMP a objetos pertenecientes a la MIB privada OLD-CISCO-CHASSIS-MIB [20]. La descripción de estos objetos se observa en la Tabla 4.21.

Objeto	Descripción
chassisType	Indica cual es el modelo o tipo del chasis.
chassisVersion	Indica cual es la versión del chasis, a nivel de hardware.
chassisId	Cadena con el identificador único del *router*. Por defecto el valor es igual al serial del chasis.
romVersion	Indica cual es la versión de ROM (*Read Only Memory*) del *router*.
processorRam	Número de *bytes* de RAM (*Random Access Memory*) disponible para el CPU.
nvRAMSize	Número de *bytes* de la memoria no volátil.
nvRAMUsed	Número de *bytes* en uso de la memoria no volátil.
configRegister	Valor del registro de configuración.

Tabla 4.21: Descripción de objetos de OLD-CISCO-CHASSIS-MIB.

Para la implementación de esta área, fue necesario codificar una función llamada "*getChassis*" que realiza las solicitudes SNMP necesarias y permite mostrar los resultados obtenidos al usuario.

La Figura 4.38 contiene un segmento del código del método "*getChassis*".

```
$session = $snmp->session($ip, SNMP_PORT, $readCommunity, SNMP_V2C);
if (!$session) {
return clearChassis();
} else {
    $varbindlist = new Net_SNMP_VarBindList();
    $varbindlist->add(new Net_SNMP_VarBind(chassisType.'.0'));
    $varbindlist->add(new Net_SNMP_VarBind(chassisVersion.'.0'));
    $varbindlist->add(new Net_SNMP_VarBind(chassisId.'.0'));
    $varbindlist->add(new Net_SNMP_VarBind(romVersion.'.0'));
    $varbindlist->add(new Net_SNMP_VarBind(processorRam.'.0'));
    $varbindlist->add(new Net_SNMP_VarBind(nvRAMSize.'.0'));
    $varbindlist->add(new Net_SNMP_VarBind(nvRAMUsed.'.0'));
    $varbindlist->add(new Net_SNMP_VarBind(configRegister.'.0'));
    $PDU = $snmp->getRequest($varbindlist);
```

Figura 4.38: Segmento de código de la función *getChassis*.

- **Flash:** Desde esta sección es posible apreciar la información que se observa en la Figura 4.39.

Tipo de Controlador	CompactFlash
Tipo de Tarjeta	CompactFlash
Memoria Flash	61.05 MB Total 34.61 MB Usados 26.44 MB Libres

Figura 4.39: Área de datos de Flash en RMSWeb.

Para ello, fue necesario consultar los objetos que se describen en la Tabla 4.22, los cuales pertenecen a la MIB privada OLD-CISCO-FLASH-MIB [18].

Objeto	Descripción
flashController	Proporciona el tipo de controlador del Flash instalado en el *router*.
flashCard	Proporciona el tipo de tarjeta Flash, instalada en el *router*.
flashSize	Tamaño total en octetos, de la memoria Flash.
flashFree	Cantidad de octetos sin usar en la memoria Flash.

Tabla 4.22: Descripción de objetos de OLD-CISCO-FLASH-MIB.

Con el propósito de obtener los datos anteriores, se implementó un método llamado "*getFlash*", a través del cual se realizan las consultas SNMP necesarias, y se muestran los resultados recuperados.

En la Figura 4.40 se tiene un segmento de código perteneciente a esta función, en el que se observa la ejecución de la solicitud GetRequest.

```
$varbindlist = new Net_SNMP_VarBindList();
$varbindlist->add(new Net_SNMP_VarBind(flashController.'.0'));
$varbindlist->add(new Net_SNMP_VarBind(flashCard.'.0'));
$varbindlist->add(new Net_SNMP_VarBind(flashSize.'.0'));
$varbindlist->add(new Net_SNMP_VarBind(flashFree.'.0'));
$PDU = $snmp->getRequest($varbindlist);
```

Figura 4.40: Segmento de código de la función *getFlash*.

- **IOS:** Esta área proporciona información relacionada al sistema, realizando consultas a objetos que forman parte de la MIBs privadas OLD-CISCO-FLASH-MIB [18] y OLD-CISCO-CHASSIS-MIB [20], respectivamente.

  Los datos consultados se observan en la Figura 4.41.

ROM - Versión del Sistema	Cisco IOS Software, 2800 Software (C2800NM-ADVIPSERVICESK9-M), Version 12.4(13a), RELEASE SOFTWARE (fc1) Technical Support: http://www.cisco.com/techsupport Copyright (c) 1986-2007 by Cisco Systems, Inc. Compiled Tue 06-Mar-07 17:01 by prod_rel_team
Archivo de la Imagen	c2800nm-advipservicesk9-mz.124-13a.bin
Tamaño de la Imagen	34.6 MB

Figura 4.41: Área de datos de IOS en RMSWeb.

En la Tabla 4.23 se describen los objetos SNMP utilizados.

Objeto	MIB	Descripción
romSysVersion	OLD-CISCO-CHASSIS-MIB	Versión del software del sistema.
flashDirName	OLD-CISCO-FLASH-MIB	Nombre de la imagen del IOS que reside en la memoria Flash.
flashDirSize	OLD-CISCO-FLASH-MIB	Tamaño en octetos de la imagen del IOS alojada en la memoria Flash.

Tabla 4.23: Descripción de objetos para el área IOS en RMSWeb.

Para implementar esta sección se codificó una función llamada "*getIOS*", responsable de realizar las solicitudes SNMP necesarias, y mostrar el resultado en la interfaz de usuario. En la Figura 4.42, se aprecia un segmento de código perteneciente a este método.

```
$varbindlist = new Net_SNMP_VarBindList();
$varbindlist->add(new Net_SNMP_VarBind(romSysVersion.'.0'));
$varbindlist->add(new Net_SNMP_VarBind(flashDirName));
$varbindlist->add(new Net_SNMP_VarBind(flashDirSize));
$PDU = $snmp->getRequest($varbindlist);
```

Figura 4.42: Segmento de código de la función *getIOS*.

- **IP Routes:** Para el desarrollo de esta sección se consultaron valores pertenecientes a la MIB II definida en el RFC 1213 [8]. En la Figura 4.43 se puede apreciar la información proporcionada.

Destino	Interface	Próximo Salto	Tipo	Protocolo	Edad
0.0.0.0	0	190.169.253.1	4	2	8

Figura 4.43: Área de IP Routes en RMSWeb.

Adicionalmente, en la Tabla 4.24 se detallan los objetos SNMP consultados durante esta operación.

Objeto	Descripción
ipRouteDest	Dirección IP destino de la ruta.
ipRouteIfIndex	Valor que identifica inequívocamente a la interfaz local a través de la cual se dará el próximo salto.
ipRouteNextHop	Dirección IP del próximo salto en la ruta.
ipRouteType	Indica el tipo de la ruta.
ipRouteProto	Indica el mecanismo de enrutamiento a través del cual la ruta es aprendida.
ipRouteAge	Número de segundos transcurridos desde la última actualización de la ruta.

Tabla 4.24: Descripción de objetos para el área IP Routes en RMSWeb.

Los valores anteriores son recuperados por medio de solicitudes SNMP sucesivas ejecutadas por el método "*getIPRoutes*", como se observa en la Figura 4.44.

```
while (!$end) {
    if (!$PDU) {
        return clearTable(6);
    } else {
        $j = 0;
        $list = $PDU->getVarBindList();
        $oid = $list[0]->getObjectIdentifier();
        if (substr($oid, 0, 20) != ipRouteDest) {
            $end = true;
        } else {
            $varbindlist = new Net_SNMP_VarBindList();
            foreach ($list as $var_bind) {
                $varbindlist->add(new Net_SNMP_VarBind($var_bind->getObjectIdentifier()));
                $table[$i][$j] = $var_bind->getValue();
                $j++;
            }
            $PDU = $snmp->getNextRequest($varbindlist);
            $i++;
        }
    }
```

Figura 4.44: Segmento de código de la función *getIPRoutes*.

- **Interfaces:** El objetivo de esta área es proporcionar información relacionada a las interfaces del *router*.

  Para ello, se realizaron consultas SNMP a objetos pertenecientes a la MIB II definida en el RFC 1213 [8]. Una descripción de estos objetos se tiene en la Tabla 4.25.

Objeto	Descripción
ifDescr	Proporciona información relacionada con la interfaz.
ifType	Indica el tipo de interfaz.
ifSpeed	Estimado del ancho de banda actual de la interfaz, en bits por segundo.

Objeto	Descripción
ifMtu	Tamaño máximo del datagrama que puede ser enviado o recibido a través de la interfaz, expresado en octetos.
ifOperStatus	Indica el estado operacional de la interfaz.
ifAdminStatus	Indica el estado deseado para la interfaz.

Tabla 4.25: Descripción de los objetos para el área Interfaces en RMSWeb.

Durante la implementación se codificó una función denominada "*getInterfaces*", encargada de realizar las solicitudes SNMP necesarias para obtener los datos y mostrar la información recopilada en la interfaz. En la Figura 4.45 se tiene un segmento de código perteneciente a este método.

```
$varbindlist = new Net_SNMP_VarBindList();
$varbindlist->add(new Net_SNMP_VarBind(ifDescr));
$varbindlist->add(new Net_SNMP_VarBind(ifType));
$varbindlist->add(new Net_SNMP_VarBind(ifPhysAddress));
$varbindlist->add(new Net_SNMP_VarBind(ifSpeed));
$varbindlist->add(new Net_SNMP_VarBind(ifMtu));
$varbindlist->add(new Net_SNMP_VarBind(ifOperStatus));
$varbindlist->add(new Net_SNMP_VarBind(ifAdminStatus));
$PDU = $snmp->getNextRequest($varbindlist);
```

Figura 4.45: Segmento de código de la función *getInterfaces*.

- **VoIP:** Desde esta sección es posible apreciar información relacionada con VoIP, tal y como se observa en la Figura 4.46.

Figura 4.46: Área de VoIP en RMSWeb.

Para recuperar esta información, se consultaron objetos pertenecientes a las siguientes MIBs privadas de *Cisco Systems*:

- CISCO-VOICE-ANALOG-IF-MIB [21].
- CISCO-VOICE-IF-MIB [22].
- CISCO-VOICE-COMMON-DIAL-CONTROL-MIB [23].

En la Tabla 4.26 se observa la descripción del grupo de objetos consultados.

Objeto	MIB	Descripción
cvaIfFXSCfgSignalType	CISCO-VOICE-ANALOG-IF-MIB	Indica el tipo de señalización de la interfaz FXS (fxsLoopStart o fxsGroundStart).
cvaIfFXSHookStatus	CISCO-VOICE-ANALOG-IF-MIB	Indica el estado de la interfaz FXS. (onHook, offHook o trunked).
cvIfCfgEchoCancelEnable	CISCO-VOICE-IF-MIB	Indica si *Echo Cancellation* está habilitado en la interfaz.
cvCommonDcCallActiveCoderTypeRate	CISCO-VOICE-COMMON-DIAL-CONTROL-MIB	Especifica la tasa de transmisión asociada a la llamada.
cvCommonDcCallActiveVADEnable	CISCO-VOICE-COMMON-DIAL-CONTROL-MIB	Indica si VAD (*Voice Activity Detection*) se encuentra habilitado para la llamada de voz.

Tabla 4.26: Descripción de los objetos para el área VoIP en RMSWeb.

Durante la implementación, se codificaron dos funciones denominadas "*getVoicePort*" y "*getVoiceCall*", responsables de realizar las consultas pertenecientes al área de resumen de puertos de voz y resumen de llamadas de voz, respectivamente. En la Figura 4.47 se observa un segmento de código en el que se realizan solicitudes SNMP, por parte del método "*getVoiceCall*".

```
$varbindlist = new Net_SNMP_VarBindList();
$varbindlist->add(new Net_SNMP_VarBind(cvCommonDcCallActiveCoderTypeRate.'.1102966'));
$varbindlist->add(new Net_SNMP_VarBind(cvCommonDcCallActiveVADEnable.'.1102966'));
$PDU = $snmp->getNextRequest($varbindlist);
```

Figura 4.47: Segmento de código de la función *getVoiceCall*.

- **CDP:** Esta área proporciona información de la configuración actual del protocolo CDP en el *router*.

  Para la implementación, los datos se dividieron en tres secciones:

  - General: Permite conocer el estado general del protocolo en el *router*, a través de los campos que se observan en la Figura 4.48.

Ejecución CDP	Sí
Intervalo del Mensaje	150
Tiempo mant. Mensaje	240

Figura 4.48: Área de CDP General en RMSWeb.

Para ello, se codificó una función llamada "*getGeneralCDP*" que realiza las solicitudes SNMP necesarias, como se observa en la Figura 4.49.

```
$varbindlist = new Net_SNMP_VarBindList();
$varbindlist->add(new Net_SNMP_VarBind(cdpGlobalRun.'.0'));
$varbindlist->add(new Net_SNMP_VarBind(cdpGlobalMessageInterval.'.0'));
$varbindlist->add(new Net_SNMP_VarBind(cdpGlobalHoldTime.'.0'));
$PDU = $snmp->getRequest($varbindlist);
```

Figura 4.49: Segmento de código de la función getGeneralCDP.

- Interface Group: Proporciona información sobre el estado del protocolo CDP en cada una de las interfaces. En la Figura 4.50 se muestra un ejemplo de los datos mostrados.

Interfaz	Habilitado	Grupo	Puerto
FastEthernet0/0	No	0	0
FastEthernet0/1	Sí	0	0
Serial0/0/0	Sí	0	0
Serial0/0/1	Sí	0	0

Figura 4.50: Área de Interface Group CDP en RMSWeb.

Para recuperar estos datos, se codificó un método denominado "*getInterfaceGroup*", responsable de realizar las solicitudes SNMP. En la Figura 4.51 se observa un segmento de código perteneciente a esta función.

```
$varbindlist = new Net_SNMP_VarBindList();
$varbindlist->add(new Net_SNMP_VarBind(cdpInterfaceIfIndex));
$varbindlist->add(new Net_SNMP_VarBind(cdpInterfaceEnable));
$varbindlist->add(new Net_SNMP_VarBind(cdpInterfaceGroup));
$varbindlist->add(new Net_SNMP_VarBind(cdpInterfacePort));
$PDU = $snmp->getNextRequest($varbindlist);
```

Figura 4.51: Segmento de código de la función *getInterfaceGroup*.

- Cache: Desde la sección General Cache y Address Cache, es posible recuperar los datos de los sistemas descubiertos por el *router*, a través del protocolo CDP.

  Con este propósito, se codificó la función "*getCache*" que realiza las operaciones necesarias para obtener la información mostrada en esta área. En la Figura 4.52 se aprecia un ejemplo de las solicitudes realizadas desde este método.

```
$varbindlist = new Net_SNMP_VarBindList();
$varbindlist->add(new Net_SNMP_VarBind(cdpCacheAddressType));
$varbindlist->add(new Net_SNMP_VarBind(cdpCacheAddress));
$varbindlist->add(new Net_SNMP_VarBind(cdpCacheVersion));
$varbindlist->add(new Net_SNMP_VarBind(cdpCacheDevicePort));
$varbindlist->add(new Net_SNMP_VarBind(cdpCachePlatform));
$varbindlist->add(new Net_SNMP_VarBind(cdpCacheCapabilities));
$varbindlist->add(new Net_SNMP_VarBind(cdpCacheSysName));
$PDU = $snmp->getNextRequest($varbindlist);
```

Figura 4.52: Segmento de código de la función *getCache*.

En las solicitudes descritas anteriormente, se consultaron los objetos de la MIB privada CISCO-CDP-MIB [24], que se observan en la Tabla 4.27.

Objeto	Descripción
cdpGlobalRun	Indica si el protocolo CDP se encuentra actualmente en ejecución en el *router*.
cdpGlobalMessageInterval	Intervalo de tiempo en el cual los mensajes CDP son generados.
cdpGlobalHoldTime	Tiempo para la recepción de mensajes CDP.
cdpInterfaceIfIndex	Valor ifIndex de la interfaz local.
cdpInterfaceEnable	Indica si el protocolo CDP se encuentra actualmente activo en la interfaz.
cdpInterfaceGroup	Indica el número de grupo que corresponde a la interfaz.
cdpInterfacePort	Número de puerto que responde a la interfaz.
cdpCacheAddressType	Indica el tipo de dirección que contiene la instancia de cdpCacheAddress.
cdpCacheAddress	Representa la primera dirección del dispositivo.
cdpCacheVersion	Versión del dispositivo.
cdpCacheDevicePort	Identificador del puerto, recibido en el mensaje CDP más reciente.
cdpCachePlatform	Proporciona información de la plataforma de hardware del dispositivo.
cdpCacheCapabilities	Indica las capacidades funcionales del dispositivo.
cdpCacheSysName	Valor del objeto sysName en el dispositivo remoto.

Tabla 4.27: Descripción de objetos de CISCO-CDP-MIB.

- **VPDN:** La información proporcionada en esta área es extraída de la MIB privada CISCO-VPDN-MGMT-MIB [25]. Los objetos consultados se describen en la Tabla 4.28.

Objeto	Descripción
cvpdnSystemTunnelType	Representa el tipo del túnel.
cvpdnSystemTunnelTotal	Número total de túneles VPDN que se encuentran actualmente activos.
cvpdnSystemSessionTotal	Número total de secciones activas en los túneles VPDN.
cvpdnSystemDeniedUsersTotal	Número total de usuarios rechazados en los túneles VPDN desde la última reinicialización del sistema.
cvpdnTunnelTunnelId	Identificador del túnel VPDN activo.
cvpdnTunnelRemoteTunnelId	Identificador del túnel VPDN remoto.
cvpdnTunnelLocalName	Nombre del túnel VPDN local activo.

Objeto	Descripción
cvpdnTunnelLocalInitConnection	Nombre del túnel VPDN remoto activo.
cvpdnTunnelState	Estado actual del túnel VPDN activo.
cvpdnTunnelActiveSessions	Número total de sesiones actualmente activas en el túnel.
cvpdnTunnelDeniedUsers	Total de usuarios rechazados en el túnel.
cvpdnTunnelLocalIpAddress	Dirección IP local del túnel.
cvpdnTunnelSourceIpAddress	Dirección IP origen del túnel.
cvpdnTunnelRemoteIpAddress	Dirección IP remota del túnel.
cvpdnTunnelSessionId	Identificador de la sesión de usuario activa en el túnel VPDN.
cvpdnTunnelSessionUserName	Nombre del usuario de la sesión activa.
cvpdnTunnelSessionState	Estado actual de la sesión de usuario activa.
cvpdnTunnelSessionDeviceCallerId	Identificador de las llamadas entrantes del usuario.
cvpdnTunnelSessionCallDuration	Duración de la llamada de la sesión actual de usuario activa.
cvpdnUnameToFailHistUname	Nombre del usuario que ocasionó la falla.
cvpdnUnameToFailHistUserId	Identificador del usuario que ocasionó la falla.
cvpdnUnameToFailHistTunnelId	Identificador del túnel en el que ocurrió la falla.
cvpdnUnameToFailHistLocalName	Nombre del túnel VPDN local en el que ocurrió la falla.
cvpdnUnameToFailHistRemoteName	Nombre del túnel VPDN remoto en el que ocurrió la falla.
cvpdnUnameToFailHistSourceIp	Dirección IP origen del túnel en el que ocurrió la falla.
cvpdnUnameToFailHistDestIp	Dirección IP destino del túnel en el que ocurrió la falla.
cvpdnUnameToFailHistFailTime	Indica el momento en el cual la falla ocurrió.
cvpdnUnameToFailHistFailType	Identifica el tipo de falla.
cvpdnUnameToFailHistFailReason	Motivo que ocasionó la falla.

Tabla 4.28: Descripción de objetos de CISCO-VPDN-MGMT-MIB.

Para llevar a cabo las solicitudes SNMP y mostrar los resultados en la interfaz de usuario, fue necesario codificar cuatro funciones llamadas "*getVpdn*", "*getActiveTunnels"*, "*getSessions"*, y "*getFailures"*.

En la Figura 4.53 se observa un segmento de código de las operaciones realizadas por el método "*getVpdn*".

```
$varbindlist = new Net_SNMP_VarBindList();
$varbindlist->add(new Net_SNMP_VarBind(cvpdnSystemTunnelType));
$varbindlist->add(new Net_SNMP_VarBind(cvpdnSystemTunnelTotal));
$varbindlist->add(new Net_SNMP_VarBind(cvpdnSystemSessionTotal));
$varbindlist->add(new Net_SNMP_VarBind(cvpdnSystemDeniedUsersTotal));
$PDU = $snmp->getNextRequest($varbindlist);
```

Figura 4.53: Segmento de código de la función *getVpdn*.

4.3.8 Iteración 8

En esta iteración se implementaron módulos que permiten monitorear y graficar diversos objetos asociados al funcionamiento y rendimiento del *router*.

Para construir las gráficas, se utilizó la herramienta *Open Flash Chart*. A continuación se describen las áreas desarrolladas.

- **Monitor CPU:** Desde esta área es posible graficar el porcentaje de uso de CPU del *router*. Para ello, se consulta un objeto perteneciente a la MIB privada OLD-CISCO-CPU-MIB [26]. En la Tabla 4.29 se observa la descripción del objeto.

Objeto	Descripción
busyPer	Porcentaje de uso de CPU en el último período de cinco segundos.

Tabla 4.29: Descripción de objetos de OLD-CISCO-CPU-MIB.

Durante la implementación se codificó un archivo llamado "*data_cpu.php*", en el que se realiza la solicitud SNMP correspondiente para poder obtener el valor del objeto y almacenarlo en una estructura conveniente, que es utilizada por la herramienta *Open Flash Chart* para construir el gráfico.

En la Figura 4.54 se tiene un ejemplo de las operaciones definidas en el archivo "*data_cpu.php*".

```
$varbindlist = new Net_SNMP_VarBindList();
$varbindlist->add(new Net_SNMP_VarBind(busyPer.'.0'));
$PDU = $snmp->getRequest($varbindlist);
if (!$PDU) {
    $cpu             = array();
    $title           = $error;
    $_SESSION['cpu'] = '';
} else {
    $list    = $PDU->getVarBindList();
    $now = getdate();
    array_push($cpu, $list[0]->getValue());
```

Figura 4.54: Segmento de código de archivo data_cpu.php.

- **Monitor Memoria:** En esta sección se consultan y grafican valores de objetos, pertenecientes a la MIB privada CISCO-MEMORY-POOL-MIB [27].

  En la Tabla 4.30 se aprecia la descripción de los objetos SNMP solicitados.

Objeto	Descripción
ciscoMemoryPoolName	Nombre asignado al repositorio de memoria.
ciscoMemoryPoolUsed	Número de bytes de la memoria que están siendo utilizados por aplicaciones en el dispositivo.
ciscoMemoryPoolFree	Indica el número de bytes libres en la memoria.

Tabla 4.30: Descripción de los objetos de CISCO-MEMORY-POOL-MIB.

Para poder calcular el porcentaje de memoria en uso que será graficado para cada *pool*, se utiliza la fórmula que se observa en la Figura 4.55.

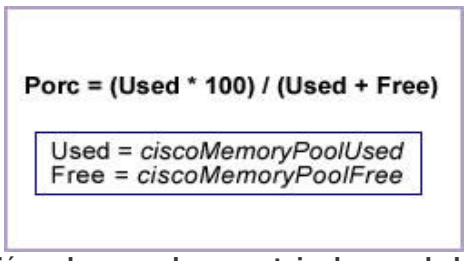

Figura 4.55: Fórmula para el porcentaje de uso de la memoria.

Estas operaciones son realizadas desde el archivo "*data_memory.php*", que realiza las consultas SNMP y proporciona los valores necesarios a la herramienta *Open Flash Chart* para generar el gráfico. En la Figura 4.56 se tiene un ejemplo de las instrucciones definidas desde este archivo.

```
$varbindlist = new Net_SNMP_VarBindList();
$varbindlist->add(new Net_SNMP_VarBind(ciscoMemoryPoolName.'.1'));
$varbindlist->add(new Net_SNMP_VarBind(ciscoMemoryPoolUsed.'.1'));
$varbindlist->add(new Net_SNMP_VarBind(ciscoMemoryPoolFree.'.1'));
$varbindlist->add(new Net_SNMP_VarBind(ciscoMemoryPoolName.'.2'));
$varbindlist->add(new Net_SNMP_VarBind(ciscoMemoryPoolUsed.'.2'));
$varbindlist->add(new Net_SNMP_VarBind(ciscoMemoryPoolFree.'.2'));
$PDU = $snmp->getRequest($varbindlist);
```

Figura 4.56: Segmento de código del archivo *data_memory.php*.

- **Monitor Interfaces:** Esta área permite monitorear el uso de entrada y salida de una interfaz del *router* seleccionada por el usuario. Para su implementación, se consultaron los valores de la MIB II [8] que se describen en la Tabla 4.31.

Objeto	Descripción
ifDescr	Proporciona información relacionada con la interfaz.
ifInOctets	Número total de octetos recibidos por la interfaz.
ifOutOctets	Número total de octetos transmitidos desde la interfaz.
ifSpeed	Estimado del ancho de banda actual de la interfaz, en bits por segundo.

Tabla 4.31: Descripción de los objetos consultados para el monitor de interfaces.

Asimismo, se desarrollaron dos archivos denominados "*data_interfaceInput.php*" y "*data_interfaceOutput.php*", responsables de calcular el porcentaje de uso de entrada y salida de la interfaz, respectivamente. La fórmula empleada para obtener los resultados a graficar se aprecia en la Figura 4.57.

$$\text{Input Utilization} = \frac{\triangle \text{ifInOctets} * 8 * 100}{\text{Nro. de Seg.} * \text{ifSpeed}}$$

$$\text{Output Utilization} = \frac{\triangle \text{ifOutOctets} * 8 * 100}{\text{Nro. de Seg.} * \text{ifSpeed}}$$

Figura 4.57: Fórmula para el porcentaje de uso de las interfaces.

Adicionalmente, en la Figura 4.58 se observa un ejemplo de las operaciones definidas en el archivo "*data_interfaceInput.php*".

```
$varbindlist = new Net_SNMP_VarBindList();
$varbindlist->add(new Net_SNMP_VarBind(ifDescr.'.'.$interface));
$varbindlist->add(new Net_SNMP_VarBind(ifInOctets.'.'.$interface));
$varbindlist->add(new Net_SNMP_VarBind(ifSpeed.'.'.$interface));
$PDU = $snmp->getRequest($varbindlist);
```

Figura 4.58: Segmento de código del archivo *data_interfaceInput.php*.

4.3.9 Iteración 9

Durante esta iteración se desarrollaron las siguientes secciones:

- **Configuración de Eventos RMON:** A través de esta área es posible crear nuevos eventos RMON para el *router*, haciendo uso de la MIB RMON definida en el RFC 1271 [28].

  Los objetos que pueden ser creados desde la interfaz se describen en la Tabla 4.32.

Objeto	Descripción
eventStatus	Representa el estado del evento.
eventDescription	Descripción del evento.
eventType	Representa el tipo de evento (none, log, snmp-trap, log-and-trap).
eventCommunity	Especifica la comunidad SNMP utilizada en el envío de traps.
eventOwner	Nombre de la entidad que ha creado el evento.

Tabla 4.32: Descripción de los objetos de MIB RMON.

Para permitir añadir nuevos eventos, se codificó una función llamada "*saveEvent*", encargada de tomar los parámetros proporcionados por el usuario, y efectuar las solicitudes SNMP respectivas.

En la Figura 4.59 se aprecia un segmento de código perteneciente a la función "*saveEvent*".

```
$varbindlist = new Net_SNMP_VarBindList();
$varbindlist->add(new Net_SNMP_VarBind(eventStatus.'.'.$id, 2, INTEGER));
$varbindlist->add(new Net_SNMP_VarBind(eventDescription.'.'.$id, $Desc, OCTET_STRING));
$varbindlist->add(new Net_SNMP_VarBind(eventType.'.'.$id, $Type, INTEGER));
if ($Community != '') {
    $varbindlist->add(new Net_SNMP_VarBind(eventCommunity.'.'.$id, $Community, OCTET_STRING));
}
$varbindlist->add(new Net_SNMP_VarBind(eventOwner.'.'.$id, $Owner, OCTET_STRING));

$PDU = $snmp->setRequest($varbindlist);
```

Figura 4.59: Segmento de código de la función *saveEvent*.

- **Configuración de Alarmas RMON:** Proporciona una interfaz para que el usuario pueda añadir y configurar alarmas RMON en el *router*, a través de los objetos definidos en la MIB RMON [28].

  En la Tabla 4.33 se aprecia la descripción de los objetos SNMP creados.

Objeto	Descripción
alarmStatus	Representa el estado de la alarma.
alarmInterval	Intervalo en segundos en el que el valor del objeto será comparado con el límite mínimo y máximo.
alarmVariable	Identificador del objeto SNMP que será monitoreado.
alarmSampleType	Representa el tipo de alarma.
alarmStartupAlarm	Indica las acciones que harán que la alarma sea activada.
alarmRisingThreshold	Límite máximo alcanzado por el objeto SNMP.
alarmFallingThreshold	Límite mínimo del objeto SNMP.

Objeto	Descripción
alarmRisingEventIndex	Identificador del evento que se activará en caso de que el objeto SNMP sobrepase el límite máximo.
alarmFallingEventIndex	Identificador del evento que se activará en caso de que el valor del objeto SNMP sea menor o igual al límite mínimo.
alarmOwner	Nombre de la alarma que ha creado el evento.

Tabla 4.33: Descripción de los objetos para la creación de Alarmas.

Para crear instancias de lo objetos anteriores, fue necesario desarrollar un método responsable de tomar los valores proporcionados por el usuario y realizar las solicitudes SNMP.

En la Figura 4.60 se observa un ejemplo de las instrucciones codificadas en la función "*saveAlarm*".

```
if ($ingress == 2 and $egress == 2){
    $varbindlist->add(new Net_SNMP_VarBind(cnfCINetflowEnable.'.'.$interface, 0, INTEGER));
}else if ($ingress == 1 and $egress == 2){
    $varbindlist->add(new Net_SNMP_VarBind(cnfCINetflowEnable.'.'.$interface, 1, INTEGER));
}else if ($ingress == 2 and $egress == 1){
    $varbindlist->add(new Net_SNMP_VarBind(cnfCINetflowEnable.'.'.$interface, 2, INTEGER));
}else if ($ingress == 1 and $egress == 1){
    $varbindlist->add(new Net_SNMP_VarBind(cnfCINetflowEnable.'.'.$interface, 3, INTEGER));
}

$PDU = $snmp->setRequest($varbindlist);
```

Figura 4.60: Segmento de código de la función *saveAlarm*.

- **Configuración NetFlow:** Desde esta área es posible cambiar los valores de configuración del protocolo Cisco NetFlow en el *router*.

  Con este propósito, se realizan modificaciones sobre los objetos de la MIB privada CISCO-NETFLOW-MIB [29], que se observan en la Tabla 4.34.

Objeto	Descripción
cnfEIExportVersion	Representa la versión que se utilizará para la exportación de los datos.
cnfEICollectorStatus	Permite crear o eliminar entradas en la tabla cnfEICollectorTable.
cnfCINetflowEnable	Habilita la transmisión de datos relacionados al protocolo NetFlow en las interfaces.

Tabla 4.34: Descripción de los objetos de CISCO-NETFLOW-MIB.

  Para la implementación de esta funcionalidad se construyó un método de nombre "*sendNetFlow*", responsable de recuperar los valores proporcionados por el usuario y realizar las operaciones SNMP respectivas para modificar la configuración del protocolo NetFlow en el dispositivo.

En la Figura 4.61 se aprecia un ejemplo de las operaciones realizadas desde la función "*sendNetFlow*".

```
$varbindlist = new Net_SNMP_VarBindList();
$varbindlist->add(new Net_SNMP_VarBind(alarmStatus.'.'.$id, 2, INTEGER));
$varbindlist->add(new Net_SNMP_VarBind(alarmInterval.'.'.$id, $interval, INTEGER));
$varbindlist->add(new Net_SNMP_VarBind(alarmVariable.'.'.$id, $oid, OBJECT_IDENTIFIER));
$varbindlist->add(new Net_SNMP_VarBind(alarmSampleType.'.'.$id, $Type, INTEGER));
$varbindlist->add(new Net_SNMP_VarBind(alarmStartupAlarm.'.'.$id, $StartupAl, INTEGER));
$varbindlist->add(new Net_SNMP_VarBind(alarmRisingThreshold.'.'.$id, $RisingT , INTEGER));
$varbindlist->add(new Net_SNMP_VarBind(alarmFallingThreshold.'.'.$id, $FallingT, INTEGER));
$varbindlist->add(new Net_SNMP_VarBind(alarmRisingEventIndex.'.'.$id, $RisingE, INTEGER));
$varbindlist->add(new Net_SNMP_VarBind(alarmFallingEventIndex.'.'.$id, $FallingE, INTEGER));
$varbindlist->add(new Net_SNMP_VarBind(alarmOwner.'.'.$id, $owner, OCTET_STRING));
$PDU = $snmp->setRequest($varbindlist);
```

Figura 4.61: Segmento de código de la función *sendNetFlow*.

- **Configuración CDP:** Esta área permite la configuración del protocolo CDP en el *router*, a través de la MIB privada CISCO-CDP-MIB [24].

  Los objetos que pueden modificarse son:

  - cdpGlobalRun.
  - cdpGlobalMessageInterval.
  - cdpGlobalHoldTime.
  - cdpInterfaceEnable.

  Para realizar las solicitudes SNMP se desarrolló una función llamada "*sendCdp*", que toma los parámetros proporcionados por el usuario, y ejecuta las operaciones necesarias, como se observa en la Figura 4.62.

```
/***** cdpInterface Table******/
$Interface      = (string)$formSNMP["Interface"];
$EnableCDP      = transformBoolean($formSNMP["EnableCDP"]);

$varbindlist = new Net_SNMP_VarBindList();
$varbindlist->add(new Net_SNMP_VarBind(cdpGlobalRun.'.0',$GlobalRun,INTEGER));
$varbindlist->add(new Net_SNMP_VarBind(cdpGlobalMessageInterval.'.0',$MsgInterval,INTEGER));
$varbindlist->add(new Net_SNMP_VarBind(cdpGlobalHoldTime.'.0',$HoldTime,INTEGER));

$PDU = $snmp->setRequest($varbindlist);
```

Figura 4.62: Segmento de código de la función *sendCdp*.

4.3.10 Iteración 10

Para finalizar con el RMSWeb, se integró la funcionalidad de cambio de idioma, permitiendo así visualizar la aplicación en español o inglés, de acuerdo con la elección del usuario.

Esta implementación se llevó a cabo con el mismo procedimiento descrito en la iteración 5 para la herramienta NMSWeb.

4.4 Fase de Pruebas

En esta fase se diseñaron y ejecutaron pruebas con el objetivo de validar el correcto funcionamiento de la biblioteca de funciones SNMP, así como las aplicaciones NMSWeb y RMSWeb.

A continuación se describen cada una de las pruebas realizadas, y los resultados obtenidos.

4.4.1 Pruebas funcionales

Las pruebas funcionales se realizan con la finalidad de comprobar que los sistemas desarrollados cumplen con los requerimientos para los cuales han sido creados. Estas pruebas son basadas en el análisis de datos de entrada y salida.

En este caso, se diseñaron un conjunto de pruebas para cada una de las funcionalidades implementadas en la biblioteca de clases SNMP, a fin de validar que cada acción proporcione el resultado esperado. Estas pruebas se realizaron a través de la aplicación NMSWeb, que permite ejecutar cada una de las funciones desarrolladas en la biblioteca.

Para el escenario de pruebas, se instaló una red con dos *hosts* en distintos sistemas operativos.

En la Tabla 4.35 se pueden observar las características del sistema operativo, direccionamiento y software, de cada uno de los *hosts* instalados.

Característica	WinXP	Ubuntu
Sistema Operativo	Windows XP ServicePack 2	Ubuntu 8.04 Desktop
IPv4 – eth0	200.109.5.70	192.168.116.128
IPv4 – eth1	192.168.116.1	N/A
IPv6 – eth0	fe80::2e0:b8ff:fed3:9d3b	2001:fa::2/64
IPv6 – eth1	2001:fa::1/64	N/A
XAMPP	1.6.6a	1.6.6
Wireshark	1.0.3	N/A

Tabla 4.35 Características de los hosts de la red de prueba.

Adicionalmente en la Figura 4.63 se aprecia la topología de red configurada para el ambiente de pruebas.

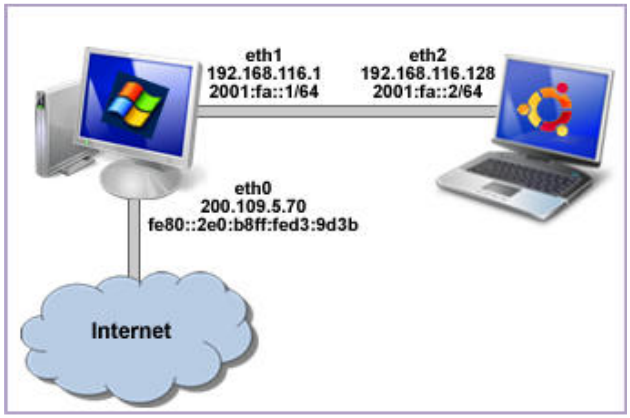

Figura 4.63 Topología de Red para las pruebas.

- **Solicitud GetRequest:** Se ejecutó una solicitud GetRequest, con el propósito de evaluar su funcionamiento. Los resultados de esta prueba fueron exitosos tanto a nivel de direcciones IPv4 e IPv6, en cada una de las plataformas de software antes mencionadas.

  En la Figura 4.64 se pueden apreciar los datos de entrada para esta consulta.

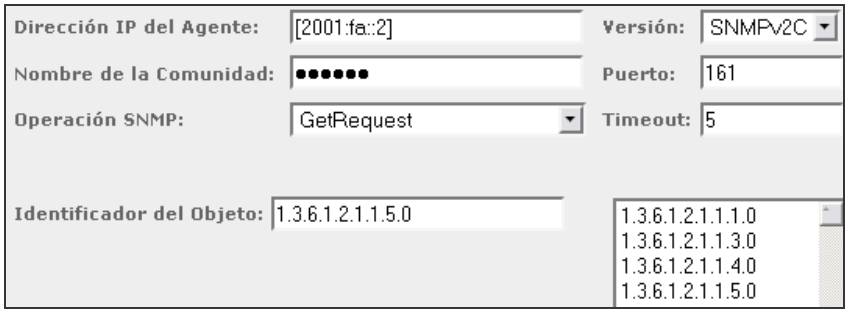

Figura 4.64: Datos de entrada para solicitud GetRequest.

En la Figura 4.65 se observan los resultados arrojados por el NMSWeb una vez que fue enviada la solicitud al *host* Ubuntu.

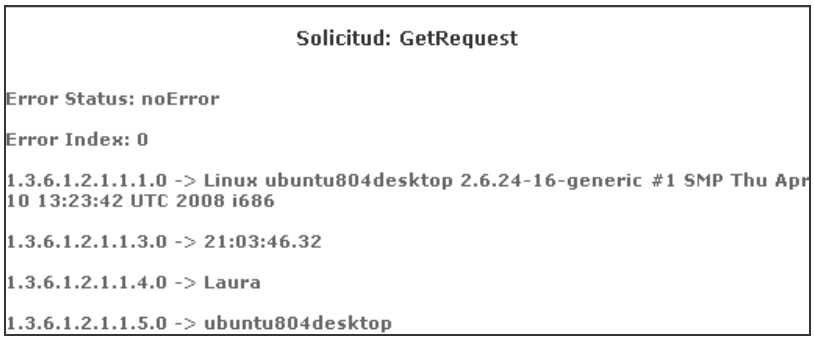

Figura 4.65: Respuesta de una solicitud GetRequest.

- **Solicitud SetRequest:** Esta prueba fue realizada con el objetivo de evaluar la primitiva denominada SetRequest que permite modificar el valor de objetos en el agente SNMP, en este caso sysName.

  Los resultados de esta prueba fueron exitosos en ambos *hosts*, sobre direcciones IPv4 e IPv6. En la Figura 4.66 se pueden observar los datos de entrada para esta solicitud.

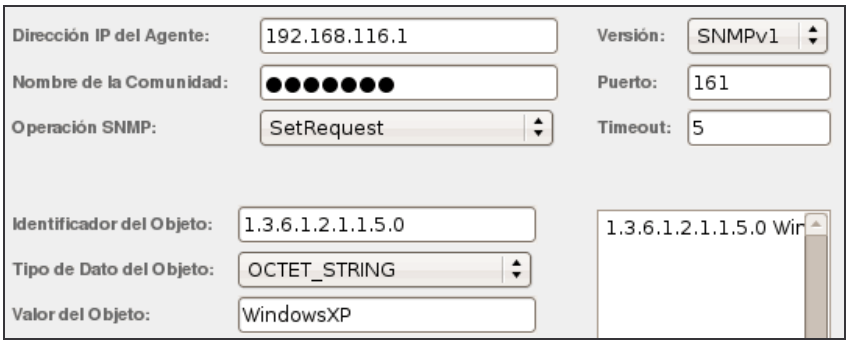

Figura 4.66: Datos de entrada para solicitud SetRequest.

Asimismo en la Figura 4.67 se observan los resultados tras la ejecución de dicha solicitud desde el *host* Ubuntu hacia el *host* WinXP.

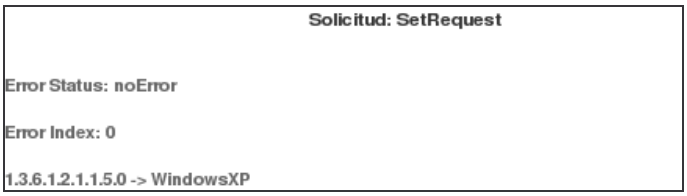

Figura 4.67: Respuesta de una solicitud SetRequest.

- **Solicitud GetNextRequest:** A través de esta prueba se comprobó la funcionalidad de la primitiva GetNextRequest implementada en la biblioteca.

  Cabe destacar que los resultados fueron favorables usando direcciones IPv4 e IPv6, y ejecutando la operación desde cada uno de los *hosts* (WinXP y Ubuntu).

  En la Figura 4.68 se pueden apreciar los datos de entrada utilizados para este caso de prueba.

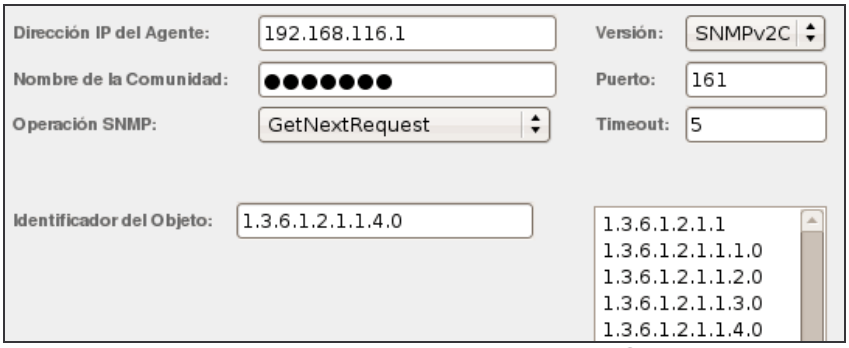

Figura 4.68: Datos de entrada para una solicitud GetNextRequest.

Adicionalmente en la Figura 4.69 se observan los datos arrojados por el NMSWeb, una vez que la solicitud fue hecha desde el *host* Ubuntu hacia el agente en el *host* WinXP.

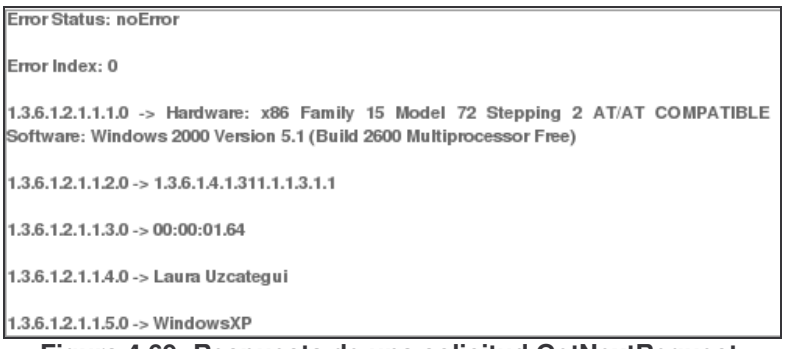

Figura 4.69: Respuesta de una solicitud GetNextRequest.

- **Solicitud GetBulkRequest:** Esta prueba se realizó con el objetivo de comprobar el correcto funcionamiento de la primitiva GetBulkRequest proporcionada por el protocolo SNMPv2c.

  En este caso fue ejecutada con la raíz de los grupos *system* e *interfaces*, con el valor 5 para el campo *max-repetitions* y el valor 1 para el campo *non-repeaters*, como se observa en la Figura 4.70.

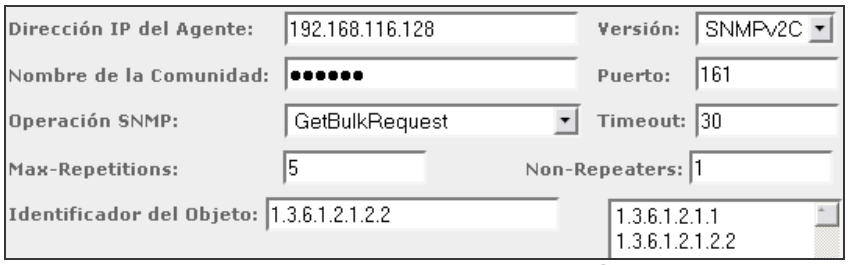

Figura 4.70: Datos de entrada para solicitud GetBulkRequest.

Los resultados de esta prueba fueron exitosos, permitiendo recuperar los valores requeridos, tanto en direccionamiento sobre IPv4 e IPv6, y en ambas plataformas de sistema operativo instaladas.

En la Figura 4.71 se pueden apreciar los datos obtenidos una vez que la solicitud fue hecha desde el *host* WinXP hacia el agente que se encuentra configurado en el *host* Ubuntu.

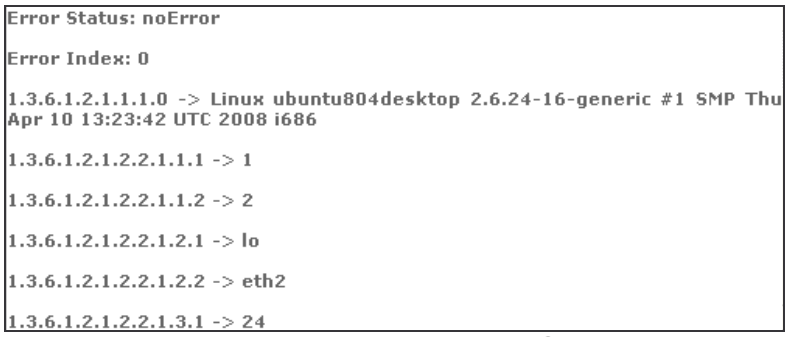

Figura 4.71: Respuesta de una solicitud GetBulkRequest.

- **Envío de notificación Trap:** A través de esta prueba se evaluó el funcionamiento de la operación que permite el envío de notificaciones tipo Trap.

  Los resultados obtenidos fueron favorables en ambos hosts, sobre direcciones IPv4 e IPv6. Los datos de entrada para la ejecución de este caso de prueba se observan en la Figura 4.72.

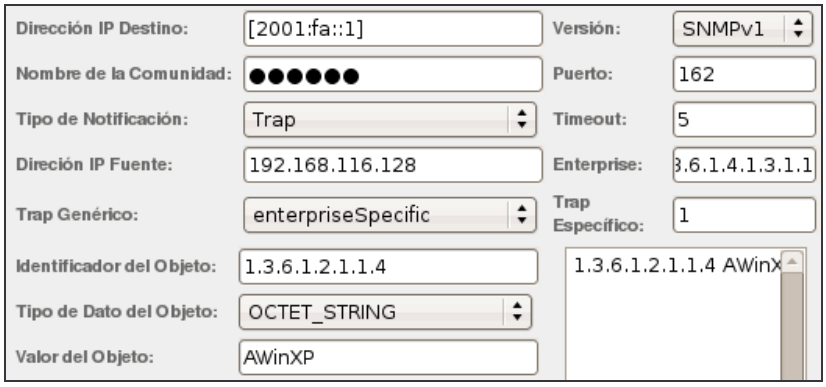

Figura 4.72: Datos de entrada para enviar un Trap.

En la Figura 4.73 se aprecia una captura con la recepción del Trap en el *host* WinXP.

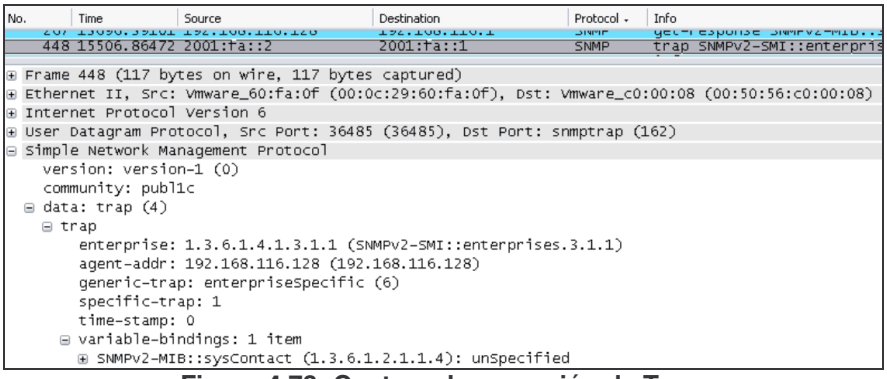

Figura 4.73: Captura de recepción de Trap.

- **Envío de notificación InformRequest:** Esta prueba se realizó con la finalidad de comprobar el correcto funcionamiento de la operación implementada para el envío de notificaciones de tipo InformRequest.

  En este caso, la prueba se ejecutó exitosamente sobre IPv4 e IPv6, en ambos sistemas operativos. En la Figura 4.74 se pueden observar los parámetros de entrada utilizados.

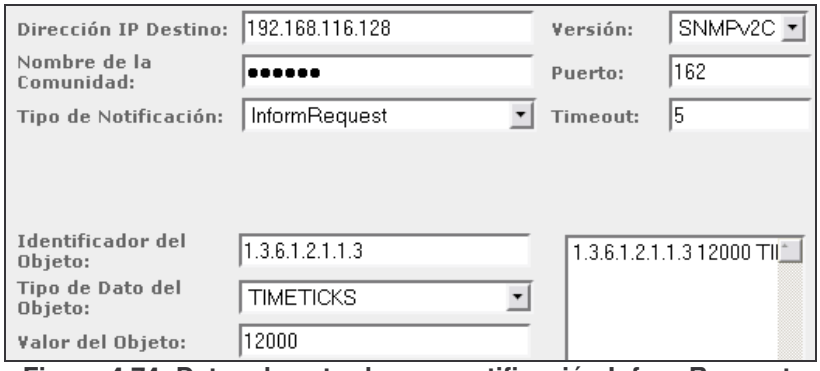

Figura 4.74: Datos de entrada para notificación InformRequest.

Por su parte, en la Figura 4.75 se aprecia una captura con el envío de la notificación hacia el *host* Ubuntu.

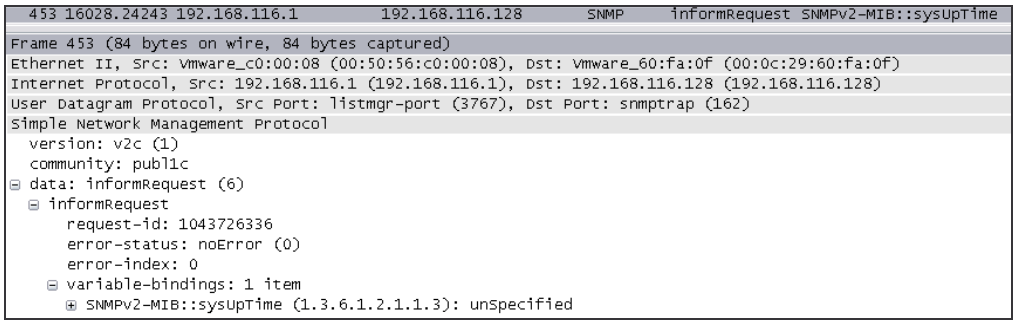

Figura 4.75: Captura de envío de un InformRequest.

- **Envío de notificación Trap SNMPv2C:** A través de esta prueba se validó el correcto funcionamiento del método encargado de enviar notificaciones trap, haciendo uso del protocolo SNMP en su versión 2c.

  Para ello desde el NMSWeb se introdujeron los parámetros de entrada que se observan en la Figura 4.76.

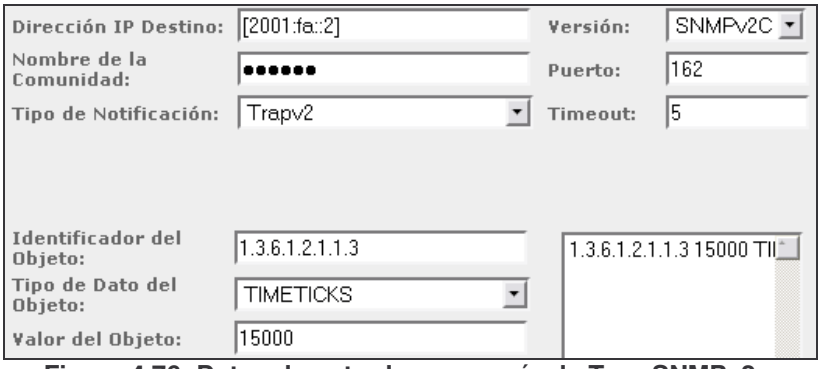

Figura 4.76: Datos de entrada para envío de Trap SNMPv2c.

Adicionalmente, en la Figura 4.77 se puede apreciar una captura con el envío del Trap SNMPv2c desde el *host* WinXP hacia el *host* Ubuntu

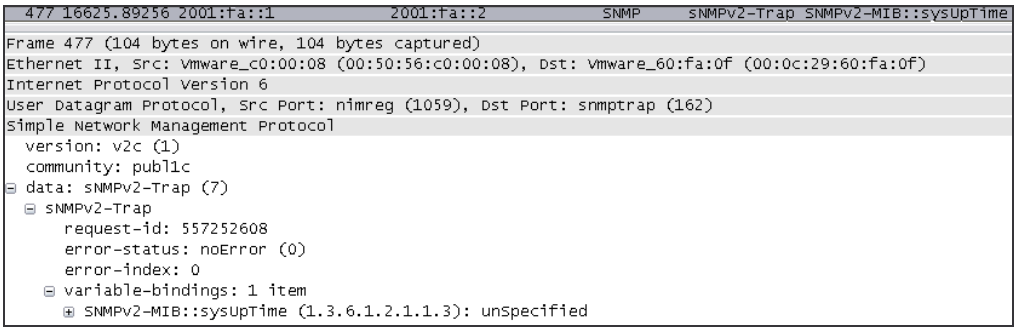

Figura 4.77: Captura de envío de Trap SNMPv2c.

La prueba resultó exitosa en ambas plataformas, sobre direccionamiento IPv4 e IPv6.

- **Operación Walk:** Esta prueba fue realizada para validar el correcto funcionamiento de la operación Walk.

  En la Figura 4.78 se observan los datos de entrada utilizados durante el caso de prueba.

Figura 4.78: Datos de entrada de la operación Walk.

Los resultados de esta prueba fueron favorables y se aprecian en la Figura 4.79, la misma se ejecutó sobre ambos sistemas operativos, utilizando direcciones IPv4 e IPv6.

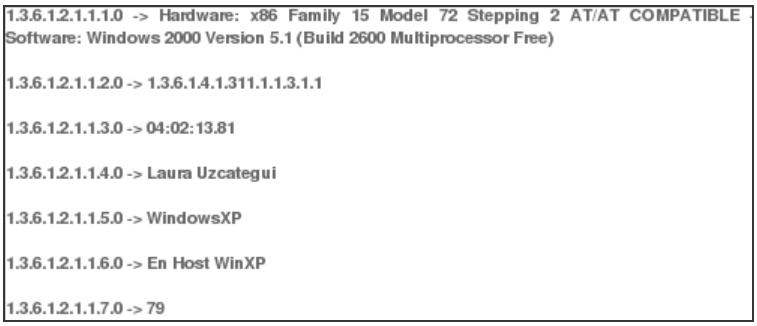

Figura 4.79: Respuesta de operación Walk.

- **Registrar Eventos:** Este caso de prueba permitió validar la opción de registros de eventos proporcionada por la biblioteca de funciones SNMP**.**

  En la Figura 4.80 se aprecia una vista del archivo generado, en el que se registran las operaciones efectuadas.

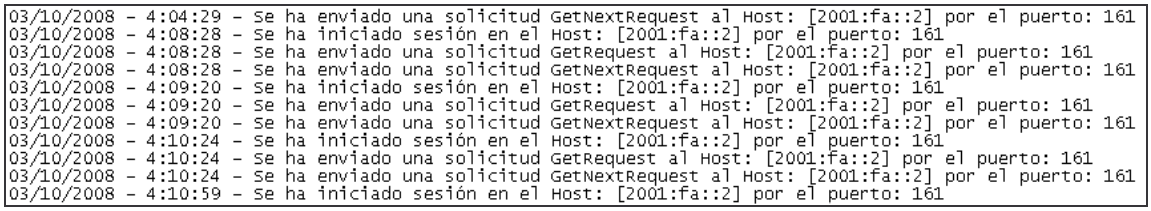

Figura 4.80: Registro de eventos en el log.

4.4.2 Pruebas de usabilidad.

La usabilidad, en desarrollo web, se conoce como la disciplina que se encarga de estudiar el diseño de sitios web, con el propósito de que los usuarios puedan interactuar con ellos de la forma más fácil, cómoda, e intuitiva posible [30].

Durante esta prueba, se evaluó la usabilidad de las herramientas NMSWeb y RMSWeb, considerando los criterios que se describen en la Tabla 4.36.

Principio	Descripción
Accesibilidad	El sitio es visible en cualquier configuración que tenga el cliente.
Navegabilidad	La información debe ser fácil de encontrar, múltiples maneras y vías de encontrar la misma información, consistencia en los elementos de navegación.
Lenguaje manejado	Debe ser fácil de entender, de acuerdo al tipo de usuario al que este dirigido la aplicación.
Conceptos utilizados	Deben ser adecuados al usuario al que está dirigida la aplicación.
Consistencia	En los títulos, cabeceras y etiquetas de links. La familiaridad y el reconocimiento deben permitir a los usuarios asociar ciertos elementos con ciertas funcionalidades.
Anticipación	Se provee al usuario toda la información necesaria en cada paso del proceso.
Facilidad de uso	La interfaz debe ser simple y sencilla de usar.
Prevención de errores	Aportar información que permita evitar errores, se debe cuidar que los mensajes de error devueltos sean sencillos y que aporten vías de solución.
Estética y diseño	La información debe estar estructurada en niveles de detalle progresivos.

Tabla 4.36: Descripción de principios de usabilidad.

Con el propósito de poder evaluar los principios descritos anteriormente, se aplicó una encuesta a un conjunto de usuarios potenciales, los cuales asignaron una puntuación comprendida entre 1 y 5, a cada uno de los criterios de usabilidad.

Posteriormente, los resultados fueron recabados y totalizados, calculando el promedio obtenido por cada uno de los criterios.

A continuación se describen los resultados de la encuesta, para cada una de las aplicaciones web.

- **NMSWeb:** En la Tabla 4.37 se observa el resultado de la evaluación de usabilidad aplicada a esta herramienta.

Caso de Prueba NMSWeb	
Principio Básico	**Valoración**
Accesibilidad	★★★★★
Navegabilidad	★★★★★
Lenguaje manejado	★★★★★
Conceptos Utilizados	★★★★★
Consistencia	★★★★★
Anticipación	★★★★★
Facilidad de Uso	★★★★★
Prevención de Errores	★★★★★
Estética y Diseño	★★★★★

Tabla 4.37: Evaluación de usabilidad para el NMSWeb.

- **RMSWeb:** En la Tabla 4.38 se aprecian los resultados de la evaluación de usabilidad aplicada.

Caso de Prueba RMSWeb	
Principio Básico	**Valoración**
Accesibilidad	★★★★★
Navegabilidad	★★★★½
Lenguaje manejado	★★★★★
Conceptos Utilizados	★★★★★
Consistencia	★★★★★
Anticipación	★★★★½
Facilidad de Uso	★★★★★
Prevención de Errores	★★★★½
Estética y Diseño	★★★★★

Tabla 4.38: Evaluación de usabilidad para el RMSWeb.

De esta manera, es posible concluir que los resultados de esta evaluación fueron favorables, ya que para ambas aplicaciones se obtuvo la máxima ponderación en la mayoría de los criterios tomados en cuenta, lo que indica que las herramientas resultan intuitivas y fáciles de usar para los usuarios potenciales.

4.4.3 Pruebas de valor frontera

Estas pruebas se realizaron con la finalidad de comprobar el comportamiento de las aplicaciones web, cuando algún dato introducido por el usuario es un valor inusual o excede el máximo permitido.

Entre los casos de prueba que fueron ejecutados se encuentran colocar datos incorrectos al llenar un formulario, datos vacios y que son obligatorios para realizar alguna de las operaciones, o valores que no son compatibles con el dato que debería ser introducido.

Los resultados de esta prueba fueron exitosos, debido a que el conjunto de valores que fueron probados en ambas aplicaciones, desplegaron avisos como los que pueden apreciarse en la Figura 4.81.

Figura 4.81 Advertencias para indicar datos incorrectos o faltantes.

4.4.4 Pruebas Cross Browser

Estas pruebas se realizaron con el propósito de evaluar el comportamiento de las aplicaciones NMSWeb y RMSWeb, cuando son ejecutadas en distintos navegadores, esperando obtener la misma funcionalidad y apariencia.

Los navegadores utilizados durante la evaluación fueron:

- Mozilla Firefox.
- Internet Explorer.
- Opera.
- Safari.
- Netscape.
- Google Chrome.

En la Tabla 4.39 se puede observar un resumen del comportamiento de las aplicaciones NMSWeb y RMSWeb, al ser ejecutadas sobre diversos navegadores.

Navegador	Versión	Comportamiento
Mozilla Firefox	3.0.3	En este navegador las aplicaciones funcionan correctamente, y la apariencia es adecuada con respecto al diseño realizado.
Internet Explorer	7.0.5730.13	En este navegador las aplicaciones funcionan correctamente. Sin embargo, al posicionar el mouse sobre los botones estos no cambian de color para indicar que se está ejecutando alguna acción sobre los mismos.
Opera	9.52	Las aplicaciones funcionan correctamente, y la apariencia está acorde con el diseño realizado.
Safari	3.1.2	En este navegador las aplicaciones funciona correctamente, y no existen cambios en la apariencia. Sin embargo, a diferencia de los otros navegadores, cuando el puntero del mouse se encuentra sobre alguno de los campos de entrada el área de texto aparece delimitada en color azul.
Netscape	9.0.0.4	Las aplicaciones funcionan adecuadamente, y mantienen la apariencia diseñada.
Google Chrome	0.2.149.30	En este navegador las aplicaciones funcionan correctamente, y la apariencia es adecuada de acuerdo al diseño. Sin embargo, cuando el puntero del mouse se encuentra en alguno de los campos de entrada el área de texto aparece delimitada en color amarillo.

Tabla 4.39: Descripción de pruebas Cross Browser para el NMSWeb y RMSWeb.

Los resultados de este conjunto de pruebas fueron favorables, ya que en los navegadores utilizados las diferencias de apariencia fueron mínimas, y el comportamiento de las aplicaciones fue el esperado.

Para finalizar, en la Figura 4.82 y Figura 4.83 se observa un ejemplo de la vista de las aplicaciones NMSWeb y RMSWeb, en los navegadores Mozilla Firefox y Safari respectivamente.

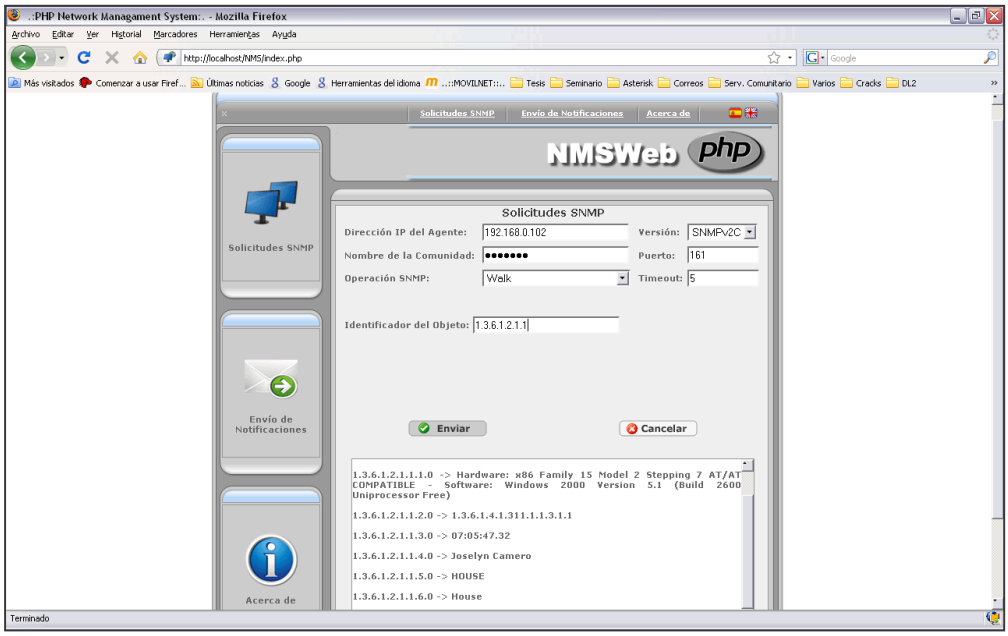

Figura 4.82: Vista del NMSWeb en Mozilla Firefox.

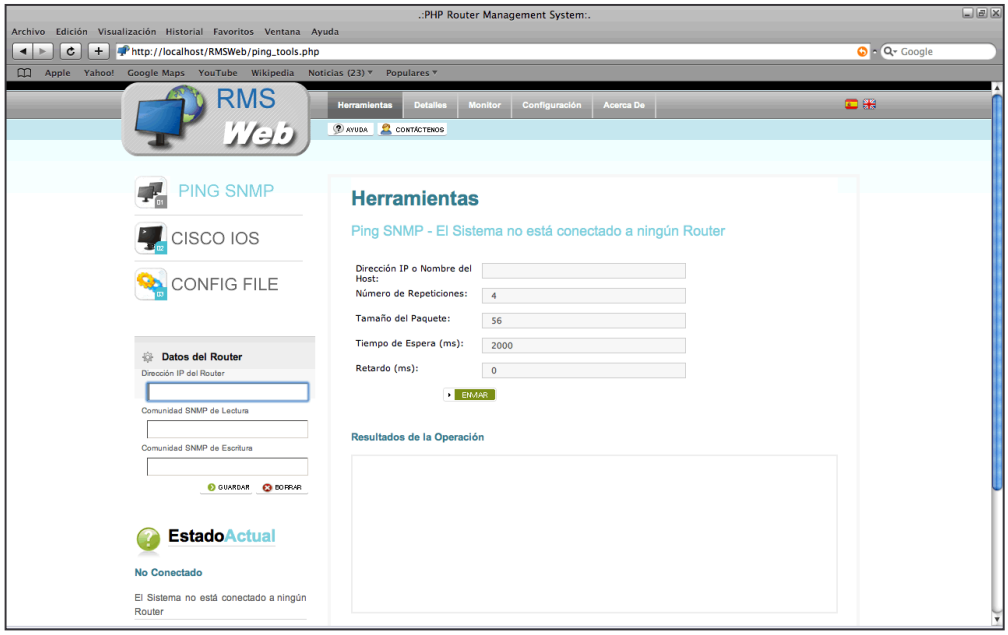

Figura 4.83: Vista del RMSWeb desde Safari.

5. Conclusiones

A medida que avanza la tecnología, aumenta la tendencia de desarrollar diferentes tipos de aplicaciones en ambiente web. Entre uno de los principales lenguajes de programación que resaltan en esta área se encuentra PHP, por razones tales como su potencia, portabilidad, aceptación, entre otras. Desde sus inicios, este lenguaje ha sido construido gracias a la participación de múltiples colaboradores que han permitido añadir clases, extensiones u operaciones, que dan respuestas a requerimientos comunes. Entre estos requerimientos resalta el soporte brindado para el manejo de diversos protocolos de red, incluido SNMP. Sin embargo, la extensión que permite hacer uso de este protocolo tiene múltiples deficiencias que desmejoran aspectos positivos del lenguaje, como por ejemplo la portabilidad.

Por estas razones, se desarrolló una biblioteca de clases que implementa el protocolo SNMP v1/2c y cubre las carencias detectadas en la extensión del lenguaje, brindando portabilidad y una documentación lo suficientemente acertada como para facilitar el uso y aprendizaje de sus funciones. Además, por ser PHP un lenguaje de código abierto, esta biblioteca podrá ser revisada y manipulada por otros desarrolladores, brindando la posibilidad de que se añadan nuevas características a la implementación del protocolo, lo que garantiza su extensibilidad.

Entre las características de la biblioteca Net_SNMP se encuentran:

- Permite el envío de las solicitudes GetRequest, GetNextRequest, SetRequest, GetBulkRequest.
- Permite el envío de las notificaciones Trap, InformRequest, Trapv2.
- Implementa la operación Walk, para consultar subárboles de la MIB.
- Proporciona un registro de eventos que permite al usuario tener conocimiento de las operaciones que han sido ejecutadas por la biblioteca.
- Brinda un mecanismo de manejo de errores, a través de la definición de excepciones.
- Proporciona soporte para los protocolos de transporte UDP/IPv4 o UDP/IPv6.
- Es portable, y funciona adecuadamente en diversas plataformas y sistemas operativos.
- Es orientada a objetos, lo que facilita su mantenimiento y escalabilidad.
- Contiene una documentación amplia y sencilla del código.

Adicionalmente se desarrollaron dos aplicaciones web para comprobar el correcto funcionamiento y la facilidad de uso de la biblioteca Net_SNMP:

- **RMSWeb**: Brinda herramientas para la administración y configuración de algunas características de *routers Cisco Systems*, específicamente para el modelo 2811.

- **NMSWeb**: Permite ejecutar las primitivas SNMP v1/v2c, y está orientada a usuarios que tengan conocimientos sobre el protocolo.

Como recomendaciones para trabajos posteriores, se sugiere:

- Agregar soporte para SNMPv3.
- Probar el desempeño de la biblioteca sobre plataformas de 64 bits.
- Añadir métodos que permitan la importación y compilación de MIBs.

Lista de Acrónimos

AIX (Advanced Interactive eXecutive): Es un sistema operativo Unix.

AJAX (Asynchronous JavaScript And XML): JavaScript Asíncrono y XML.

ARPANET (Advanced Research Projects Agency Network): Red de la Agencia de Proyectos de Investigación Avanzada de EEUU.

ASCII (American Standard Code for Information Interchange): Código Estándar para el Intercambio de Información.

ASN.1 (Abstract Syntax Notation One): Notación de Sintaxis Abstracta.

BER (Basic Encoding Rules): Reglas Básicas de Codificación.

CCITT (Consultative Committee for International Telegraphy and Telephony): Comité Consultivo Internacional Telegráfico y Telefónico.

CDP (Cisco Discovery Protocol): Protocolo de Descubrimiento de Cisco.

CHM (Microsoft Compiled HTML Help): Formato propio para archivos de ayuda de Microsoft.

CPU (Central Processing Unit): Unidad Central de Procesamiento.

CRC (Class Responsibility Collaborator): Clases, responsabilidades, Colaboradores. Artefacto utilizado en el diseño de sistemas orientados a objetos.

EGP (Exterior Gateway Protocol): Protocolo usado para intercambio de información entre *gateways* exteriores.

FXS (Foreign Exchange Station): Interfaz para la conexión de teléfonos analógicos al computador.

GNU (GNU is Not Unix): Acrónimo recursivo que significa GNU no es Unix. Representa al software no propietario, basado en el sistema informático Unix.

GPL (General Public License): Licencia Pública General.

HP-UX (Hewlett Packard Unix): Versión de Unix desarrollado por Hewlett Packard.

HTML (HyperText Markup Language): Lenguaje de Etiquetas de Hipertexto.

IAB (Internet Architecture Board): Comité encargado de vigilar el desarrollo técnico de la Internet por la Internet Society (ISOC).

ICMP (Internet Control Message Protocol): Protocolo de Mensajes de Control de Internet.

IETF (Internet Engineering Task Force): Grupo de Trabajo de Ingeniería de Internet.

IOS (Internetwork Operating System): Sistema Operativo de Interconexión de Redes.

IP (Internet Protocol): Protocolo Internet.

ISO (International Organization for Standardization): Organización Internacional para la Estandarización.

Mac OS (Macintosh Operating System): Sistema Operativo de Macintosh.

MIB (Management Information Base): Base de Información de Gestión.

mSQL (Mini Structured Query Language): Es un sistema manejador de base de datos relacional que permite acceder a los datos sin necesidad de grandes cantidades de memoria.

NMS (Network Management System): Sistema de Administración de Red.

NMSWeb (Network Management System Web): Sistema Web para la Administración de Red.

OID (Object Identifier): Identificador de Objeto.

OSI (Open Systems Interconnection): Interconexión de Sistemas Abiertos.

PDA (Personal Digital Assistant): Asistente Personal Digital.

PDF (Portable Document Format): Formato de Documento Portable.

PDU (Protocol Data Unit): Unidad de Dato de Protocolo.

PEAR (PHP Extension and Application Repository): Repositorio de Extensiones y Aplicaciones PHP.

PERL (Practical Extraction and Report Language): Lenguaje Práctico para la Extracción e Informe.

PHP (PHP Hypertext Preprocessor): Lenguaje de programación interpretado, utilizado en desarrollos web.

PING (Packet Internet Grouper): Utilidad que comprueba el estado de conexión con equipos remotos, a través de paquetes ICMP.

RAM (Random Access Memory): Memoria de Acceso Aleatorio.

RFC (Request For Comment): Solicitud de Comentario. Serie de documentos oficiales que describen la familia de protocolos de Internet.

RMON (Remote Monitoring): Monitoreo Remoto.

RMSWeb (Router Management System Web): Sistema Web para la Administración de Router.

ROM (Read Only Memory): Memoria de Sólo Lectura.

SGMP (Simple Gateway Management Protocol): Protocolo Simple de Administración de Gateway.

SMI (Structure of Management Information): Estructura de Información de Gestión.

SNMP (Simple Network Management Protocol): Protocolo Simple de Administración de Redes.

TCP (Transmission Control Protocol): Protocolo de Control de Transmisión. Es un protocolo de la capa de transporte de OSI orientado a conexión que proporciona una transmisión confiable de datos.

TFTP (Trivial File Transfer Protocol): Protocolo de Transferencia de Archivos.

TLV (Type, Length, Value): Tipo, Longitud, Valor. Tripleta utilizada por las reglas de codificación estándar (BER).

UDP (User Datagram Protocol): Protocolo de Datagrama de Usuario.

VAD (Voice Activity Detection): Detección de Actividad de Voz.

VoIP (Voice over Internet Protocol): Voz sobre Protocolo Internet.

VPDN (Virtual Private Dialup Network): Red Privada de Marcado.

WML (Wireless Markup Language): Es un lenguaje utilizado para escribir aplicaciones para medios inalámbricos.

XAMPP (X Apache, MySQL, PHP, Perl): Servidor portable que agrupa Apache, MySQL, PHP y Perl.

XML (eXtensible Markup Language): Lenguaje Extensible de Marcas. Es un metalenguaje extensible de etiquetas.

XP (eXtreme Programming): Programación Extrema.

Glosario de Términos

Administración de Redes
Conjunto de técnicas utilizadas para mantener a una red operativa, eficiente, segura, y constantemente monitoreada.

Base de Datos
Organización sistemática de una serie de datos relacionados entre sí y almacenados en un soporte informático.

Cisco Systems
Empresa multinacional dedicada a la fabricación, venta, mantenimiento y consultoría de equipos de telecomunicaciones.

Clase Abstracta
Es una clase base que contiene definiciones comunes, que no se puede instanciar, y de la cual se derivan múltiples clases.

Datagrama
Unidad de información transmitida por los protocolos de nivel de red.

Estándar de Facto
Patrón o norma que no ha sido legitimada por un organismo de estandarización y ha sido impuesto por una o varias empresas.

Flash
Tipo de memoria usada por los routers Cisco Systems para almacenar el archivo comprimido de la imagen del Cisco IOS.

Framework
Conjunto de APIs y herramientas destinadas a la construcción de un determinado tipo de aplicaciones.

FreeBSD
Sistema operativo libre, derivado de la implementación de Unix de la Universidad Berkeley.

Internet
Red de computadores a nivel mundial.

JavaScript
Lenguaje de programación interpretado, utilizado principalmente en el desarrollo web.

Lenguaje C
Lenguaje de programación de bajo y alto nivel al mismo tiempo.

Linux
Sistema operativo tipo Unix, que se distribuye bajo la Licencia Pública General de GNU.

Macro
Conjunto de instrucciones que ejecutan una función automáticamente dentro de un programa.

Modelo OSI
Marco de referencia para la definición de arquitecturas de interconexión de sistemas de comunicaciones.

Motor Zend
Responsable de analizar el código PHP, y definir la sintaxis del lenguaje.

MySQL
Sistema de gestión de base de datos relacional, multihilo y multiusuario.

NetBSD
Sistema operativo derivado del Unix de la Universidad de Berkeley.

NetFlow
Protocolo de red desarrollado por Cisco Systems para la recolección de información de tráfico IP.

Open Source
Práctica de desarrollo de software que promueve el acceso al código fuente de los sistemas computacionales.

Paquete
Cantidad mínima de datos que se transmite en una red o entre dispositivos.

Protocolos de Red
Definen las diferentes reglas y normas que rigen el intercambio de información entre nodos de la red.

Refactorización
Técnica de la ingeniería del software para reestructurar un código fuente, alterando su estructura interna sin cambiar su comportamiento externo o funcionalidades.

Reutilización de Código
Técnicas que garantizan que una parte o la totalidad de un código existente se pueda emplear en la construcción de otro programa.

Router
Dispositivo de hardware para la interconexión de redes de computadoras, que opera en la capa de red del modelo OSI.

Shell
Interprete de comando de un sistema operativo.

Software
Conjunto de programas y procedimientos necesarios para hacer posible la realización de una tarea específica.

Solaris
Es un sistema operativo desarrollado por Sun Microsystems.

Switch
Dispositivo para la interconexión de redes de computadoras que opera en la capa de enlace del modelo OSI.

Ubuntu
Distribución del sistema operativo Linux.

Unix
Sistema operativo multitarea y multiusuario, que fue desarrollado a principios del año 1969, y se caracteriza por ser portátil y versátil.

Usabilidad
Atributo de calidad de una página o sitio web, que determina la facilidad de la interfaz para ser utilizada.

Windows
Sistema operativo desarrollado por la empresa Microsoft.

Referencias Bibliográficas

[1] A. Tanenbaum. Computer Networks (3a. ed.). Prentice Hall. Marzo, 1996.

[2] J. Davin, J. Case, M. Fedor, M.L. Schoffstall. Simple Gateway Monitoring Protocol. RFC 1028. Noviembre, 1987.

[3] J. Case, M. Fedor, M. Schoffstall, J. Davin. Simple Network Management Protocol. RFC 1067. Agosto, 1988.

[4] R. Mauro, J. Schmidt. Essential SNMP (1a. ed.). O'Reilly. Julio, 2001.

[5] M. Miller. Managing Internetworks with SNMP. M & T Books. Mayo, 1997.

[6] O. Dubuisson. ASN.1 Communication between Heteregoneus Systems. Morgan Kaufmann Publishers. Septiembre, 2000.

[7] K. McCloghrie, M. Rose. Management Information Base for Network Management of TCP/IP-based Internets. RFC 1156. Mayo, 1990.

[8] K. McCloghrie, M. Rose. Management Information Base for Network Management of TCP/IP-based Internets: MIB-II. RFC 1213. Marzo, 1991.

[9] K. McCloghrie, M. Rose. Structure and Identification of Management Information for TCP/IP-based Internets. RFC 1155. Mayo, 1990.

[10] K. McCloghrie, J. Schoenwaelder. Structure of Management Information Version 2. RFC 2578. Abril, 1999.

[11] J. Case, M. Fedor, M. Schoffstall, J. Davin. Simple Network Management Protocol. RFC 1157. Mayo, 1990.

[12] R. Presuhn, J. Case, K. McCloghrie, M. Rose, S. Waldbusser. Version 2 of the Protocol Operations for the Simple Network Management Protocol. RFC 3416. Diciembre, 2002.

[13] W. Gilmore. A Programmer's Introduction to PHP 4.0. Apress. Enero, 2001.

[14] A. Wormus, C. Lucke, S. Schmidt, S. Stefanov. PHP Programming with PEAR. PACKT Publishing. Septiembre, 2006.

[15] C. Andres, K. Beck. Extreme Programming Explained (2a. ed.). Addison Wesley Professional. Noviembre, 2004.

[16] The PEAR Documentation Group. PEAR Manual. http://pear.php.net/manual/en. Agosto, 2008.

[17] J. Johnson. Cisco Ping MIB File. ftp://ftp.cisco.com/pub/mibs/v2/CISCO-PING-MIB.my. Mayo, 1994.

[18] Cisco Systems. Old Cisco Flash MIB File. ftp://ftp.cisco.com/pub/mibs/v1/OLD-CISCO-FLASH-MIB.my. Mayo, 1994.

[19] R. Kavasseri. Cisco Copy Config MIB. ftp://ftp.cisco.com/pub/mibs/v2/CISCO-CONFIG-COPY-MIB.my. Diciembre, 1997.

[20] Cisco Systems. Old Cisco Chassis MIB. ftp://ftp.cisco.com/pub/mibs/v1/OLD-CISCO-CHASSIS-MIB-V1SMI.my. Mayo, 1994.

[21] H. Shih. Voice Analog Interface MIB File. ftp://ftp.cisco.com/pub/mibs/v2/CISCO-VOICE-ANALOG-IF-MIB.my. Septiembre, 1996.

[22] H. Shih. Voice Interface MIB File. ftp://ftp.cisco.com/pub/mibs/v2/CISCO-VOICE-IF-MIB.my. Septiembre, 1996.

[23] C. White. Voice Common Dial Control MIB File. ftp://ftp.cisco.com/pub/mibs/v2/CISCO-VOICE-COMMON-DIAL-CONTROL-MIB.my. Junio, 1999.

[24] E. Pham. CDP MIB File. ftp://ftp.cisco.com/pub/mibs/v2/CISCO-CDP-MIB.my. Noviembre, 2001.

[25] J. Chan. VPDN Management MIB. ftp://ftp.cisco.com/pub/mibs/v2/CISCO-VPDN-MGMT-MIB.my. Julio, 1997.

[26] J. Johnson. Old Cisco CPU MIB File. ftp://ftp.cisco.com/pub/mibs/v1/OLD-CISCO-CPU-MIB.my. Mayo, 1994.

[27] S. Wang. Cisco Memory Pool MIB. ftp://ftp.cisco.com/pub/mibs/v2/CISCO-MEMORY-POOL-MIB.my. Julio, 2001.

[28] S. Waldbusser. Remote Network Monitoring Management Information Base. RFC 1271. Noviembre, 2001.

[29] N. Kundu, P. Aitken. Cisco NetFlow MIB. ftp://ftp.cisco.com/pub/mibs/v2/CISCO-NETFLOW-MIB.my. Enero, 2004.

[30] J. Nielsen. Usabilidad: Diseño de sitios web. Prentice Hall. 2000.