Universidad de las Ciencias Informáticas

Título: Módulo de Intérpretes para el Laboratorio Virtual Configuración y Administración de servicios de una Red de Área Local del Proyecto PROLAVI.

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

Autor:

José Alfredo Vilató Frómeta

Tutor:

Ing. Yaself Machado Tugores

Co-Tutor:

Ing. Gelson Rafael Saurin Ojeda

Ciudad de La Habana, Mayo 2011

Año 53 de la Revolución

Frase

Si no existe la organización, las ideas, después del primer momento de impulso, van perdiendo eficacia.

Ernesto "Che" Guevara.

Declaración de Autoría.

Declaro ser autor del presente trabajo de diploma y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales del mismo y a realizar uso de éste en su beneficio. Para que así conste firmo el presente a los ____ días del mes de _____ del año _____.

_____

Autor:

José Alfredo Vilató Frómeta.

_____ _____

Tutor: Co-Tutor:

Ing. Yaself Machado Tugores. Ing. Gelson Rafael Saurin Ojeda.

Datos de Contacto.

Tutor: Ing. Yaself Machado Tugores.

Edad: 26 años.

Ciudadanía: Cubano.

Institución: Universidad de las Ciencias Informáticas.

Título: Ingeniero en Ciencias de la Computación.

Categoría Docente: Profesor Instructor.

E-mail: ytugores@uci.cu.

Graduado de la Universidad de las Ciencias Informáticas.

Co-Tutor: Ing. Gelson Rafael Saurin Ojeda.

Edad: 26 años.

Ciudadanía: Cubano.

Institución: Universidad de las Ciencias Informáticas.

Título: Ingeniero en Ciencias de la Computación.

Categoría Docente: Profesor Instructor.

E-mail: grsaurin@uci.cu.

Graduado de la Universidad de las Ciencias Informáticas.

Dedicatoria.

A mis padres:

Por la dedicación y el amor incondicional que me han brindado, por apoyarme en todas las metas que me he trazado en la vida.

A mis hermanos:

Que siempre están presentes en el momento que los necesito, por ser ejemplos a seguir de abnegación y sacrificio.

A mi familia:

Por estar siempre presente para brindarme su ayuda en todo momento.

A mi novia:

Por todo el amor que me ha dado y por apoyarme en todos estos años de la carrera.

Agradecimientos.

Quisiera agradecerle a cada una de las personas que han estado a mi lado e hicieron posible la culminación de esta investigación.

A mis padres:

Que han sido mi inspiración para convertirme en el profesional que hoy soy, ustedes que no han dejado de luchar ni un minuto pensando siempre en educar a sus hijos, inculcándome siempre con amor y cariño a ser honrado por eso y todo su sacrificio a ustedes agradezco en este día. Los amo.

A mis hermanos:

Por sus buenos consejos, por ser mis mejores amigos y la ayuda brindada en cada día de mi vida, porque siempre han estado presentes en cada paso que doy. Gracias.

A mis sobrinos:

Ustedes que son mis primeros hijos con los he aprendido mucho, ustedes también merecen mi agradecimiento en este día Kiki, Ronaldo, Sabrina y el pequeño Jose.

A mi familia:

Sin su apoyo no lo habría logrado. Gracias a todos

A mi novia:

Que ha estado a mi lado con su amor y comprensión en los momentos difíciles. Hoy más que nunca te extraño y necesito que estés presente en mi vida. Te amo.

A mi segunda familia:

Los padres de mi novia que me han acogido como un hijo más dándome su cariño, Chichi y Carmen, a Niuvis mi cuñada eres una hermana más para mí, al nuevo sobrino que nos trae mucha alegría en nuestras vidas y a todos los demás que no menciono por temor a que se me olvide no plasmar algún nombre aquí, Gracias.

A mis compañeros y amigos:

Mis compañeros del proyecto Prolavi y los amigos que conocí durante la carrera, a los de antes también. Por darme su amistad, de todos me llevo algo. No quiero mencionar nombres para que no se me olvide nadie. Gracias a todos.

A mis profesores:

A ustedes que dieron lo mejor de sí para educarme desde que comencé mis estudios y sobre todo a los de la UCI por formarme como un profesional, A Sailyn, Gelson y Osvaldo por su ayuda para la culminación de esta investigación. Gracias.

A la Revolución:

Por crear esta excelente universidad en la cual tuve la oportunidad de estudiar la Universidad de las Ciencias Informáticas.

Resumen.

En los últimos años los Laboratorios Virtuales se han convertido en herramientas ampliamente utilizadas, con el objetivo de facilitar y mejorar los procesos de aprendizajes en los centros de estudios. Muchas son las materias que hacen uso de los éstos para fortalecer los conocimientos adquiridos en las aulas. Con el auge de la computación, es necesario el estudio de temas relacionados con las redes de computadoras, con el fin de que los estudiantes desarrollen habilidades para lograr configurar los distintos servicios que se brindan en una red. La presente investigación aborda el desarrollo de un módulo de intérpretes que permite la validación de la configuración de algunos de los servicios existentes en una red, para luego ser incorporado al Laboratorio Virtual de Administración y Configuración de servicios de una red LAN.

Para lograr el desarrollo del módulo fue necesario el estudio de temas relacionados con el proceso de compilación en los intérpretes, de las materias de redes de computadoras que están presentes en el módulo y de las distintas herramientas que se valoraron para el desarrollo del mismo, como es el caso de C++ como lenguaje de programación y el uso de las bibliotecas Qt en el desarrollo del módulo.

Palabras Clave.

Laboratorios Virtuales, Redes de Computadoras, Configuración, Servicios, Intérpretes, LAN, Proceso de Compilación.

Índice.

Índice de Figuras.

Índice de Tablas.

Introducción.

El actual desarrollo de las Tecnologías de la Información y las Comunicaciones (TICs) permite que día a día estén presentes en nuestra sociedad. Son innumerables los beneficios alcanzados mediante la aplicación de las TICs en las distintas esferas que necesita un país para su actual desarrollo.

La educación no está exenta de todas las trasformaciones que trae consigo la aplicación de las nuevas TICs. Esto queda evidenciado con su vinculación al proceso de enseñanza, dando la posibilidad de aplicar nuevos métodos en el aprendizaje y la transmisión del conocimiento a los estudiantes.

Un nuevo concepto que se maneja en el desarrollo del software con fines educativos es el de Realidad Virtual (RV), mediante la aplicación de la misma podemos lograr que el usuario se adentre en un mundo virtual que se asemeja mucho a lo que conocemos en la realidad. La RV introduce nuevas herramientas como son los Laboratorios Virtuales (LV), que tratan de recrear el ambiente de un Laboratorio Tradicional (LT). Hoy en día la falta de instrumentos en las distintas prácticas que se realizan en un LT, traen como consecuencia un mayor desarrollo de Laboratorios Virtuales para apoyar la enseñanza, principalmente en asignaturas como Física, Matemática, Biología y muchas otras que se relacionan con la rama de la computación.

La Universidad de las Ciencias Informáticas (UCI) se ha adentrado en el desarrollo de los LV, en específico el Proyecto de Laboratorios Virtuales (PROLAVI), de la Facultad 5. Actualmente el proyecto desarrolla un LV de Configuración y Administración de una red de LAN (Red de Área Local). En el mismo se quiere lograr validar varias configuraciones como son, el servicio de dirección IP (Protocolo de Internet) de una computadora, configuración de un servicio FTP (Protocolo de Trasferencia de Ficheros), configuración del servicio DNS (Sistema de Nombres de Dominio), configuración de un servicio Proxy y configuración del servicio Samba, todas las configuraciones se realizarán mediante la utilización de los ficheros originales de Linux. Para lograr la validación de las configuraciones de los servicios, es necesario un proceso de compilación, que no es más que una acción realizada por una herramienta la cual toma como entrada un código fuente y realiza a partir del mismo los análisis léxicos, sintácticos y semánticos para llegar a la correcta configuración [1].

Para el laboratorio es necesaria una herramienta que forme parte del mismo, que sea robusta, confiable y eficiente, de forma que permita que los usuarios que se adentren en la práctica puedan configurar los servicios y además conocer los errores cometidos en la configuración de los mismos.

A partir de todo lo anterior se tiene como **Situación Problémica** que**:** Actualmente la validación de la tarea desarrollada por el estudiante debe hacerla el profesor de forma manual, lo que implica una gran cantidad de trabajo para la calificación de las mismas, debido a que no existe un software que permita la validación de las configuraciones de los servicios de una red LAN.

Por tanto, como **Problema Científico** de la investigación se propone: ¿Cómo elevar la eficacia de los profesores en la validación de las tareas prácticas realizadas por los estudiantes, en la configuración de los servicios de una red LAN?

Por lo que el **Objeto de Estudio** queda precisado como: Validaciones para servicios de redes LAN.

Para darle solución al problema planteado se formula el siguiente **Objetivo General**: Desarrollar un módulo de intérpretes que permita la validación de las configuraciones de los servicios de una red LAN.

La investigación se va a centrar en el **Campo de Acción:** Desarrollo de intérpretes para validar la configuración de los servicios de una red LAN.

Como **Idea a Defender** se propone que: Con la aplicación del módulo desarrollado se va a poder elevar la eficacia en la validación de las configuraciones de los servicios que presenta el Laboratorio Virtual de Configuración y Administración de una red LAN.

Para dar cumplimiento al objetivo general planteado de este trabajo se definieron las siguientes **tareas de investigación:**

1. Elaboración del marco teórico a través del estudio del estado del arte en la actualidad sobre los Laboratorios Virtuales y las técnicas de compilación.
2. Análisis y selección de las herramientas, lenguaje y metodología a utilizar para el desarrollo del módulo.
3. Análisis y diseño del módulo.

4. Elaboración de la propuesta de solución.
5. Implementación de la solución propuesta.
6. Validación de la de solución.

Durante el desarrollo de la investigación se hará uso de los siguientes **métodos de investigación:**

Métodos Teóricos:

- ✓ Analítico-Sintético: Se hace uso de este método para buscar información relacionada con la investigación, extrayendo los elementos más importantes que se relacionan con el objeto de estudio de la misma.
- ✓ Modelación: Se utiliza para representar el conocimiento acumulado durante la investigación, así como los diferentes diagramas que apoyarán el proceso de desarrollo de la herramienta, como el diagrama de casos de uso y diagramas de clases.
- ✓ Histórico-Lógico: Este método se emplea para profundizar en el estado del arte del desarrollo de intérpretes semánticos y ver cómo han ido evolucionando desde su surgimiento, además de realizar un estudio crítico de trabajos relacionados con el objeto de la investigación.

Métodos Empíricos:

- ✓ Consulta de las fuentes de información: Se emplea para seleccionar la información necesaria para realizar el marco teórico.
- ✓ Consulta de especialistas: Se utiliza para recibir los criterios de validación sobre la aplicabilidad y utilidad de lo realizado.
- ✓ Pruebas: Para validar los resultados obtenidos con la solución propuesta. Dentro de estas tenemos las pruebas de caso de uso.

Para conocer de manera general sobre el tema de investigación se presenta una breve descripción de los capítulos que la conforman:

Capítulo 1: Fundamentación Teórica.

Explica los fundamentos sobre el tema de las redes, los distintos servicios que se desean configurar e importancia de los mismos, el proceso de compilación y el análisis

de las fases del el proceso de compilación. Además se describen las posibles herramientas, metodologías así como los lenguajes de programación a utilizar en el desarrollo de la solución.

Capítulo 2: Descripción de la Propuesta de Solución.

Se describe las herramientas seleccionadas para el desarrollo y enumera los requisitos funcionales y no funcionales contenidos en la solución propuesta. También se muestra el modelo de dominio y la descripción de los casos de uso

Capítulo 3: Análisis y Diseño.

Se muestran los diagramas de clases del análisis y el diseño agrupados por paquetes, los diagramas de secuencia así como las descripciones de las clases de diseño.

Capítulo 4: Implementación y Validación.

Se procede a la implementación del sistema de acuerdo con los resultados del análisis y el diseño elaborado anteriormente, se exponen además, algunos resultados de la aplicación integrada al Laboratorio Virtual de Configuración y Administración de una red LAN.

Anexos: En esta sección se incluyen imágenes del módulo integrado al Laboratorio Virtual.

Glosario de Términos: Se elaboró un Glosario de Términos con el objetivo de facilitar la comprensión del lenguaje utilizado.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA.

El presente capítulo tiene como objetivo abordar los elementos relacionados con la investigación, lo concerniente a los Laboratorios Virtuales como nuevas herramientas de apoyo a las asignaturas, las características fundamentales de las redes LAN y los principios del funcionamiento de un compilador mostrando las diferentes fases contenidas en el proceso de compilación para un intérprete. Además se da una breve descripción de las posibles herramientas, lenguajes y metodologías a utilizar el desarrollo del sistema.

1.1. Laboratorios Virtuales.

Los LV son herramientas flexibles desde el punto de vista pedagógico, que apoyan la concepción del aprendizaje constructivista, por lo que están centrados en el alumno como sujeto de su propio aprendizaje. Son elementos claves en la formación integral, no se puede pensar en un ingeniero que no haya realizado prácticas de laboratorio en su trayectoria de formación. Hoy en día podemos dividir el desarrollo de LV en varios tipos: Laboratorios Virtuales desktop, Laboratorios Virtuales remotos y Laboratorios Virtuales para la web siendo estos últimos los que más se desarrollan [2].

Entre las ventajas de los Laboratorios Virtuales podemos mencionar que [2]:

- ✓ Son menos costosos que un Laboratorio Tradicional y se pueden repetir las prácticas en condiciones idénticas, algo difícil de lograr en la realidad, hasta alcanzar el objetivo deseado.
- ✓ Elimina los riesgos que siempre existen en la interacción con la realidad tanto para los instrumentos como los estudiantes, lo que da confianza al estudiante en la realización de la práctica de laboratorio.
- ✓ Al hacer las prácticas de laboratorio varias veces, el estudiante puede tener una retroalimentación inmediata de la práctica.
- ✓ Permiten que los estudiantes realicen las prácticas luego de los horarios de clase lo que apoya al auto aprendizaje y ayuda a crear habilidades.

Entre las desventajas que encontramos en los Laboratorios Virtuales tenemos que:[5]:

- ✓ Si los estudiantes no realizan las prácticas como dicen las indicaciones, con el fin de alcanzar un objetivo propuesto, puede ser que estos solo sean un espectador y las prácticas no contribuyan a su formación.
- ✓ No siempre se pueden hacer las simulaciones requeridas, por lo difícil que son de recrear ciertos experimentos, lo que sin duda nunca serian como en un LT y aunque se parezca bastante a la realidad siempre habrá sensación de que hay pérdida de realismo.

Hoy en día existen muchos Laboratorios Virtuales, como por ejemplo en la Universidad Nacional de Educación a Distancia (UNED) en Madrid, donde se desarrollan Laboratorios Virtuales para la enseñanza de la Física [7].

La Universidad Politécnica de Cataluña, Barcelona donde se pusieron en marcha mediante este experimento virtual, estudios de aspectos teóricos-prácticos en el dominio de campos magnéticos.

La Revolución virtual llegó también a las universidades cubanas. La computación ha resuelto la carencia de las instalaciones de laboratorios reales, insuficientes en todo el país a raíz del bloqueo económico.

En la Universidad Central “Marta Abreu” de las Villas, en Villa Clara se desarrolló el Laboratorio de Física General a distancia para los alumnos de las carreras tecnológicas de los centros de educación superior.

También en la Universidad virtual del Ministerio de Ciencia Tecnología y Medio Ambiente (CITMA) como resultado de la virtualización, se realizaron Laboratorios Virtuales abarcadores sobre los temas de Estadística descriptiva y no paramétrica, Cálculo diferencial integral, Matemática Numérica Probabilidades y Combinatoria, Cinemática, Dinámica y Leyes de conservación, Estática y Mecánica de los fluidos. La simulación se usa preferentemente cuando el experimento real es para la enseñanza de la Hidráulica en las carreras universitarias y tecnológicas de las especialidades de Ingeniería Civil, Mecánica, Química e Hidráulica, en la CUJAE, se concibió un Laboratorio Virtual con 22 prácticas, con datos sobre diferentes esferas como Mecánica de los Fluidos, Máquinas Hidráulicas, Hidrometría, entre otros.

En el proyecto se presentan las experiencias obtenidas tras varios años de aplicación de los Laboratorios Virtuales, así como las características y ventajas que los mismos tienen en la formación del estudiante y en la calidad de la docencia [6].

1.1.1. Soluciones existentes de Laboratorios Virtuales de redes.

Son varios los Laboratorios Virtuales desarrollados en el mundo sobre configuraciones de las redes principalmente sobre la web por ejemplo:

La Implementación de un Laboratorio Virtual de redes en la Facultad de Ingeniería y Ciencias Hídricas de Universidad Nacional del Litoral en Argentina que posee cuatro laboratorios destinados a la carrera de Ingeniería en Informática, los cuales son utilizados para el desarrollo de trabajos prácticos en las materias Redes y Comunicaciones I y II [3].

En las bibliografías consultadas no se encontró temas relacionados con el desarrollo de Laboratorios Virtuales desktop para redes LAN en Cuba.

En la Universidad de las Ciencias Informáticas solo se conocen investigaciones sobre propuestas de arquitecturas para el desarrollo de Laboratorios Virtuales de Redes en la Web llevado a cabo por la Facultad4 [4].

1.2. Características de las redes LAN.

Una red LAN está conformada por un grupo de ordenadores conectados entre sí, estas tienen un área limitada como puede ser un edificio o un conjunto de ellos como por ejemplo los centros educativos. Cada ordenador que se encuentra en esta red es llamado nodo. Las redes LAN se pueden conectar entre ellas a través de líneas telefónicas y ondas de radio lo que formaría entonces una red de área amplia (WAN). Las LAN facilitan compartir información y dispositivos entre los que se destacan las impresoras y los módems. Éstas permiten la comunicación entre los usuarios mediantes e-mail o chat por lo que podemos decir que su principal objetivo es el intercambio de información y compartir recursos. Son redes fáciles de administrar y generalmente operan a velocidades entre 10 y 100 Mbps [8].

1.3. Servicios de una red LAN.

Diariamente se suman a los internautas que existen hoy en día un gran número de usuarios pertenecientes a todas las latitudes de mundo, esto se hace posible mediante una serie de servicios que se brindan a los usuarios que pertenecen a una red u obtienen direcciones publicas mediante un proveedor de Internet. Un servicio muy importante es el de obtener una dirección IP la cual va a identificar a nuestro ordenador a lo largo de la red. Las redes desde sus inicios se fueron pensadas con el fin de conectar usuarios y compartir archivos por lo que en la actualidad hay que hablar sobre dos servicios importantes que permiten que esto se haga realidad ellos son FTP y Samba. Con el desarrollo y las ventajas de la Internet han surgido varios sitios web a los cuales los usuarios se conectan diariamente, sería muy difícil tener que recordar cada dirección IP de los servidores a los cuales nos conectamos y esto no es necesario debido a que en la mayoría las redes hay un servidor DNS prestando servicios. Otro servicio muy importante es el que nos brinda el Proxy el cual permite y regula las conexiones desde las Intranets hacia Internet.

1.3.1. Servicio de Dirección IP.

Las direcciones IP representan un identificador numérico que se le asigna a un ordenador en una red que utilice el protocolo IP. Estas pueden cambiar varias veces al día según sea necesario, se clasifican en dos tipos en cuanto asignación, dinámicas y fijas, como su nombre lo dice las dinámicas pueden variar mediante la asignación de IP dinámica por el protocolo de configuración dinámica de host (DHCP) y las fijas son configuradas por el usuario de manera manual, no varían y son otorgadas principalmente a servidores que brindan algún tipo de servicio como DHCP, DNS o albergan sitios web. Las direcciones IP se refieren a las redes y no a las computadoras en sí, por lo que si se mueve un ordenador de una red a otra, su dirección IP va a cambiar [9].

Cada dirección IP está formada por un par (**netid, hostid**), donde **netid** identifica la red y **hostid** identifica al anfitrión dentro de esta red. Por lo tanto los bits de una dirección de todos los anfitriones en una misma red comparten un prefijo común. Esta división de las direcciones IP en dos partes es para realizar el enrutamiento de manera más eficiente [10].

Dependiendo del número de host que se necesiten para cada red, las direcciones se han dividido en clases primarias: A, B, C, D y E; las de clase D está formada por direcciones que no pertenecen a un host sino a un conjunto de ellos y la clase E no se utiliza ya que son direcciones reservadas.

Tipo	*Dirección mas baja*	*Dirección mas alta*
A	1.0.0.0	126.0.0.0
B	128.1.0.0	191.255.0.0
C	192.0.1.0	223.255.255.0
D	224.0.0.0	239.255.255.255
E	240.0.0.0	247.255.255.255

Figura 1.1 Clasificación de direcciones IP.

Las direcciones IP también se pueden clasificar en públicas y privadas. Las públicas pueden ser accesibles desde cualquier otro ordenador conectado a Internet y las privadas solo son accesibles desde cualquier otro host de su propia red o de otras redes privadas interconectas mediantes routers. Las direcciones IP privadas pueden salir a Internet mediante routers o proxy que contengan una dirección IP pública y se encuentran en los rangos de:

- ✓ Clase A: 10.0.0.0 a 10.255.255.255 (8 bits red, 24 bits hosts). Uso para Intranet
- ✓ Clase B: 172.16.0.0 a 172.31.255.255 (12 bits red, 20 bits hosts). 16 redes clase B contiguas, uso en universidades y grandes compañías.
- ✓ Clase C: 192.168.0.0 a 192.168.255.255 (16 bits red, 16 bits hosts). 256 redes clase C contiguas, uso de compañías medias y pequeñas, además de pequeños proveedores de internet.

La dirección IP que tiene asignado un ordenador en un momento determinado, tiene que ser diferente a todas las demás que están vigentes en el conjunto de la red visible por el host. En el caso de Internet no deben existir dos ordenadores con IP públicas iguales. Pero si pueden ser iguales en caso de pertenecer a redes diferentes y sin ningún camino de comunicación.

1.3.2. Servicio DNS.

DNS nos permite no tener que recodar cada dirección IP de la página o servidor que se desea visitar si conocemos su nombre de dominio.

La función principal del DNS es el mapeo de nombres a direcciones IP, por ejemplo la dirección IP del sitio web www.cubasi.cu es 200.55.129.8 y la mayoría de los internautas para acceder a ella teclea en un navegador web la dirección web www.cubasi.cu y no la dirección IP ya a que es más fácil de recordar [11]. Un DNS está compuesto por tres partes fundamentales:

- ✓ Cliente DNS.
- ✓ Servidor DNS.
- ✓ Zona de autoridad.

Cliente DNS*:* En cliente DNS es el usuario que realiza la petición preguntando por el nombre de algún dominio en Internet.

Servidor DNS: Existen tres tipos de servidores: Servidor Maestro, Esclavo y de Caché. Los Servidores Maestros son los encargado de responder la petición realizada por el cliente y es el que conoce los registros de las zonas originales y de autoridad. Los Servidores Esclavos tienen cierta facultad para responder las peticiones de los clientes pero obtienen sus respuestas desde los Maestros.

Servidor Caché: Ofrece servicios de resolución de nombres como respuesta a las peticiones hechas por los clientes de DNS, dichos servicios de resolución de nombres son guardados para poder acceder a dicha información más rápidamente. Este tipo de servidores no tiene ninguna autoridad sobre las zonas de autoridad. Los servidores DNS son los encargados de hacer las consultas producto de las peticiones solicitadas por los clientes DNS. Para ello el servidor DNS hace uso de 2 tipos de consultas:

- ✓ Consultas Iterativas [11].
- ✓ Consultas Recursivas [11].

La diferencia de las consultas radica en que cuando son iterativas quien asume toda la carga es el cliente y cuando es recursiva la asume el servidor.

Otra parte fundamental de los DNS es la información de la **Zona de Autoridad**, éstas permiten al servidor maestro cargar la información de una zona. Cada Zona de Autoridad abarca al menos un dominio y posiblemente sus sub-dominios, si estos últimos no son delegados a otras zonas de autoridad. La información de cada Zona de Autoridad es almacenada de forma local en un fichero en el Servidor DNS.

1.3.3. Servicio FTP.

EL protocolo FTP es muy utilizado en la red de Internet. Es fundamental para la transferencia de grades bloques de datos por la red. El mismo es ofrecido por la capa de Aplicación del modelo de capas de TCP/IP, utiliza normalmente el puerto de red 20 y 21 y se basa en la arquitectura Cliente/Servidor por lo hace uso de dos componentes fundamentales:

- ✓ **Un Cliente FTP:** Cuando un navegador no está equipado con la función FTP, o si se quiere cargar archivos en un ordenador remoto, se necesitará utilizar un programa cliente FTP. Un cliente FTP es un programa que se instala en el ordenador del usuario, y que emplea el protocolo FTP para conectarse a un servidor FTP y transferir archivos, ya sea para descargarlos o para subirlos.
- ✓ **Un Servidor FTP:** Un servidor FTP es un programa que se ejecuta en un equipo servidor normalmente conectado a Internet (aunque puede estar conectado a otros tipos de redes). Su función es permitir el intercambio de datos entre diferentes servidores y ordenadores. Las aplicaciones más comunes de los servidores FTP suelen ser el alojamiento web, en el que sus clientes utilizan el servicio para subir sus páginas web y sus archivos correspondientes, o como servidor de backup (copia de seguridad) de los archivos que pueda tener una empresa [12].

Un problema básico de éste protocolo es que está pensado para ofrecer sus servicios a máxima velocidad de conexión pero no a máxima seguridad, debido a que el intercambio de información no se realiza de forma cifrada lo que podría causar que algún posible atacante se apoderase de los archivos transferidos. Este problema se soluciona mediante aplicaciones como SCP (protocolo de copia segura) y SFTP (protocolo de trasferencia de archivos seguros), incluidas en el paquete SSH (intérprete de órdenes segura) que permite transferir archivos pero cifrando todo el tráfico de la información.

La llegada de la red de redes (World Wide Web) y los navegadores hizo que ya no fuera necesario conocer los complejos comandos de FTP, el cual se puede utilizar escribiendo la dirección del servidor al que se quiere conectar en el navegador web, indicando con **ftp://** que se va a contactar con un servidor [12].

El acceso a servidores FTP se controla mediante usuarios que poseen los distintos privilegios permitiéndoles así poder modificar archivos existentes o subir sus propios archivos, por lo que el servidor guarda la información de las distintas cuentas de usuarios. Entre los avances de FTP se puede mencionar el Cliente FTP basado en web, mediante el cual se puede acceder al servidor FTP remoto como si se estuviera realizando cualquier otro tipo de navegación web, permitiendo las mismas acciones de crear, copiar, renombrar, eliminar archivos y directorios, cambiar permisos, editar, ver, subir, descargar archivos, así como cualquier otra función del protocolo FTP que el servidor remoto permita. Los servidores FTP admiten dos modos de conexión del cliente: modo pasivo o activo. Además es importante conocer el tipo de transferencia de archivos que se va a utilizar con el fin de no dañar la información, estas puedes ser:

- ✓ Ascii: Adecuado para transferir archivos que sólo contengan caracteres imprimibles (archivos ASCII, no archivos resultantes de un procesador de texto), por ejemplo páginas HTML (lenguaje de marcado de hipertexto), pero no las imágenes que puedan contener.
- ✓ Binario: Es usado cuando se trata de archivos comprimidos, ejecutables para computadoras, imágenes y archivos de audio.

Una de las importancias de FTP es que en la mayoría de servidores, es la única manera de conectar con los sitios web, para así poder subir y bajar archivos.

1.3.4. Servicio Proxy.

Un servidor Proxy es un programa que funciona como intermediario entre el cliente (Mozilla, Internet Explorer) y un servidor web que contiene información que se quiere obtener. Éste permite a otros equipos conectarse a una red de forma indirecta a través de él. Cuando un equipo de la red desea acceder a una información o recurso, es realmente el Proxy quien realiza la comunicación y a continuación traslada el resultado al equipo inicial. En unos casos esto se hace así porque no es posible la comunicación directa y en otros casos porque el Proxy añade una funcionalidad adicional, que no es más que la de mantener los resultados obtenidos por ejemplo una página web en una caché que permita acelerar sucesivas consultas coincidentes [13]. El Proxy brinda diferentes ventajas como son ahorro, filtrado, velocidad, modificación y anonimato.

Por estas ventajas se puede controlar a los usuarios, restringir o dar permiso solo al Proxy. También filtrar los sitios a los que pueden acceder o no los usuarios, modificar datos según la conveniencia y aumentar la velocidad de repuesta dado que se guarda las respuestas de las peticiones realizadas anteriormente, para darla a otro usuario si lo desea. El Proxy cuenta con algunas desventajas entre las cuales podemos mencionar el abuso, la carga e intromisión. Los Proxy muchas veces son sobrecargados debido a que regulan el trabajo de muchos usuarios. Como son un paso más entre el origen y el destino pueden causar intromisión, a pesar de que algunos usuarios puedan no querer pasar por él y menos si hace salva de la información.

Para la web existe el Proxy Web. Se trata de una aplicación específica para el acceso a la web, aparte de la utilidad general de un Proxy, estos proporcionan una caché para las páginas web y los contenidos descargados, que es compartida por todos los equipos de la red, con la consiguiente mejora en los tiempos de acceso para consultas coincidentes. Al mismo tiempo libera la carga de los enlaces hacia Internet [14].

Los Proxy se pueden clasificar como transparente, inverso o abierto. Los transparentes trabajan de manera que las conexiones son enrutadas dentro del Proxy sin configuración por parte del cliente, y habitualmente sin que el propio cliente conozca de su existencia. Éste es el tipo de Proxy que utilizan los proveedores de servicios de internet. El Inverso es un servidor Proxy instalado en el domicilio de uno o más servidores web y todo el tráfico entrante de Internet y con el destino de uno de esos servidores web pasa a través de él ofreciendo ventajas de:

- ✓ Seguridad: el servidor Proxy es una capa adicional de defensa y por lo tanto protege los servidores web.
- ✓ Distribución de Carga: Puede distribuir la carga entre varios servidores web. En ese caso, él puede necesitar reescribir las direcciones de cada página web según en qué servidor se encuentre la información solicitada.
- ✓ Caché de contenido estático: Puede descargar los servidores web almacenando contenido estático como imágenes u otro contenido gráfico.

Los abiertos permiten peticiones de cualquier ordenador que se encuentre o no en su red, este uso es muy beneficioso, pero al aplicarle una configuración "abierta" a todo internet se convierte en una herramienta para su uso indebido como puede ser el

envío de correos spam por los que muchos servidores deniegan acceso a estos Proxy y sus servicios [15].

1.3.5. Servicio Samba.

Samba es una implementación del protocolo de archivos compartidos de Microsoft Windows que permite que ordenadores con GNU/Linux, MacOS X o Unix en general se vean como servidores o actúen como clientes en redes de Windows. Especialmente Samba consiste en dos programas denominados **smbd** y **nmbd**. Ambos programas utilizan el protocolo NetBIOS (Network Basic Input/Output System) para acceder a la red, por lo cual pueden conversar con ordenadores de Windows. Mediante el uso de estos programas Samba puede ofrecer varios servicios al igual que Windows entre ellos encontramos:

- ✓ Compartir uno o más sistemas de archivos.
- ✓ Compartir impresoras, instaladas tanto en el servidor como en los clientes.
- ✓ Ayudar a los clientes, con visualizador de clientes de red.
- ✓ Autentificar clientes contra un dominio Windows.

El programa **smbd** se encarga de ofrecer servicios de acceso remoto a archivos e impresoras, implementando para ello el protocolo SMB. También ofrece los dos modos de compartición de recursos que tiene Windows, basado en usuarios y basado en recursos. En el caso del modo basado en usuarios el acceso a recursos se realiza en función de los nombres de usuarios que se encuentran registrados en un dominio y en el modo basado en recursos es mediante contraseñas que se le asignan a los recursos y conociendo éstas es que se puede acceder.

El programa **nmbd** permite que el sistema Unix participe en el mecanismo de resolución de nombre de Windows, además hace posible la contestación a peticiones de resolución de nombres y el anuncio a recursos compartidos. De esta manera los sistemas GNU/Linux, Mac OS X o Unix aparecen en el Entorno de Red como cualquier sistema Windows ofreciendo servicios al resto de la red.

Samba ofrece varias utilidades, entre las más relevantes se encuentran:

- ✓ Smbclient: posee una interfaz similar a la utilidad FTP que permite a un usuario de

otro sistema que no sea Windows conectarse a recursos SMB, listar, trasferir y enviar ficheros.

- ✓ Swat: (Herramienta de Administración Web Samba). Permite la configuración de manera local o remota utilizando un navegador web.
- ✓ Sistemas de ficheros SMB para Linux: Esta utilidad permite montar recurso SMB en jerarquía de Linux al igual que sucede con los directorios compartidos vía en Windows [16].

Samba actualmente es la implementación libre que permite que varios sistemas puedan convivir en una misma red ofreciendo servicios a los que pertenecen a la misma.

1.4. ¿Qué es un Intérprete?

Los ordenadores y los humanos utilizan diferentes lenguajes para comunicarse entre sí, para poder establecer una comunicación hombre-máquina es necesario que exista como mediador de la comunicación un traductor. Un traductor es cualquier programa que toma como entrada un texto escrito en un lenguaje llamado fuente y da como salida otro texto en un lenguaje, denominado objeto [1].

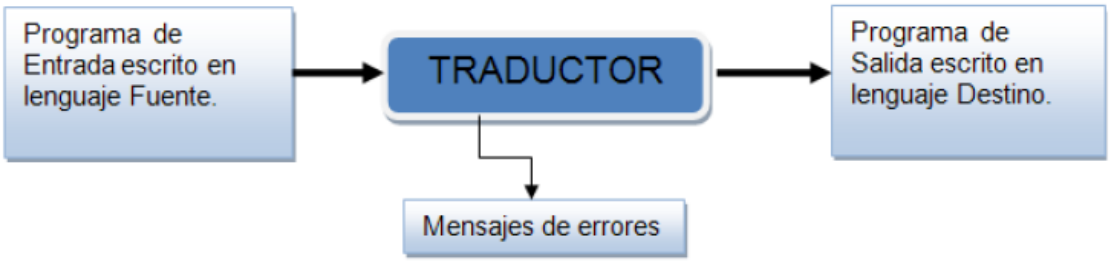

Figura 1.2 Esquema de un Traductor.

Entre los traductores podemos encontrar a los intérpretes, estos no generan un programa equivalente, sino que toman una sentencia del programa fuente en un lenguaje de alto nivel y lo traducen al código equivalente y al mismo tiempo lo ejecutan. En principio, cualquier lenguaje de programación se puede interpretar o compilar, pero se puede preferir un intérprete a un compilador dependiendo del lenguaje que se esté usando y de la situación en la cual se presenta la traducción.

Los intérpretes se utilizan con frecuencia en situaciones relacionadas con la enseñanza o con el desarrollo de software, donde los programas son probablemente traducidos y vueltos a traducir muchas veces. Históricamente, con la escasez de memoria de los primeros ordenadores, se puso de moda el uso de intérpretes frente a los compiladores, pues el programa fuente sin traducir y el intérprete juntos daban una ocupación de memoria menor que la que utilizaban los compiladores. Es por ello los primeros ordenadores personales iban siempre acompañados de un intérprete.

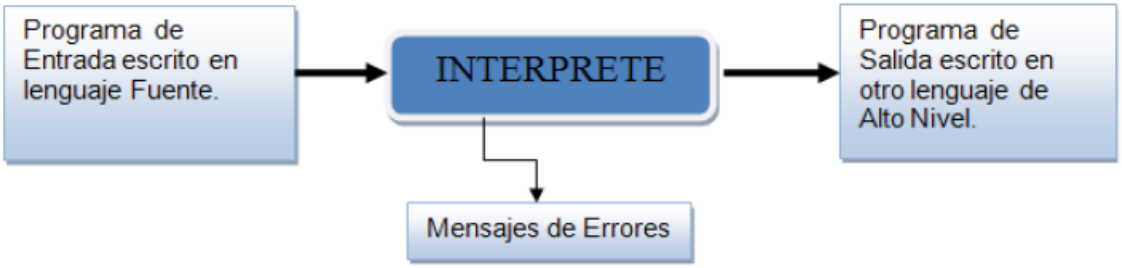

Figura 1.3 Esquema de un Intérprete.

La mejor información sobre los errores por parte del compilador así como una mayor velocidad de ejecución del código resultante, hizo que poco a poco se impusieran los compiladores sobre los intérpretes. Hoy en día, y con el problema de la memoria prácticamente resuelto, se puede hablar de un gran predominio de los compiladores frente a los intérpretes.

Se puede decir que se está frente a un compilador cuando se tiene de entrada un lenguaje fuente escrito en un lenguaje de programación de alto nivel y se obtiene un programa objeto en un lenguaje de bajo nivel (ensamblador o código de máquina), a dicho traductor se le denomina compilador [17].

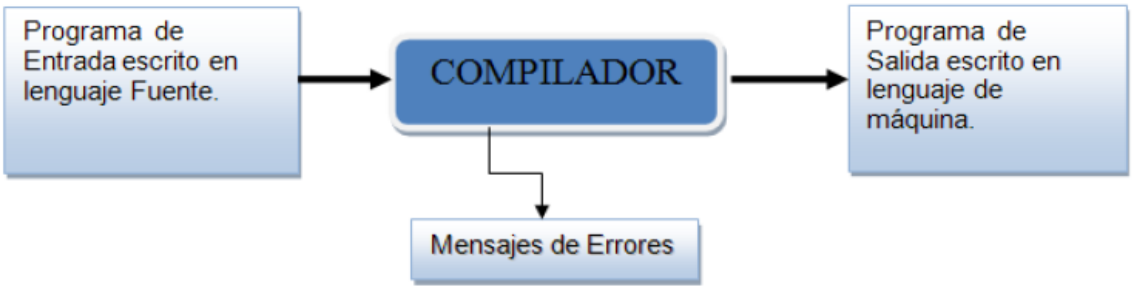

Figura 1.4 Esquema de un Compilador.

En la década del 50 se consideró a los compiladores programas difíciles de escribir. El primer compilador para FORTRAN (Formuale Translator) que es un lenguaje que permitió escribir fórmulas matemáticas de manera traducible por un ordenador, tardó 18 años-persona para realizarse y éste era muy sencillo. El actual desarrollo de compiladores e intérpretes se ve muy influenciado en los distintos avances realizados que permiten hoy en día la implementación de los mismos de una manera más fácil, entre ellos es importante mencionar los aportes de Chomsky en las gramáticas de libre contexto y las propuestas de Rabin y Scott sobre los autómatas deterministas y no deterministas para el reconocimiento léxico de los lenguajes [17].

1.4.1. Conceptos básicos para intérpretes y compiladores.

Cuando se habla de compiladores o intérpretes hay algunos conceptos que se deben dominar para un mejor entendimiento de todo lo relacionado con el tema. A continuación se formulan una serie de definiciones importantes a conocer para todos los que en algún momento se enfrasquen en la construcción de un compilador o un intérprete [1] y [18].

DEFINICIÓN 1.0 Se denomina alfabeto (Σ) a un conjunto arbitrario, finito y no vacío, de símbolos. Entre los alfabetos más comunes se pueden mencionar el alfabeto binario {0, 1} y el alfabeto latino {a, b, c, d, e, f, g,. . .}.

DEFINICIÓN 1.1 Una cadena vacía es una cadena sin elementos, y se denota por ε.

DEFINICIÓN 1.2 Una cadena sobre un determinado alfabeto Σ se define como sigue:

1. ε (cadena vacía) es una cadena sobre Σ.

2. Si x es una cadena sobre Σ, y a ЄΣ entonces xa es una cadena sobre Σ, para todo a ЄΣ.

4. Solo pueden obtenerse cadenas sobre Σ, aplicando 1 y 2.

DEFINICIÓN 1.3 Un lenguaje sobre Σ es un conjunto de cadenas sobre dicho alfabeto que obedece una cierta regla de formación.

DEFINICIÓN 1.4 Una gramática es un cuádruplo G = {N, Σ, P, S} donde:

N: Conjunto finito no vacío de símbolos no terminales.

Σ: Conjunto finito no vacío, disjunto de N, de símbolos terminales.

P: Conjunto de Reglas de Producción.

S: Axioma o símbolo distinguido.

DEFINICIÓN 1.5 Se denomina token o lexema a la unidad básica de un compilador, que puede representar un símbolo o conjunto de símbolos con un significado semántico.

DEFINICIÓN 1.6 Se denomina Léxico a un conjunto de palabras que forman parte de un lenguaje específico.

DEFINICIÓN 1.7 Se denomina Sintaxis a un conjunto de reglas necesarias para construir frases correctas en un lenguaje.

DEFINICIÓN 1.8 Se denomina Semántica al significado de frases generadas por la sintaxis y el léxico.

DEFINICIÓN 1.9 Se denomina Patrones a las reglas que genera la secuencia de caracteres que pueden representar un componente léxico.

DEFINICIÓN 2.0 Se denomina Lexema a la cadena de caracteres que concuerda con un patrón y describe un componente léxico.

DEFINICIÓN 2.1 Se denomina Expresiones Regulares a los métodos por medio de los cuales se pueden realizar búsquedas dentro de cadenas de caracteres, sin importar si la búsqueda requerida es de dos caracteres en una cadena de diez o si es necesario encontrar todas las apariciones de un patrón definido de caracteres en un archivo de millones de caracteres.

DEFINICIÓN 2.2 Una gramática es ambigua si derivando de forma diferente con el mismo tipo de derivación se llega al mismo resultado.

1.4.2. Características de los intérpretes y compiladores.

Los interpretes comparten muchas de sus operaciones con los compiladores, y ahí pueden incluso ser traductores híbridos, de manera que quedan en alguna parte entre los intérpretes y los compiladores. Existen lenguajes cuyos traductores se idearon como intérpretes y otros como compiladores.

No obstante, para un lenguaje dado, pueden existir tanto compiladores como intérpretes. Los intérpretes y compiladores tienen diversas ventajas e inconvenientes que los hacen complementarios:

- ✓ Un intérprete facilita la búsqueda de errores, pues la ejecución de un programa puede interrumpirse en cualquier momento para estudiar el entorno (valores de las variables, etc.). Además, el programa puede modificarse sobre la marcha, sin necesidad de volver a comenzar la ejecución.

- ✓ Un compilador suele generar programas más rápidos y eficientes, ya que el análisis del lenguaje fuente se hace una sola vez, durante la generación del programa equivalente. En cambio, un intérprete se ve obligado generalmente a analizar cada instrucción tantas veces como se ejecute.

- ✓ Un intérprete permite utilizar funciones y operadores más potentes, como por ejemplo ejecutar código contenido en una variable en forma de cadenas de caracteres. Usualmente, este tipo de instrucciones es imposible de tratar por medio de compiladores. Los lenguajes que incluyen este tipo de operadores y que por tanto exigen un intérprete, se llaman interpretativos. Los lenguajes compilativos, que permiten el uso de un compilador, prescinden de este tipo de operadores.

- ✓ En los intérpretes el programa se puede ejecutar de inmediato, sin esperar a ser compilado pero en los compiladores la ejecución es más lenta, pues cada instrucción debe ser traducida a código máquina tantas veces como sea ejecutada.

Comparando su actuación con la de un ser humano, un compilador equivale a un traductor profesional que, a partir de un texto, prepara otro independiente traducido a otra lengua, mientras que un intérprete corresponde al intérprete humano, que traduce de viva voz las palabras que oye, sin dejar constancia por escrito.

Tanto los intérpretes como los compiladores están compuestos por varias fases dentro de su proceso de compilación o interpretación las mismas se describen a partir de la siguiente sección.

1.5. Proceso de Compilación.

Es el proceso por el cual se traducen las instrucciones escritas en un determinado lenguaje de programación a lenguaje objeto, el mismo es aplicable para intérpretes y compiladores. Este cuenta con diferentes fases que se describen a continuación [19].

1.5.1. Fases del Proceso de Compilación.

Conceptualmente un compilador opera por fases, cada una de las cuales trasforma al programa fuente de una representación de la otra [1] y [19].

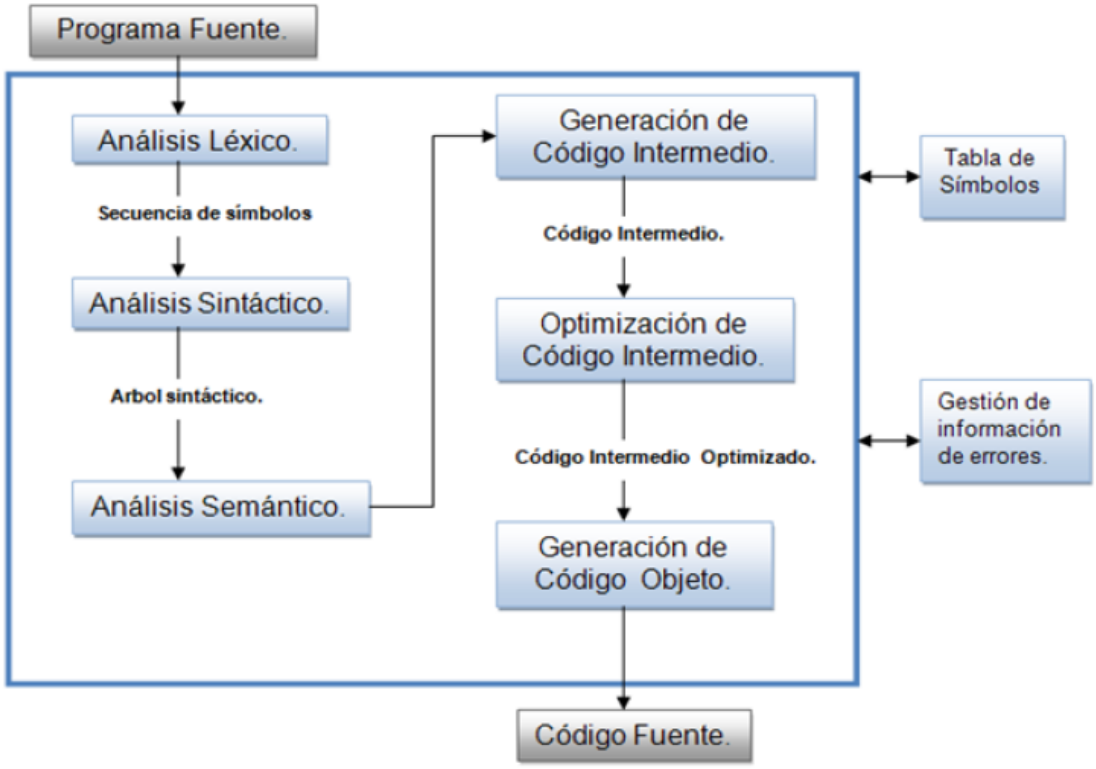

Figura 1.5 Fases de un Compilador o Intérprete.

1.5.1.1. Análisis Léxico.

El analizador léxico o scanner es el encargado de leer el código fuente de entrada, el cual es una secuencia de símbolos pertenecientes al alfabeto de un determinado lenguaje. Éste lee la secuencia de caracteres o símbolos que constituye el programa fuente, carácter a carácter, los mismos son leídos de izquierda a derecha, y se

agrupan en componentes léxicos que tienen un significado colectivo, a estos componentes se les denomina (tokens). Los tokens pueden ser de varios tipos pero los más elementales son [20]:

- ✓ Palabras reservadas del lenguaje. (ej. if, while, do, entre otros).
- ✓ Identificadores. (ej. nombre de variables, nombre de funciones).
- ✓ Operadores (ej. =, +, -, *, <, >, entre otros).
- ✓ Símbolos especiales (ej. [],(), {}).
- ✓ Constantes literales o numéricas (ej. 1085).

Se considera una buena práctica a cada token asignarle una estructura lexicológica, consistente en un par de la forma **<tipo del token, info>,** donde "tipo del token" almacena la categoría lexicológica del token, e "info" el valor del token en particular (ejemplo: el valor de la constante, nombre del identificador, etc.) [19].

El analizador léxico opera bajo petición del analizador sintáctico devolviendo un componente léxico, según el analizador sintáctico lo va necesitando para avanzar en la gramática. Los componentes léxicos son los símbolos terminales de la gramática [20].

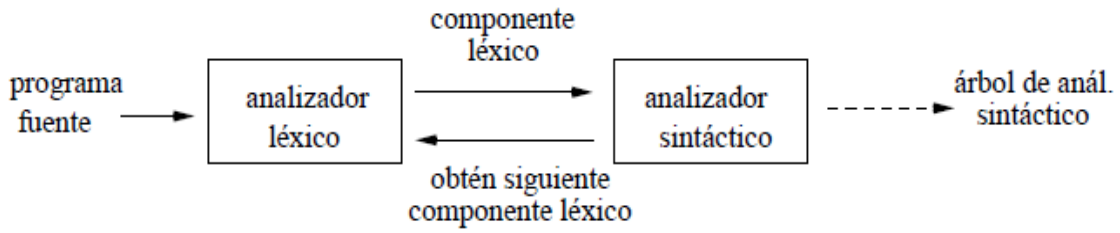

Figura 1.6 Analizador Léxico.

Entre otras funciones que el scanner realiza se encuentran:

- ✓ Manejo de ficheros de entrada del programa fuente (abrirlo, leer caracteres y cerrarlo).
- ✓ Eliminar caracteres no válidos para un token (espacios en banco, comentarios entre otros).
- ✓ Contabilizar número de líneas y columnas para emitir un error.

Si el analizador léxico no reconoce algún lexema en el lenguaje emite un error léxico informando que los símbolos entrados no se corresponden con el lenguaje.

1.5.1.2. Tabla de Símbolos.

Una tarea fundamental en un compilador es la de almacenar los identificadores utilizados en un programa y sus atributos principales, de manera que en cualquier momento pueda conocerse de un identificador, su tipo y alcance. Esta información se almacena generalmente en una estructura conocida como tabla de símbolos, la cual tiene una entrada para cada identificador y sus atributos. Los tokens que representan constantes o identificadores se almacenan en la tabla de símbolos a medida que van apareciendo. Durante todo el proceso de compilación se invierte una parte significativa del tiempo en el manejo de la tabla de símbolos. En la fase de análisis léxico se insertan los símbolos según aparecen en el programa fuente y en las restantes fases se van agregando atributos a medida que se conocen. Además, para cada identificador que se analice, es importante conocer sus atributos, por lo que el acceso a la tabla de símbolos se realiza constantemente. Aquí se hace evidente que las tablas deben organizarse de forma tal que permitan una búsqueda eficiente, es por eso que hay que seleccionar adecuadamente la estructura de datos que represente la misma [19].

1.5.1.3. Análisis Sintáctico.

El análisis sintáctico se puede determinar como una función que toma de entrada la secuencia de componentes léxicos producidas por el analizador léxico (**1.5.1.1**) y produce como salida el árbol sintáctico. Éste se encarga de examinar la secuencia de tokens para determinar si el orden de la secuencia es correcto de acuerdo a las reglas de definición sintáctica del lenguaje [21].

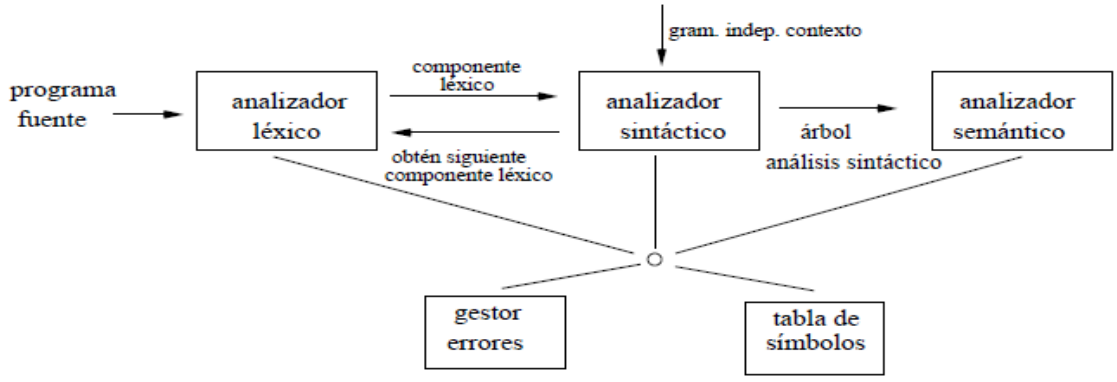

Figura 1.7 Analizador Sintáctico.

Para construir un buen analizador sintáctico se necesita de una gramática bien definida, que no contenga ambigüedades (**DEFINICIÓN 2.2**), debido a que las gramáticas, son las especificaciones sintácticas y precisas de los lenguajes. Este trabaja con la primera componente de la tupa del token.

Si se logra reconocer una configuración completa según las reglas gramaticales entonces la entrada está sintácticamente correcta, por lo que solo quedaría revisar si semánticamente está correctamente escrita.

1.5.1.4. Análisis Semántico.

La fase de análisis semántico revisa el programa fuente tratando de encontrar errores semánticos y reúne la información sobre los tipos para la fase posterior de generación de código. En ella se utiliza la estructura jerárquica determinada por la fase de análisis sintáctico para identificar los operadores, operandos de expresiones y proposiciones.
Esta operación está relacionada con la segunda componente de la tupla devuelta en el análisis léxico (**<tipo del token, info>**). Al comprobar la validez semántica de un programa se pudiera decir que se realiza un reconocimiento sintáctico-semántico, pero los errores semánticos no pueden ser detectados por el analizador sintáctico, puesto que se relacionan con interdependencias entre las diferentes partes de un programa que no son reflejadas en un análisis gramatical [22].
Un componente importante del análisis semántico es la verificación de tipos donde se verifica si cada operador tiene los operandos permitidos por lenguaje fuente [19].

Ejemplo: A continuación se muestra un código escrito en C++, sintácticamente correcto pero semánticamente erróneo:

```
class Suma
{   private int a;
    private string b;
    private int c;
        Public int RealizarSuma();
        { return c= a + b; }
}
```

Como se muestra anteriormente el código esta sintácticamente correcto pero no semánticamente, debido a que el analizador semántico detecta la incompatibilidad de tipos de las variable presentes en la suma (c = a + b), observe que se intentó sumar una cadena de caracteres (**string**), con un literal entero (**int**), lo cual es considerado un error semántico por el compilador y se produce un error semántico [18].

1.5.1.5. Gestión de Errores.

Si los compiladores o intérpretes tuvieran que procesar solamente programas correctos, su diseño e implementación se simplificaría en buena medida. Pero los programadores escriben programas incorrectos frecuentemente, y un buen intérprete o compilador debe ayudar al programador a localizar e identificar los errores. Los errores en un programa pueden clasificarse en 4 grandes grupos [19]:

- ✓ Léxicos.
- ✓ Sintácticos.
- ✓ Semánticos.
- ✓ Lógicos o de programación.

Errores Léxicos: Los errores léxicos son aquellos que ocurren cuando un símbolo es analizado por el scanner y éste no encuentra ningún token válido, por lo que informa que ha ocurrido un error léxico.

Errores Sintácticos: Los errores sintácticos son detectados durante el análisis sintáctico y se producen cuando los tokens no cumplen con las reglas gramaticales del lenguaje.

Errores Semánticos: Los errores semánticos se producen durante el proceso de análisis semántico, detectándose al comprobar que cada operando involucrado en las operaciones sea de un tipo permitido por la misma o según las reglas semánticas que deba cumplir el lenguaje fuente.

Errores Lógicos: Los errores lógicos están relacionados con el cumplimiento de condiciones en un ambiente dado, o con redundancia en el código. Ejemplo de ellos pudieran ser que se hace referencia a una variable no declarada aún. Un intérprete o compilador que se detenga ante el primer error que se encuentre no es muy eficaz. El

tratamiento de los errores en cualquiera de las fases debe cumplir con los siguientes requisitos:

- ✓ Reportar la presencia de los errores de forma clara y precisa.
- ✓ Recuperarse de los errores rápido y ser capaz de continuar parar detectar los errores siguientes.
- ✓ No demorar significativamente el procesamiento de los programas correctos.

1.5.1.6. Generación de Código Intermedio.

Después de los análisis sintácticos y semánticos, algunos compiladores generan una representación intermedia explicita del programa fuente. Se puede considerar a esta representación intermedia como un programa para máquina abstracta. La representación inmediata debe tener dos propiedades importantes, debe ser fácil de producir y fácil de traducir al programa objeto. Las representaciones inmediatas pueden ser de varias formas [1]:

- ✓ Tercetos.
- ✓ Cuartetos.
- ✓ Polaca Inversa.

Las representaciones inmediatas se clasifican en:

- ✓ Formas Intermedias de Alto Nivel: son aquellas que se suelen emplear en las primeras fases de análisis.
- ✓ Formas Intermedias de Nivel Medio: son válidas para representar un conjunto amplio de lenguajes fuente, no siendo dependientes de uno en concreto. Válidas para representar un conjunto extenso de arquitecturas de hardware.
- ✓ Formas Intermedias de Bajo Nivel: permiten traducir a distintos micros de una misma arquitectura, creando una dependencia a ésta.
- ✓ Formas Intermedias Multinivel: Aquéllas que conjugan varias de las características anteriores.

1.5.1.7. Optimización de Código Intermedio.

La fase de optimización se encarga de transformar el código intermedio en un nuevo código de función equivalente pero de menor tamaño o de menor tiempo de ejecución. Algunas de las transformaciones que puede llevar a cabo la fase de optimización son:

- ✓ Eliminar el cálculo de expresiones cuyo valor no se usa.
- ✓ Fundir en uno el cálculo repetido de la misma expresión.
- ✓ Sacar de los lazos las expresiones cuyo valor no cambia en ellos.
- ✓ Reducir el uso de memoria local reutilizando el espacio de una variable muerta.

1.5.1.8. Generación de Código Objeto.

La fase de generación de código objeto se encarga de generar el programa nativo usando el juego de instrucciones específico de la máquina o CPU objeto, y el formato para archivos ejecutables del sistema operativo. Entre otras cosas, también se le asignan direcciones definitivas a las rutinas y variables que componen el programa.

1.6. Metodologías del desarrollo de Software.

Una metodología de desarrollo es una colección de documentación formal referente a los procesos, las políticas y los procedimientos que intervienen en el desarrollo del software. Una metodología es una guía que se sigue, que va indicando qué hacer y cómo se debe actuar cuando se quiere obtener algún tipo de investigación.

En términos informáticos, una metodología de desarrollo de software es un conjunto de procedimientos, técnicas, herramientas y un soporte documental que ayuda a los desarrolladores a realizar nuevos software [23].

Las metodologías se clasifican en robustas y ágiles. Las metodologías robustas son aquellas en las que se hace una fuerte planificación del proyecto durante todo el proceso de desarrollo, encaminadas para proyectos de gran tamaño. Están divididas por etapas en las cuales se genera gran cantidad de documentación. En este tipo de metodologías se realiza además un profundo desarrollo en el análisis y diseño de los sistemas antes de su construcción. Ejemplo de ellas son:

- ✓ RUP: Rational Unified Process (Proceso Unificado de Desarrollo).
- ✓ MSF: Microsoft Solutions Framework (Marco de Soluciones Microsoft).

Las metodologías ágiles por otra parte están encaminadas para proyectos de menos volumen y en los que la documentación no juega el papel más importante como en con en el caso anterior. Son orientadas a las personas y no a los procesos [24]. Ejemplo:

- ✓ XP: Extreme Programming (Programación Extrema).
- ✓ Family of Crystal Methodologies (Familia de Metodologías Crystal).

1.6.1. Programación Extrema (XP).

La Programación Extrema es una metodología ligera de desarrollo de software que se basa en la simplicidad, la comunicación y la realimentación o reutilización del código desarrollado. La metodología consiste en una programación rápida o extrema y es utilizada para proyectos de corto plazo. Esta metodología es basada en pruebas unitarias, la re-fabricación y la programación en pares.

Propuestas de la Programación Extrema:

- ✓ Empieza en pequeño y añade funcionalidad con retroalimentación continua.
- ✓ El manejo del cambio se convierte en parte sustantiva del proceso.
- ✓ El costo del cambio no depende de la fase o etapa.
- ✓ No introduce funcionalidades antes que sean necesarias.
- ✓ El cliente final se convierte en miembro del equipo.

1.6.2. Proceso Unificado de Desarrollo (RUP).

Es un proceso de desarrollo de software, definido como un conjunto de actividades necesarias para transformar los requisitos de un usuario en un sistema de software. Esta metodología es adaptable para proyectos a largo plazo y establece refinamientos sucesivos de una arquitectura ejecutable. RUP es un proceso de desarrollo de software que junto a UML constituyen la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos [25]. En sus características cuenta con que es:

- ✓ Iterativo e Incremental: En el Proceso Unificado los proyectos se entregan en etapas iteradas. En cada iteración se analiza la opinión, la estabilidad y calidad del producto, y se refina la dirección del proyecto así como también los riesgos involucrados. Esta serie de iteraciones trae como resultado un incremento del producto desarrollado que añade o mejora las funcionalidades del sistema en desarrollo.

- ✓ Dirigido por casos de uso: En el Proceso Unificado los casos de uso se utilizan para capturar los requisitos funcionales y para definir los contenidos de las iteraciones. Cada iteración toma un conjunto de casos de uso o escenarios y desarrolla todo el camino a través de las distintas disciplinas: diseño, implementación, prueba, etc.

- ✓ Centrado en la arquitectura: El Proceso Unificado asume que no existe un modelo único que cubra todos los aspectos del sistema. Por dicho motivo existen múltiples modelos y vistas que definen la arquitectura de software de un sistema [25].

1.7. Herramientas CASE.

Las Herramientas de Ingeniería de Software Asistidas por Computadora (CASE) son un conjunto de métodos, utilidades y técnicas que facilitan la automatización del ciclo de vida del desarrollo de sistemas de información, completamente o en alguna de sus fases. Estas herramientas brindan ayuda a los analistas, ingenieros de software y desarrolladores, por ejemplo permiten el modelado de los sistemas mediante diferentes diagramas y generación de código a partir de éstos, y viceversa.

El uso de las herramientas CASE aumenta la calidad del software desarrollado, debido a que una funcionalidad se basa en la comprobación automática de errores. Permite la reutilización de componentes de software. Además de que acelera el proceso de desarrollo del software y permite además un desarrollo gradual e iterativo.

La principal ventaja de la utilización de una herramienta CASE, es la mejora de la calidad de los desarrollos realizados y, en segundo término, el aumento de la productividad. Para conseguir estos dos objetivos es conveniente contar con una organización y una metodología de trabajo, además de la propia herramienta.

1.7.1. Visual Paradigm.

Es una herramienta CASE que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientado a objetos, construcción, pruebas y despliegue. Además soporta todos los diagramas UML, el diagrama entidad-relación ayudando a una rápida construcción de aplicaciones con calidad y a un menor coste. Produce documentación del sistema en formato PDF, HTML y otros. Se puede modelar en varios idiomas.

Entre sus características encontramos:

- ✓ Permite realizar de ingeniería directa e inversa.
- ✓ El modelo y el código permanecen sincronizados en todo el ciclo de desarrollo.
- ✓ Permite integrarse con diferentes Ambientes de Desarrollo Integrado (IDE).
- ✓ Generación de Código: Modelo a código, diagrama a código.

1.7.2. Rational Rose Enterprise Edition.

Es una herramienta CASE privativa, basada en el Lenguaje Unificado de Modelación (UML), que permite crear los diagramas que se van generando durante el proceso de Ingeniería en el Desarrollo del Software.

Rational Rose utiliza un proceso de desarrollo iterativo controlado, donde el desarrollo se lleva a cabo en una secuencia de iteraciones. Cada iteración comienza con una primera aproximación del análisis, diseño e implementación para identificar los riesgos del diseño, los cuales se utilizan para conducir la iteración, primero se identifican los

riesgos y después se prueba la aplicación para que éstos se hagan mínimos. Cuando la implementación pasa todas las pruebas que se determinan en el proceso, ésta se revisa y se añaden los elementos modificados al modelo de análisis y diseño. Una vez que la actualización del modelo se ha modificado, se realiza la siguiente iteración.

Características principales:

- ✓ Admite varias notaciones como UML.
- ✓ Permite desarrollo multiusuario.

- ✓ Genera documentación del sistema.
- ✓ Disponible en múltiples plataformas.

1.8. El Lenguaje Unificado de Modelado UML.

UML ofrece soporte para clases, clases abstractas, relaciones, comportamiento por iteración, empaquetamiento, entre otros. Estos elementos se pueden representar mediante nueve tipos de diagramas, que son: de clases, de objetos, de casos de uso, de secuencia, de colaboración, de estados, de actividades, de componentes y de desarrollo.

UML presenta características generales y razones por las que resulta interesante su aplicación para efectos de la representación de una Arquitectura de Software. Permite el soporte para algunos de los conceptos asociados a las Arquitecturas de Software, como los componentes, los paquetes, las librerías y la colaboración. Además, admite la descripción de componentes en la Arquitectura de Software en dos niveles; se puede especificar solo el nombre del componente o especificar las clases o interfaces que implementan éstos.

Este lenguaje está compuesto por tres elementos fundamentales, los elementos, los diagramas y las relaciones:

- ✓ Elementos: abstracciones que constituyen los bloques básicos de construcción.
- ✓ Diagramas: es la representación de un conjunto de elementos: visualizan un sistema desde diferentes perspectivas.
- ✓ Relaciones: Permiten ligar los elementos.

1.9. Lenguajes de Programación.

Un lenguaje de programación puede usarse para crear programas que controlen el comportamiento físico y lógico de una máquina, para expresar algoritmos con precisión, o como modo de comunicación humana. Está formado por un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones. Al proceso por el cual se escribe, se prueba, se depura, se compila y se mantiene el código fuente de un programa informático se le llama programación. Entre estos lenguajes se pueden encontrar:

- ✓ C# es un lenguaje de programación orientado a objetos desarrollado y estandarizado por Microsoft como parte de su plataforma .NET. Su sintaxis básica deriva de C/C++ y utiliza el modelo de objetos de la plataforma.NET, similar al de Java aunque incluye mejoras derivadas de otros lenguajes.

- ✓ C++ es un lenguaje de programación diseñado a mediados de los años 1980. La intención de su creación fue el extender al exitoso lenguaje de programación C con mecanismos que permitan la manipulación de objetos. Desde el punto de vista de los lenguajes orientados a objetos, el C++ es un lenguaje híbrido. Posteriormente se añadieron facilidades de programación genérica, que se sumó a los otros dos paradigmas que ya estaban admitidos (programación estructurada y la programación orientada a objetos). Por esto se suele decir que el C++ es un lenguaje de programación multiparadigma.

1.10. Entorno de Desarrollo Integrado.

Un IDE es un programa informático compuesto por un conjunto de herramientas de programación. Puede dedicarse en exclusiva a un sólo lenguaje de programación o bien, poder utilizarse para varios. Un IDE es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica. Los IDEs pueden ser aplicaciones por sí solas o pueden ser parte de aplicaciones existentes. Entre ellos encontramos el:

- ✓ Microsoft Visual Studio.NET: es un IDE que cuenta con pequeñas herramientas auxiliares de completamiento de códigos como el Visual Assist lo que ayuda mucho a la codificación de la aplicación. Aunque es una herramienta privativa hay que reconocer que se encuentra en la cima de las herramientas IDE, soporta distintos lenguajes lo que hace que sea muy utilizada por los usuarios de Windows.

- ✓ QT Creator: Es un IDE para el desarrollo de programas mediante las librerías de Qt (Quasar Technologies). Ha alcanzado gran desarrollo en muy poco tiempo logrando situarse junto con Eclipse como IDEs preferidos de Linux.

Conclusiones Parciales.

En este capítulo se reúnen las bases para el conocimiento del módulo que se desarrolla, así como las metodologías y herramientas que se estudiaron para el desarrollo del mismo. Es importante destacar que todo el software que se utilizará es software libre, de código abierto y multiplataforma, por lo que el módulo resultante no necesitará ninguna licencia para su uso y podrá ser portado fácilmente a otros sistemas operativos.

CAPÍTULO 2: DESCRIPCIÓN DE LA PROPUESTA DE SOLUCIÓN.

El presente capítulo tiene como objetivo abordar elementos fundamentales de la solución propuesta y las herramientas que se seleccionaron para ser utilizadas en el desarrollo del sistema. También se muestran los requisitos funcionales y no funcionales del sistema así como la descripción del modelo de dominio y los diagramas de casos de uso.

2.1. Descripción de la Solución.

Luego de la investigación realizada que se describe en el **Capítulo 1**, se decide desarrollar un módulo que logre realizar la validación de las diferentes configuraciones que se harán en el Laboratorio Virtual. Todos los servicios a configurar en la práctica poseen un lenguaje bien definido por una sintaxis, lo que permite dar solución a la problemática de la investigación con el desarrollo de intérpretes que realicen un proceso de compilación. Implementando siempre hasta la fase del análisis semántico, debido a que hasta la implementación de esta fase se logra los objetivos perseguidos en la investigación con el fin de determinar los posibles errores cometidos en las configuraciones de los usuarios. Las restantes fases no son necesarias debido a que no se requiere generar ningún código intermedio, por lo que no hace falta la optimización, ni la generación de éste a código objeto. Para cada intérprete a desarrollar se implementarán los módulos siguientes dentro del proceso de compilación:

Analizador Léxico: El mismo va a realizar la acción de escanear la configuración entrada para ir conformando los distintos componentes léxicos que pertenecen al lenguaje, para esto se hará uso de términos fundamentales como: los tokens (**DEFINICIÓN 1.5**) y los patrones (**DEFINICIÓN 1.9**), en los patrones para identificar las palabras reservadas del lenguaje se utiliza el patrón de secuencia de caracteres que forman la palabra clave y para el caso de identificadores y otros se utiliza haciendo uso de las expresiones regulares (**DEFINICIÓN 2.1**) o los autómatas finitos deterministas **Figura 2.1**. En la tabla a continuación se muestra un ejemplo de lo antes expuesto en el caso de las componentes de una configuración IP.

Token	Lexema	Description del Patrón
tk_Address	"address"	address
tk_iface	"iface"	iface

Tabla 2.1 Ejemplo de los términos que maneja el Analizador Léxico.

Identificadores

```
identifier = letter (letter | digit)*
```

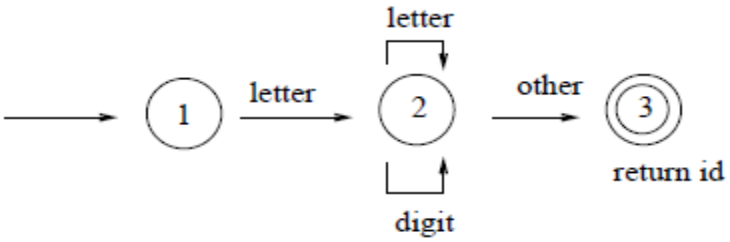

Figura 2.1 Ejemplo AFD para leer un Identificador.

Tabla de Símbolos: En la misma se van a ir guardando tokens que representan constantes o identificadores a medida que aparecen en el lenguaje. Para la tabla se escogió una lista como estructura de dato para guardar la información.

Analizador Sintáctico: En esta fase el intérprete emplea la gramática para ir conformando el AST que se analizará en la próxima fase. Es importante mencionar que la gramática con que trabaja el analizador sintáctico no contiene ambigüedades. Para el caso de los intérpretes a desarrollar las gramáticas confeccionadas se clasifican como regular a la derecha.

Ejemplo: Gramática para reconocer una dirección IP utilizada por el analizador sintáctico.

<IPlinux> :: <head><Body>
<head> :: tk_auto tk_lo tk_iface tk_lo tk_inet tk_loopback tk_auto tk_id
<Body> :: <Configuration> <Configurations>
<Configurations>:: <Configuration> <Configurations> | e
<Configuration>:: tk_iface tk_id tk_inet tk_static
<ADDRESS> <NETMASK> <NETWORK> <BROADCAST> <GATEWAY>

<ADDRESS>:: tk_address <Valor>
<NETMASK>:: tk_netmask <Valor>
<NETWORK>:: tk_network <Valor>
<BROADCAST>:: tk_broadcast <Valor>
<GATEWAY> :: tk_gateway <Valor>
<Valor>:: tk_number tk_point tk_number tk_point tk_number tk_point tk_number

Analizador Semántico: El analizador semántico es una de las fases más importantes en los intérpretes, debido a que es necesario comprobar la validez de las configuraciones que se harán en las prácticas, en el caso específico de una configuración IP el analizador semántico es capaz de reconocer si una dirección pertenece o no a la subred configurada. Para lograr esto se hace un chequeo en el AST verificando si la configuración entrada es válida.

Gestión de Errores: La gestión de errores es la última fase a implementar en los intérpretes, necesario para validar las configuraciones. El mismo va permitir mantener informado al usuario de los distintos errores cometidos durante las configuraciones. Estos errores se van a conformando según se analiza el lenguaje y se avanza en cada una de las fases del análisis.

2.2. Selección del Ambiente de Desarrollo para el módulo.

Para desarrollar del módulo se escogió el siguiente ambiente de desarrollo.

- ✓ Como metodología de desarrollo de software a utilizar se escoge RUP ya que es una metodología para el desarrollo de software orientado a objetos, además de ser la metodología seleccionada para aplicar por el proyecto.
- ✓ Mientras que la herramienta CASE que se selecciona es el Visual Paradigm, que es una herramienta libre muy profesional y multiplataforma que soporta el ciclo de vida completo del desarrollo de software. Además genera código en una amplia gama de lenguajes, entre ellos C++. Posee una interfaz amigable y profesional.
- ✓ El módulo como lenguaje de programación utiliza C++, que es un lenguaje de programación orientado a objeto, lo cual es esencial para incrementar la productividad, calidad y reutilización del software. Ofrece gran riqueza de

operadores y expresiones, flexibilidad, concisión, eficiencia y robustez.

- ✓ Para el diseño de las interfaces se escoge Qt Designer que es muy potente, permite diseñar Interfaces Gráficas de Usuario (GUI) de una forma muy sencilla y con bibliotecas Qt basado en el código abierto. Este software está provisto de herramientas que facilitan la creación de formas, botones, ventanas de diálogo y permite su pre visualización. Qt Designer está disponible para casi todas las plataformas por lo que la aplicación resultante utilizada en cualquier Sistema.
- ✓ Entre los IDE de desarrollo estudiados se escogió QT Creator es libre y multiplataforma. Es importante mencionar que QT Creator posee una extensa biblioteca de clases y herramientas, las cuales se encuentran bien documentadas lo que de gran ayuda en el desarrollo del software.

2.3. Modelo de Dominio.

En este epígrafe se muestra el modelo de dominio, en el mismo se modelan los principales conceptos con los que se trabajarán en el desarrollo del módulo y las relaciones existentes entre ellos.

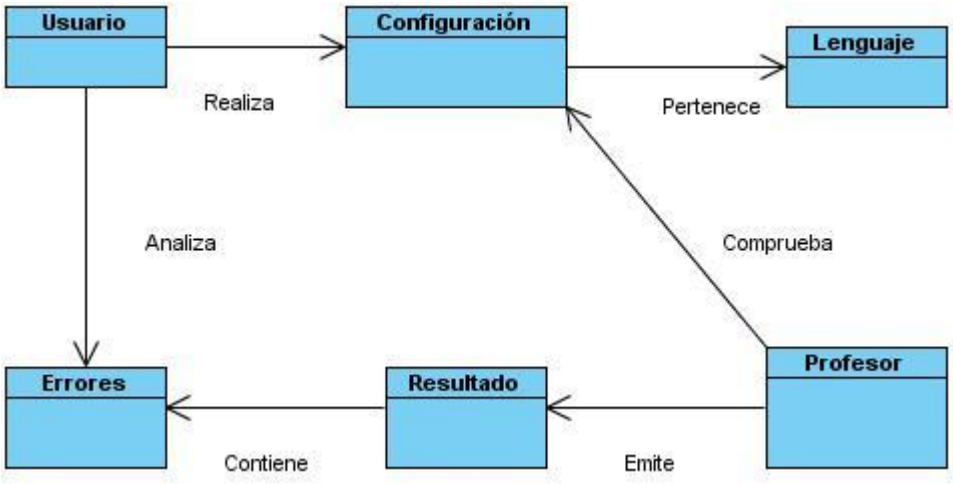

Figura 2.2 Diagrama del Modelo de Dominio.

Glosario de Término del Modelo de Dominio.

A continuación se describen los conceptos presentes en el modelo anterior para facilitar su comprensión.

Usuario: Es el encargado de realizar la configuración de la práctica que se está realizando.

Configuración: Es realizada por los usuarios cuando estos entran a configurar un servicio.

Lenguaje: Una configuración pertenece a un lenguaje en específico, éste varía según el servicio que se va a configurar.

Profesor: Es el encargado de evaluar la configuración realizada en la práctica por el usuario.

Resultado: Es emitido por el profesor como evaluación de la configuración realizada.

Errores: En los resultados se describen los errores que cometió el usuario para su posterior análisis.

2.4. Requerimientos del Sistema.

Un requerimiento es una condición o capacidad que tiene que ser alcanzada o poseída por un sistema o componente de un sistema para satisfacer un contrato, estándar, u otro documento impuesto formalmente. A continuación se muestras los requisitos funcionales y no funcionales del sistema.

2.4.1. Requisitos Funcionales.

Son capacidades o condiciones que el sistema debe cumplir. Los requisitos funcionales se mantienen invariables sin importar con que propiedades o cualidades se relacionen.

R1. Permitir autenticarse.

El sistema debe permitir que el usuario entre sus datos para autenticarse.

R2. Cargar la configuración básica de los ficheros a configurar.

El sistema debe permitir que el usuario vea la configuración que se carga por defecto.

R3. Permitir modificar los ficheros.

El sistema debe permitir al usuario configurar los distintos ficheros.

R4. Permitir comprobar la configuración realizada por los usuarios.

El sistema debe permitir comprobar la configuración realizada por el usuario para ver si es correcta.

R5. Mostrar los errores cometidos en la configuración.

El sistema debe mostrar los errores cometidos al final de la configuración.

R6. Guardar reporte.

El sistema debe permitir al usuario salvar el historial de las configuraciones realizadas.

2.4.2. Requisitos No Funcionales.

Los requisitos no funcionales son propiedades o cualidades que el producto debe tener, debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido o confiable. A continuación se muestra los requisitos no funcionales que debe cumplir el sistema.

- ✓ **Requerimientos de Usabilidad.**
  El sistema deberá tener una interfaz amigable con una buena utilización de los elementos de diseño, con adecuada combinación de colores.

  El sistema deberá ser usado por usuarios capacitados para el uso de la herramienta y que hayan leído previamente el manual de ayuda.

- ✓ **Requerimientos de Soporte.**
  El sistema debe tener alguna documentación o ayuda.

- ✓ **Requerimientos de Hardware del Sistema.**
  Las computadoras que utilizarán el software a desarrollar deberán tener 256 MB de memoria RAM como mínimo.

- ✓ **Requerimientos de Software del Sistema.**
  Las computadoras que utilizarán el software deben tener instalado, Windows XP, Windows Seven o alguna distribución GNU/Linux.

2.5. Modelo de Casos de Uso del Sistema.

En este epígrafe describen los actores del sistema a desarrollar, y se definen los casos de uso del sistema, además de que realizar una descripción textual de los mismos.

2.5.1. Actores del Sistema.

Los actores de un sistema son agentes externos, roles que las personas o dispositivos juegan cuando interactúan con el software. En este caso quien hará uso del sistema será el estudiante o profesor, que como actor del sistema se llamará usuario.

Actor	Descripción
Usuario	Es quien interactúa con el sistema para ejecutar las funcionalidades del sistema. El usuario puede ser el estudiante o el profesor.

Tabla 2.2 Actor del Sistema.

2.5.2. Diagrama de Casos de Uso del Sistema.

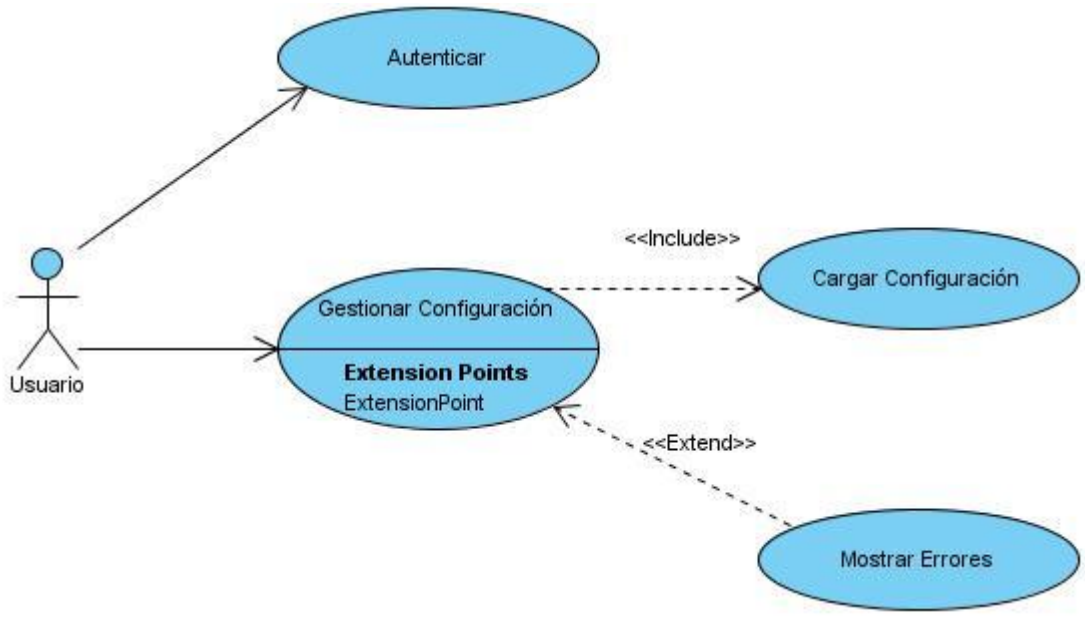

Figura 2.3 Diagrama de Caso de Uso.

2.5.3. Descripción de los Casos de Usos del Sistema.

Cada caso de uso tiene una descripción de las funcionalidades que ejecutará el sistema propuesto como respuesta a las acciones del usuario. Las tablas presentadas a continuación argumentan los flujos operacionales de cada caso de uso.

Caso de Uso:	Autenticarse.
Actores:	Usuario.
Resumen:	El caso de uso inicia cuando el usuario entra los datos para autenticarse que sería: usuario y número de pasaporte o número de cédula y el sistema procede a validar los datos.
Precondiciones:	Se visualiza una ventana de autenticación.
Referencias	R1
Prioridad	Crítica
Flujo Normal de Eventos	
Sección "Autenticarse"	
Acción del Actor	**Respuesta del Sistema**
1. El caso de uso inicia cuando el usuario entra los datos para autenticarse y acepta.	1.1. Valida que los datos entrados estén escritos correctamente. 1.2. Muestra la ventana de control de configuración terminando así el caso de uso.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
2. El usuario Acepta y termina el caso de uso.	1.2. Muestra un diálogo especificando cual es el error cometido al entrar los datos.
Poscondiciones	Se muestra la ventana de control de configuraciones.

Tabla 2.3 Descripción del CU Autenticar.

Caso de Uso:	Gestionar Configuración.	
Actores:	Usuario.	
Resumen:	El caso de uso inicia cuando el usuario comienza a configurar los servicios y comprueba o salva la configuración.	
Precondiciones:	Se debe haber cargado la configuración básica del servicio seleccionado.	
Referencias	R1,R2,R3	
Prioridad	Crítica	
Flujo Normal de Eventos		
Sección "Gestionar Configuración"		
Acción del Actor		**Respuesta del Sistema**
1 El caso de uso inicia cuando el usuario comienza a configurar los servicios y comprueba o salve el reporte de configuración.		1.1 El sistema ejecuta una de las acciones que se muestran a continuación. a) Si el actor desea comprobar la configuración. b) Si el actor desea guardar el reporte.
Poscondiciones	Se muestra la ventana de control de configuraciones.	

Tabla 2.4 Descripción del CU Gestionar Configuración.

Prioridad	Crítica	
Flujo Normal de Eventos		
Sección "Comprobar Configuración"		
Acción del Actor		**Respuesta del Sistema**
1 El usuario selecciona la opción comprobar. 2. El usuario acepta terminando así el caso de uso.		1.1 El sistema verifica que no se hayan cometido errores en la configuración y le muestra una ventana de diálogo donde le dice que salve la configuración.
Flujos Alternos		
Acción del Actor		**Respuesta del Sistema**
		1.1 El sistema muestra los errores cometidos en la configuración terminando así el caso de uso.
Poscondiciones	Se muestra la ventana de de la configuración seleccionada.	

Tabla 2.5 Descripción del CU Comprobar Configuración.

Prioridad	Crítica
Flujo Normal de Eventos	
Sección " Guardar Reporte "	
Acción del Actor	**Respuesta del Sistema**
1 Selecciona la opción Reporte. 3. Si el usuario desea salvar nombra el fichero de Reporte y selecciona la dirección donde desea salvarlo. 4. Acepta, terminando así el caso de uso.	1.1 El sistema verifica la configuración, genera el historial y muestra una ventana para salvar o cancelar el Reporte. 2.1 El sistema le muestra una ventana con la dirección del fichero salvado.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
2 Selecciona Cancelar. 3 Acepta y termina así el caso de uso.	2.1 El sistema muestra una ventana de diálogo que le dice que debe salvar el Reporte.
Poscondiciones	Se muestra la ventana de de la configuración seleccionada.

Tabla 2.6 Descripción del CU Guardar Reporte.

Caso de Uso:	Cargar Configuración.
Actores:	
Resumen:	El caso de uso inicia cuando el usuario selecciona configurar un servicio.
Precondiciones:	El usuario tiene que haberse autenticado.
Referencias	R2,R3
Prioridad	Crítica
Flujo Normal de Eventos	
Sección "Cargar Configuración"	
Acción del Actor	**Respuesta del Sistema**
1. El caso de uso inicia cuando el usuario selecciona configurara un servicio.	1.1 El sistema carga la configuración básica de los servicios a configurara terminando así el caso de uso.
Poscondiciones	Se muestra la ventana de control de configuraciones.

Tabla 2.7 Descripción del CU Cargar Configuración.

Caso de Uso:	Mostrar Errores.	
Actores:		
Resumen:	El caso de uso inicia cuando el usuario configura un servicio y comprueba la configuración realizada.	
Precondiciones:	El usuario tiene que configurar un servicio y comprobar la configuración	
Referencias	R2,R3	
Prioridad	Crítica	
Flujo Normal de Eventos		
Sección "Cargar Configuración"		
Acción del Actor		**Respuesta del Sistema**
1. El caso de uso inicia cuando el usuario selecciona configurara un servicio.		1.1 El sistema carga la configuración básica de los servicios a configurara terminando así el caso de uso.
Poscondiciones	Se muestra la ventana de control de configuraciones.	

Tabla 2.8 Descripción del CU Mostrar Errores.

Conclusiones Parciales.

En la actualidad existen varias tecnologías a utilizar para el desarrollo de una aplicación, es importante hacer un estudio profundo de las herramientas antes de comenzar a desarrollar una aplicación, siempre teniendo en cuenta los requerimientos del sistema. En este capítulo se resumió el funcionamiento de cada herramienta seleccionada a utilizar, así como algunas de las razones del por qué fueron utilizadas, además de las descripciones de los casos de uso del sistema en cuestión.

CAPÍTULO 3: ANÁLISIS Y DISEÑO.

En este capítulo se muestran los diagramas de clases de diseño y el análisis del sistema. También se detalla la relación estructural entre los objetos que interactúan, dígase diagramas de clases, diagramas de secuencia y del diseño. Además se muestran los diagramas de análisis y diseño de uno de los intérpretes realizados dado que todos se conforman de igual manera.

3.1. Diagrama de Clases de Análisis.

A continuación se expone el diagrama general de clases de análisis para comprender mejor los procesos que se llevan a cabo en el dominio del sistema. Se realizó un diagrama general y no por casos de usos para un mejor entendimiento.

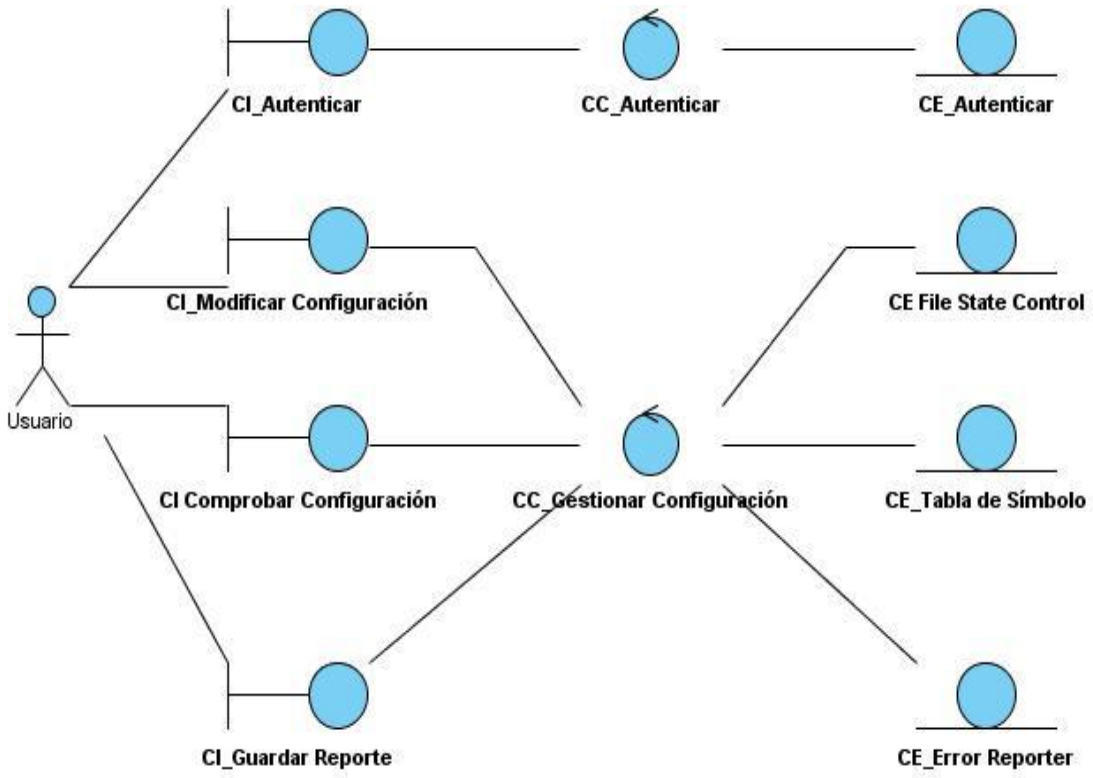

Figura 3.1 Diagrama General de Clases del Análisis.

3.2. Diagramas de Secuencia.

Para observar la interacción entre los objetos de la clase del análisis, se realizaron los diagramas de secuencia de los casos de uso mencionados en el capítulo anterior.

3.2.1. Diagrama de Secuencia del Caso de Uso Autenticar.

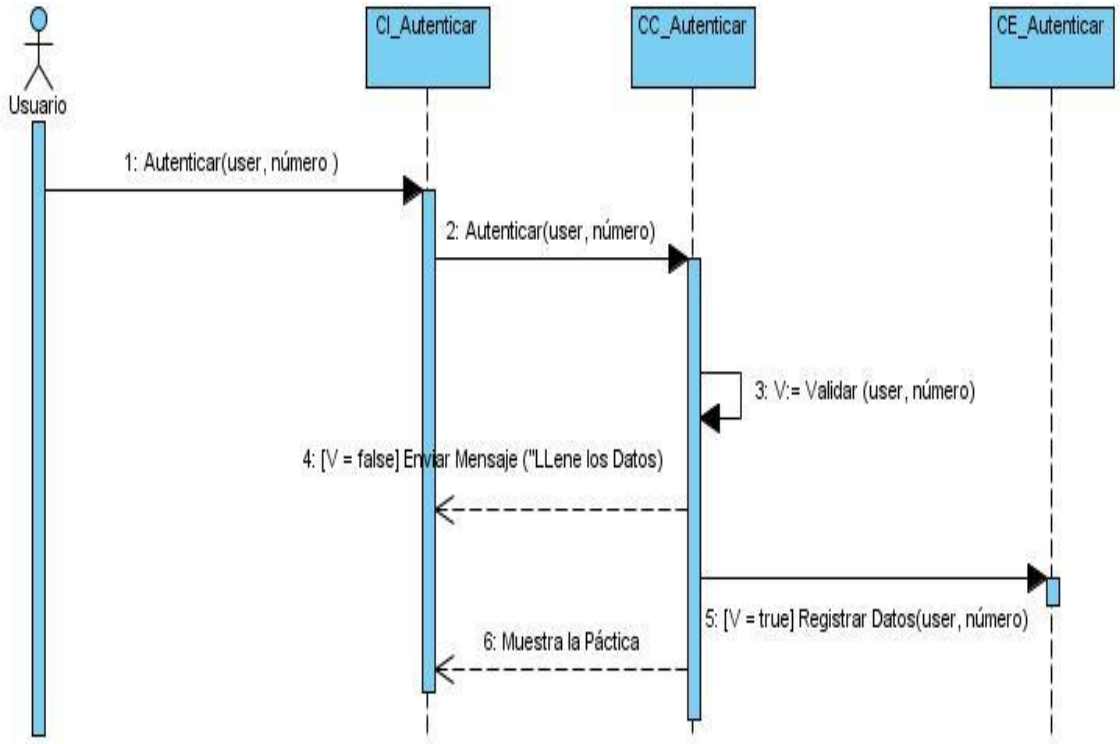

Figura 3.2 Diagrama de Secuencia del Caso de Uso Autenticar.

3.2.2. Diagrama de Secuencia del Caso de Uso Comprobar.

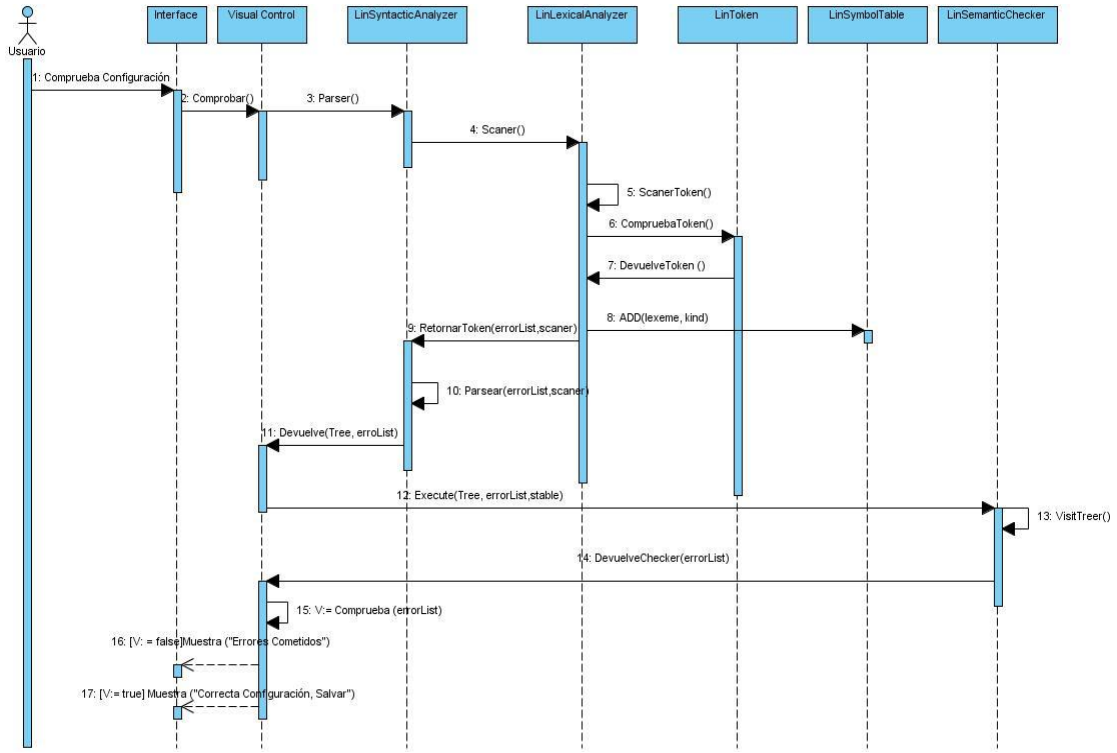

Figura 3.3 Diagrama de Secuencia del Caso de Uso Comprobar.

3.2.3. Diagrama de Secuencia del Caso de Uso Reporte.

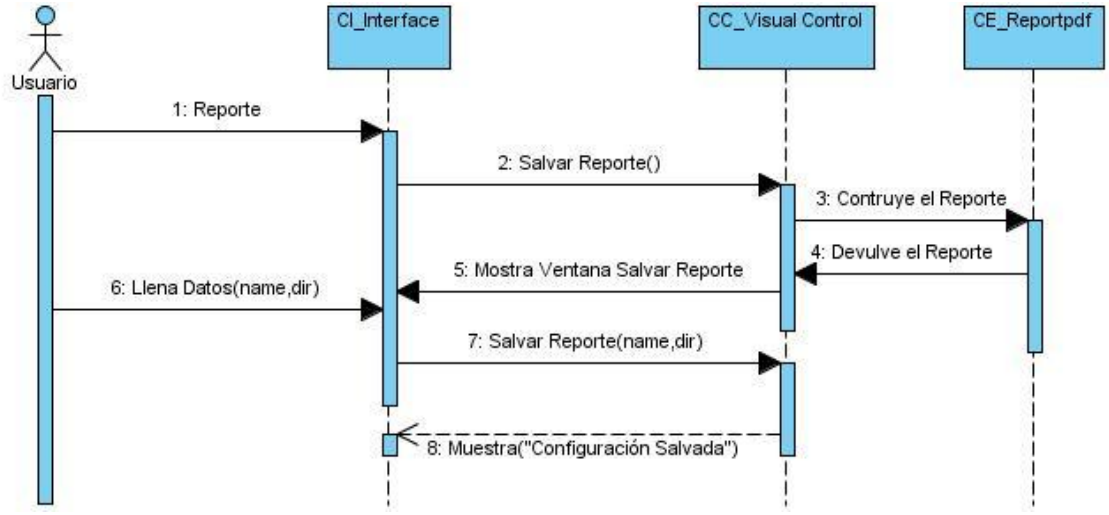

Figura 3.4 Diagrama de Secuencia del Caso de Uso Reporte.

3.3. Diagrama de Clases del Diseño del Sistema.

El diagrama de clases del diseño fue organizado por paquetes, para una mejor comprensión del mismo. Se realizaron tres paquetes más uno que es el de paquete Framework, el cual se relaciona con la biblioteca Qt específicamente por lo cual no se realizaron las clases de diseño para este paquete. Los otros paquetes desarrollados son: el paquete GUI el cual se muestra las clases que conforman la interfaz general del módulo y los paquetes Utility e Intérpretes, los cuales agrupan las clases bases del interprete y las clases que conforman las diferentes fases por las que transita el intérprete respectivamente.

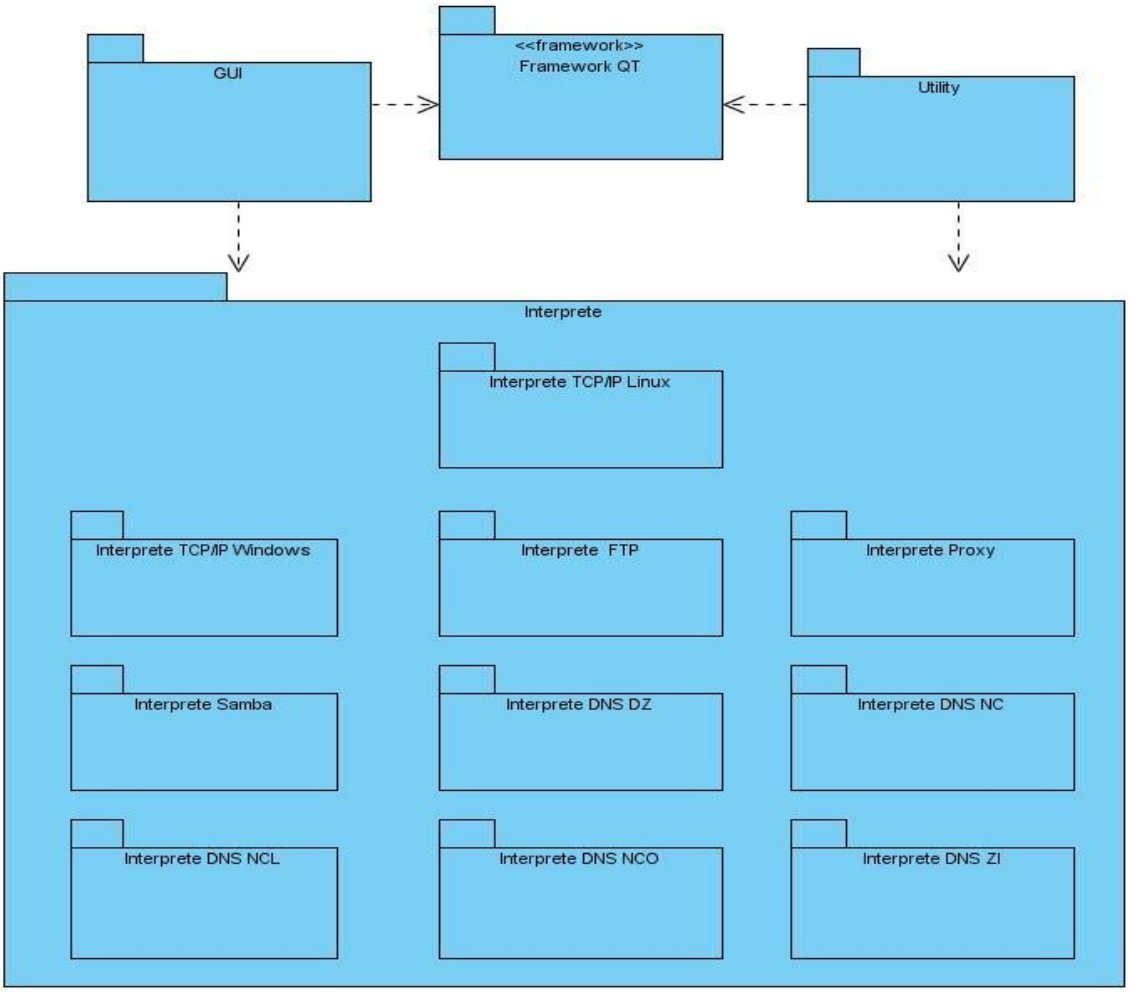

Figura 3.5 Diagrama General de Paquetes de Clase del Diseño.

3.3.1. Diagrama de Clases del Paquete GUI.

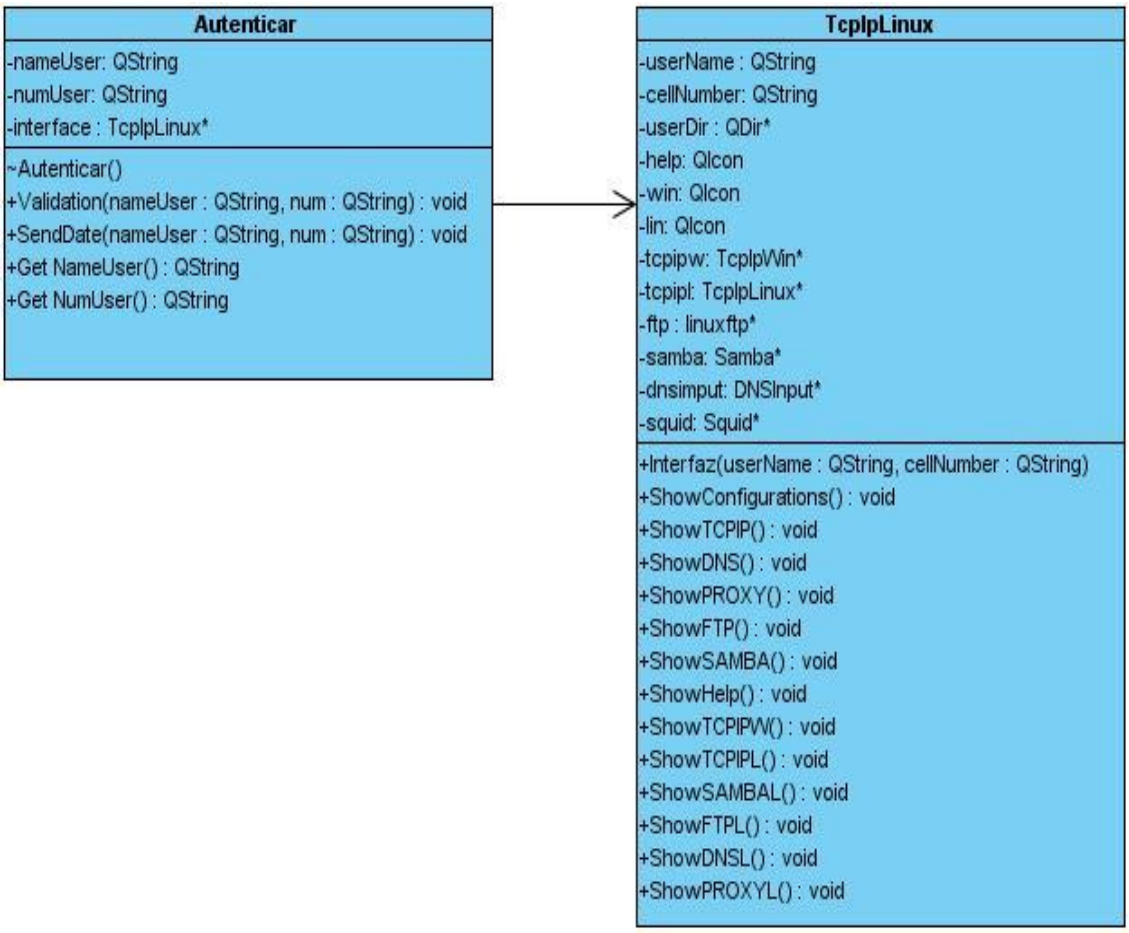

Figura 3.6 Diagrama de Clases de Diseño del Paquete GUI.

3.3.2. Diagrama de Clases del Paquete Utility.

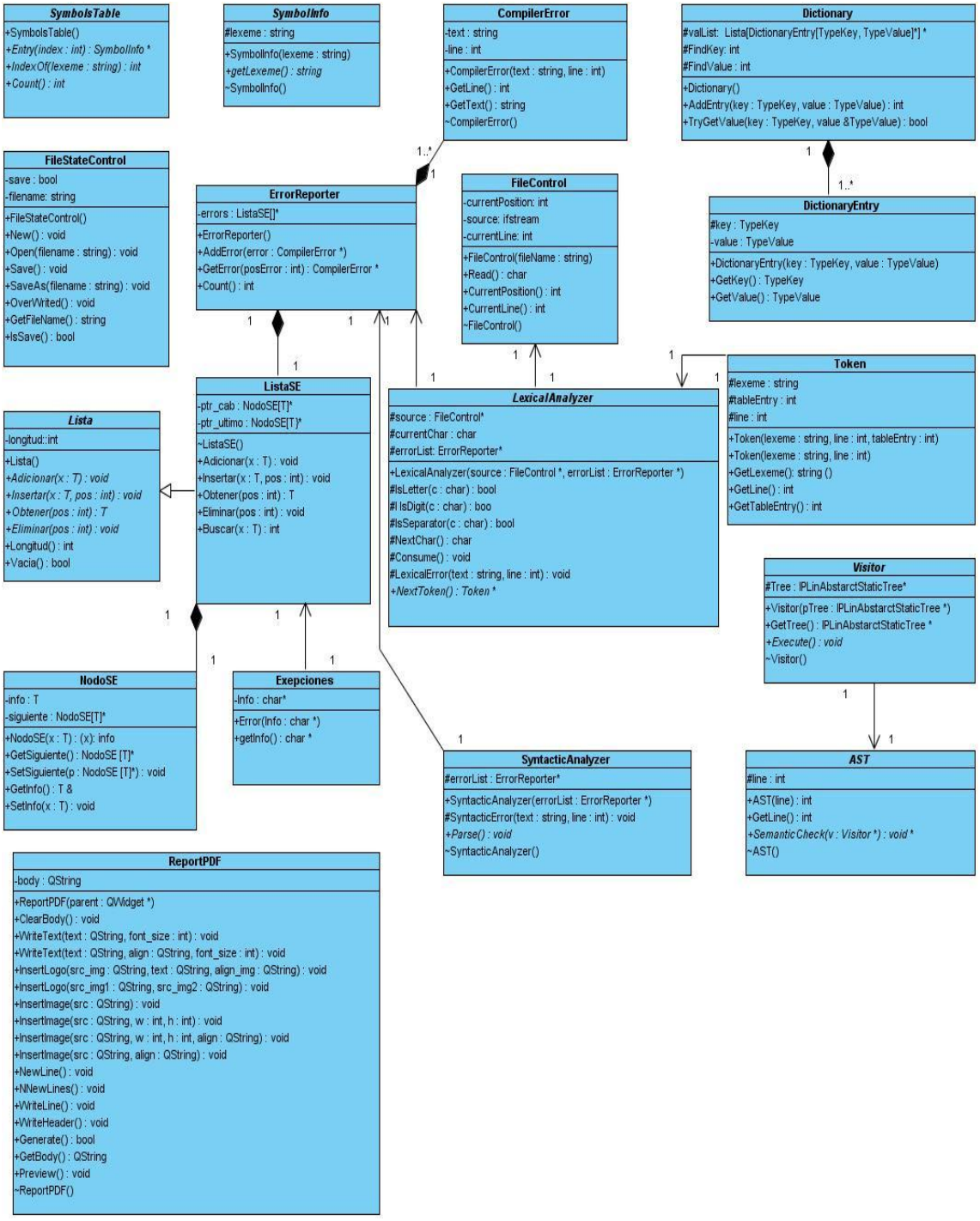

Figura 3.7 Diagrama de Clases de Diseño del Paquete Utility.

3.3.3. Diagrama de Clases del Paquete Intérprete.

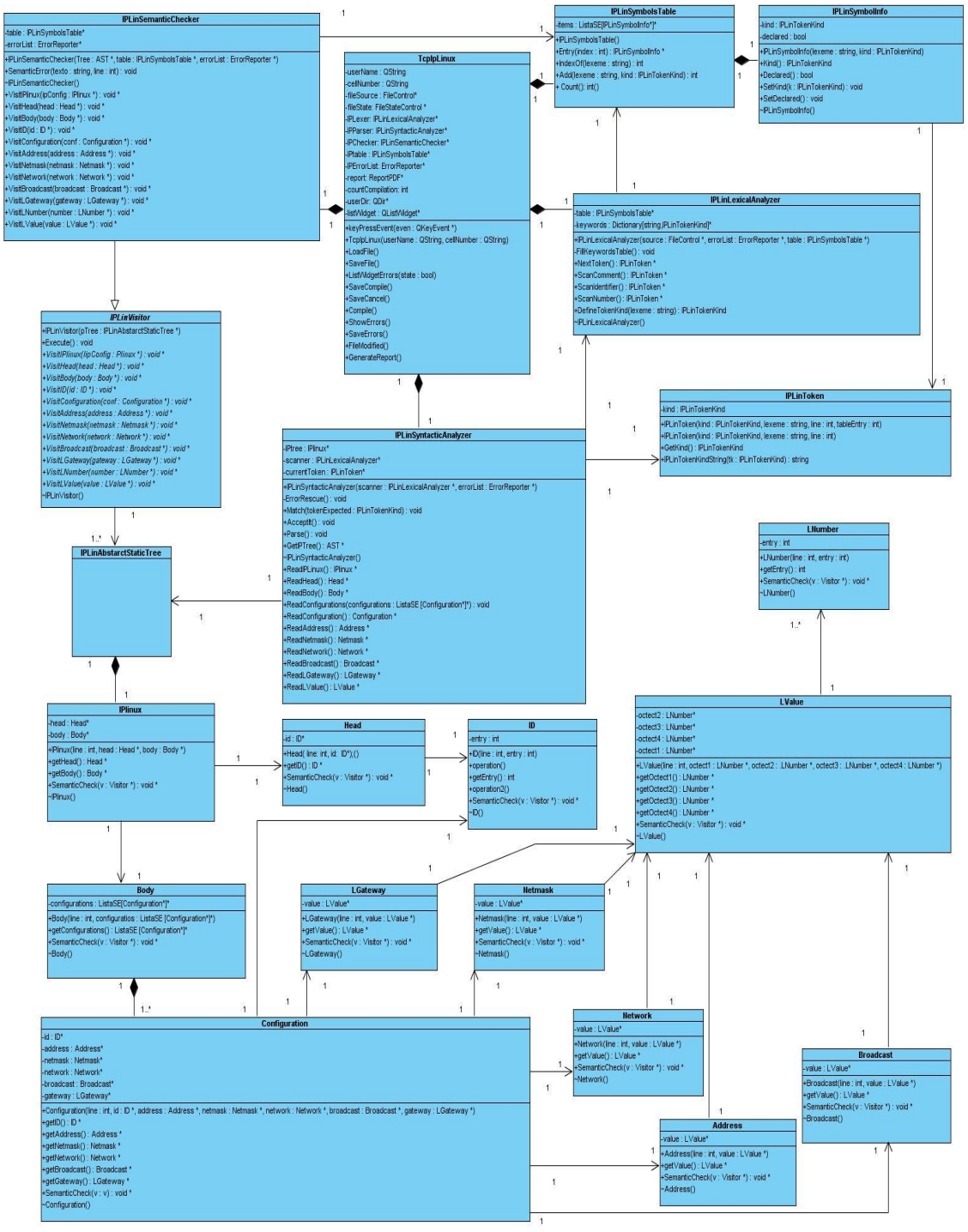

Figura 3.8 Diagrama de Clases de Diseño del Paquete Intérprete.

CAPÍTULO 4: IMPLEMENTACIÓN Y VALIDACIÓN.

En este capítulo se muestras los diagramas de componentes que forman el sistema además de los temas concernientes a la implementación y validación de la aplicación basada en todas las descripciones de los capítulos anteriores.

4.1. Diagrama de Componentes.

Las clases del análisis y el diseño se hacen físicas mediante componentes. A continuación se muestra como se agruparon en paquetes todos los componentes según sus clases.

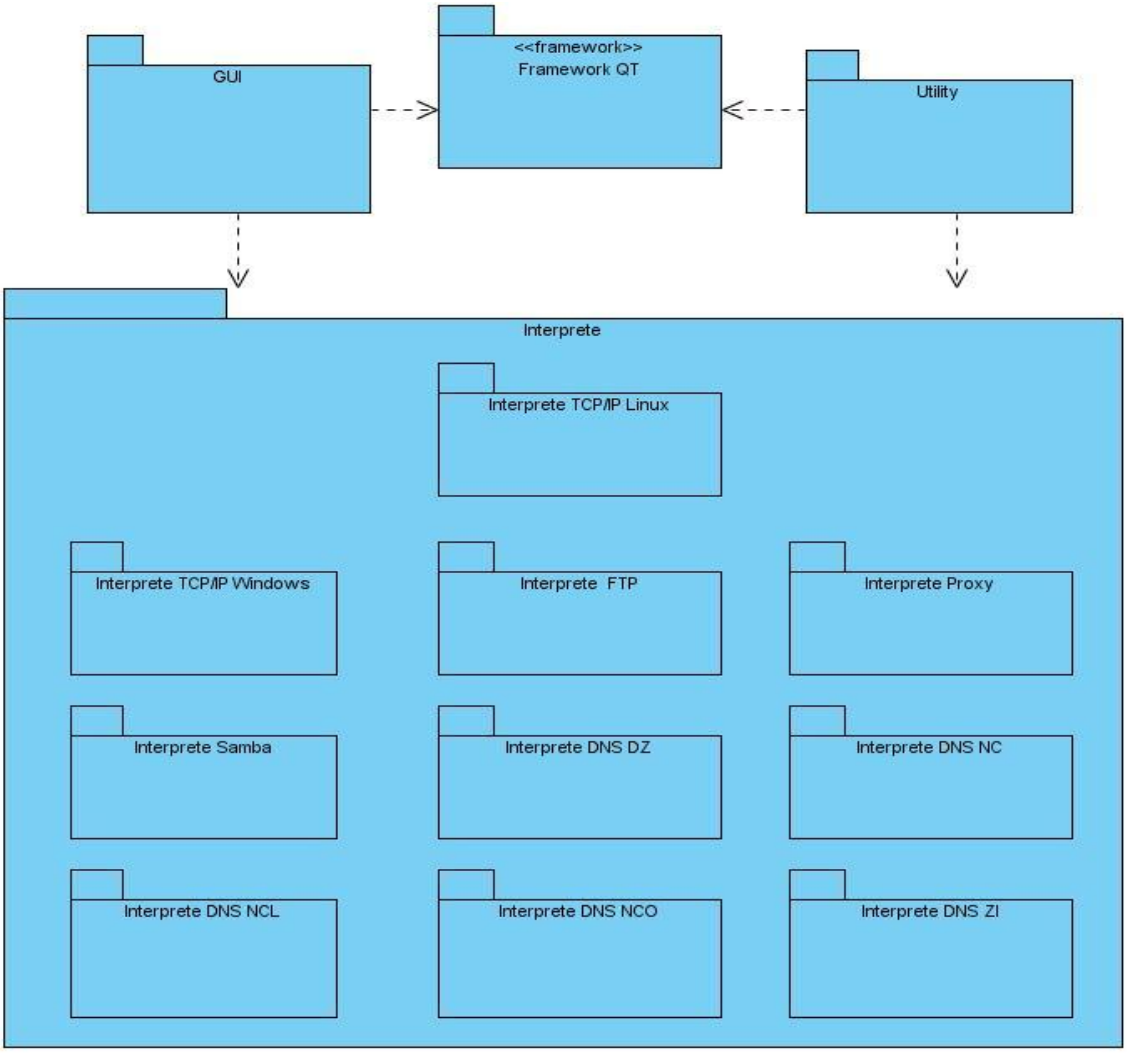

Figura 4.1 Diagrama del Paquete de Componentes.

4.2. Diagrama de Componentes del Paquete Interface.

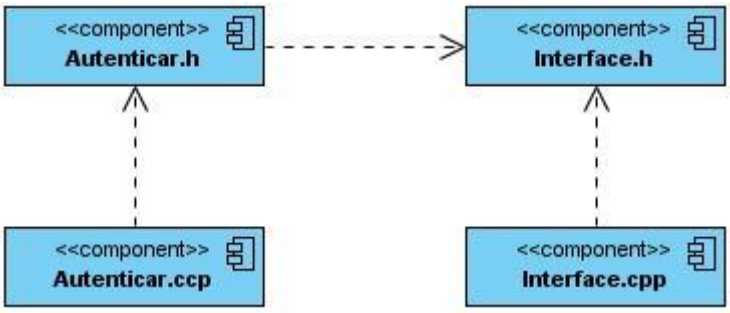

Figura 4.2 Diagrama de Componentes del Paquete Interface.

4.3. Diagrama de Componentes del Paquete Utility.

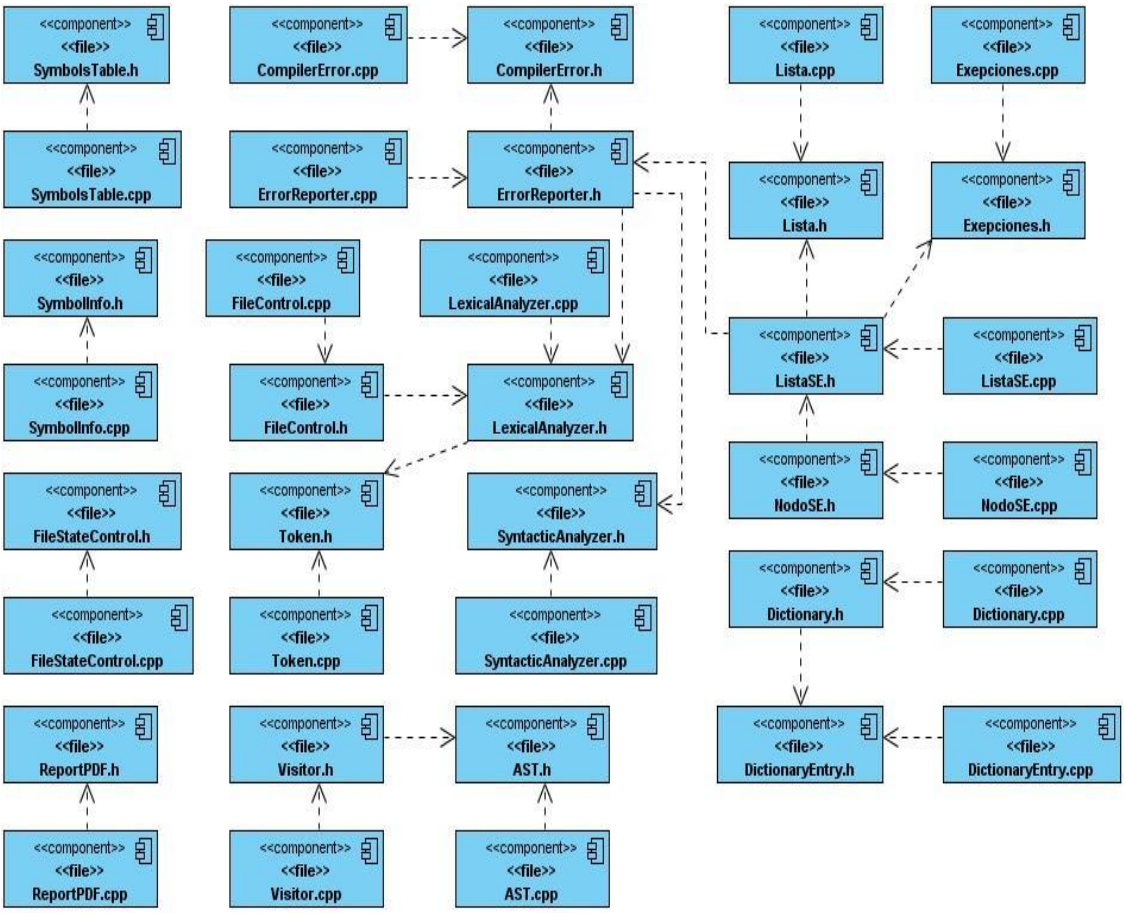

Figura 4.3 Diagrama de Componentes del Paquete Utility.

4.4. Diagrama de Componentes del Paquete Intérprete.

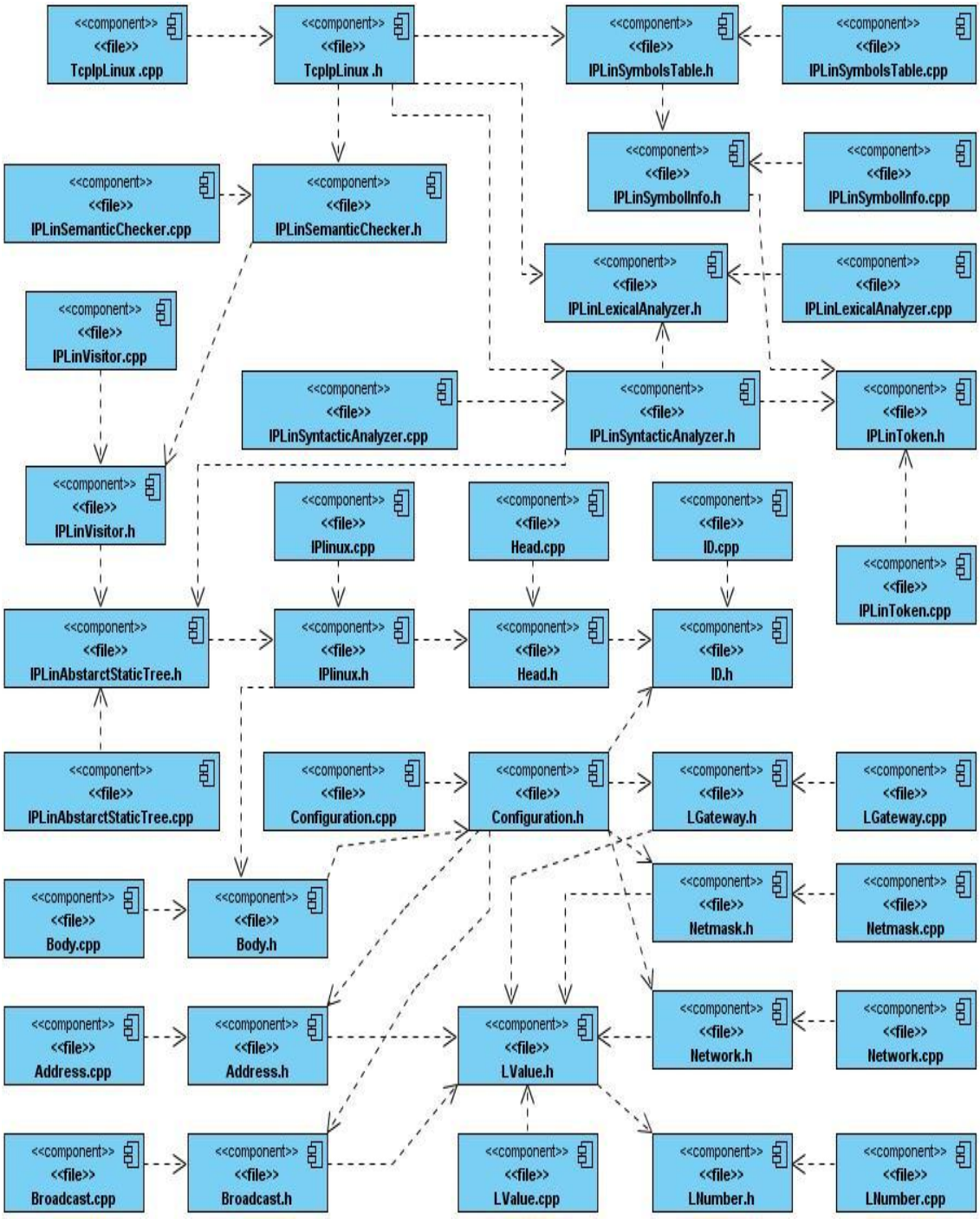

Figura 4.4 Diagrama de Componentes del Paquete Intérprete.

4.5. Validación.

Para la validación de esta herramienta se utiliza la prueba por funcionalidad de casos de uso. Estas se centran principalmente en la evaluación de la calidad del producto y se lleva acabo de acuerdo a la siguiente práctica:

- ✓ Validar que el software funciona de acuerdo a las especificaciones que se hicieron en el diseño.
- ✓ Validando que los requerimientos hayan sido implementados apropiadamente
- ✓ Validación del diseño.

En las pruebas realizadas se confecciona una descripción que responde a: Entrada, Resultado y Descripciones.

4.5.1 Validación del Caso de Uso Autenticar.

Entrada

El usuario debe entrar el nombre, el número de pasaporte o número de cédula.

Figura 4.5 Interfaz de Autenticación.

Resultado

Si la autenticación está correcta el usuario ingresa a la práctica, si no se le muestran las ventanas emergentes del error cometido.

✓ Error de campos vacíos.

Figura 4.6 Interfaz de Alerta de Campo Vacío.

✓ Error en los datos de autenticación.

Figura 4.7 Interfaz de Alerta de Datos Incorrectos.

Descripción

Se verificó la correcta autenticación del usuario mediante las validaciones requeridas.

4.5.2. Validación del Caso de Uso Comprobar Configuración.

Entrada

Se comprueban varias configuraciones bajo diferentes condiciones y se chequea que se realicen todas las tareas el proceso de compilación.

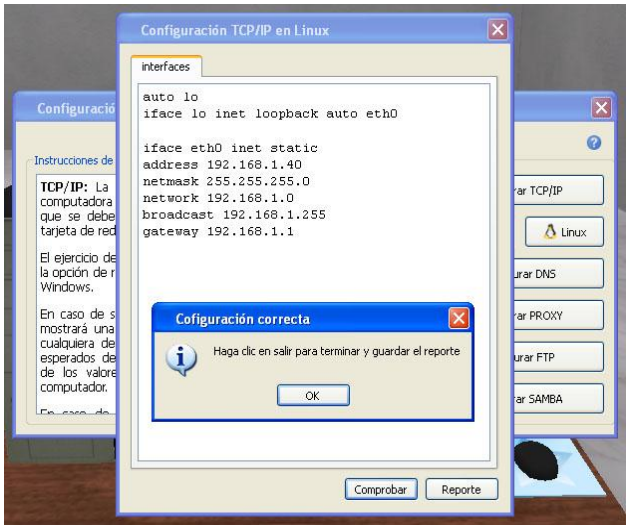

Figura 4.8 Interfaz de Configuración.

Resultado

Si la configuración esta correcta todas las tareas comprendidas en el proceso de compilación se desarrollan y se muestra la ventana de guardar el reporte, sino se le muestran los errores de la configuración realizada.

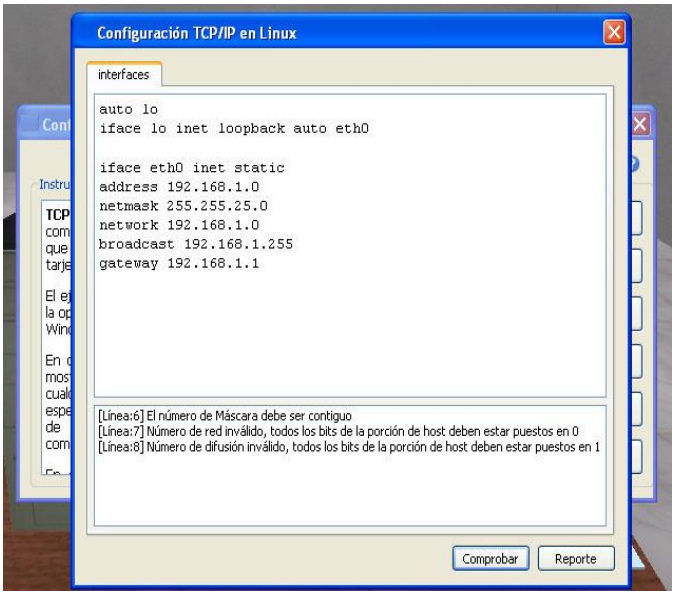

Figura 4.9 Interfaz de Errores de Configuración.

Descripción

Se verifica el correcto funcionamiento de los fases que conforman el proceso de compilación entre ellas llenar lista de errores.

4.5.3. Validación del Caso de Uso Reporte de la Configuración.

Entrada

Guardar las configuraciones realizadas por los usuarios en la práctica.

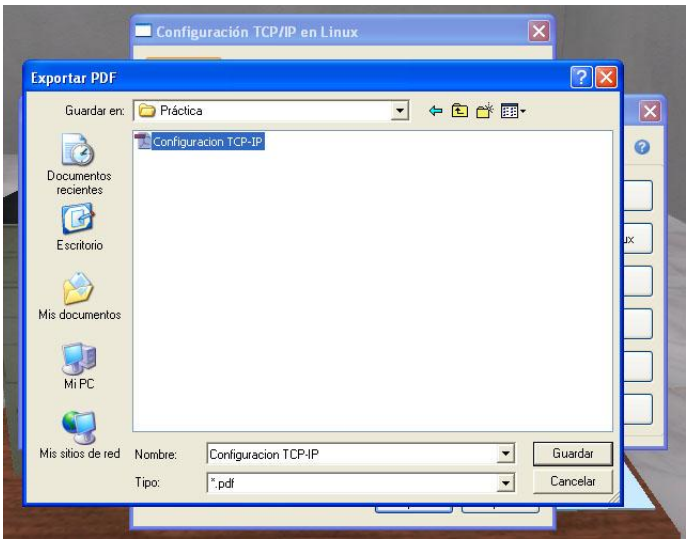

Figura 4.10 Interfaz de Guardar el Reporte.

Resultado

El fichero generado contiene cada operación de las configuraciones realizadas en la práctica.


Gobierno Bolivariano de Venezuela | Ministerio del Poder Popular para la Educación Universitaria

BICENTENARIO

5/17/2011 3:18:13 PM

Nombre y Apellidos: Ronaldo Silva

Identificador: 123652415

Código de barra:

Configuración y Administración de una Red de Área Local(LAN)

Configuración TCP/IP(Linux)

Historial de la configuración realizada por el usuario:

Configuración número 1:

auto lo

iface lo inet loopback auto eth0

iface eth0 inet static

address 192.168.1.40

netmask 255.255.255.0

network 192.168.1.0

broadcast 192.168.1.255

gateway 192.168.1.1

Resultado final

Configuración correcta

Figura 4.11 PDF Generado.

Descripción

Se verificar que el fichero generado una vez guardado contenga todos los datos establecidos en la plantilla.

Como parte de la validación de la herramienta hay que mencionar que por parte de los clientes hubo una buena aceptación de la herramienta y quedaron satisfechos con la misma.

Conclusiones Parciales.

En este capítulo se brindó información de las clases que componen la aplicación, también se describieron a las funcionalidades del módulo y las distintas pruebas de casos de uso realizadas al sistema. Mediante las pruebas realizadas podemos afirmar que el módulo desarrollado cumple con el objetivo propuesto de lograr todas las validaciones correctamente en los diferentes servicios.

Conclusiones.

Para cumplir con los objetivos propuestos en la investigación, teniendo en cuenta las características de modulo que se quería lograr y las herramientas para su desarrollo, fue necesario hacer un estudio de las tecnologías actuales que posibilitarían la implementación del mismo. Además se realizó el proceso de Ingeniería del Software utilizando el Proceso Unificado del Software, con el que se logró plasmar la captura de los requisitos funcionales y no funcionales, la identificación de casos de uso del sistema en conjunto con su descripción en formato expandido, los diagramas que se generan del flujo de trabajo Análisis y Diseño y los diagramas de componentes propios de la solución.

Concluida la investigación podemos asegurar que el módulo desarrollado permite realizar las validaciones de la configuración que requiere el Laboratorio Virtual de Administración y Configuración de servicios de una red LAN. El mismo cumple con las especificaciones requeridas, además de que su integración al LV es sencilla así como su utilización por parte de los usuarios que van a realizar la práctica. Podemos afirmar que en la presente investigación se alcanzó el objetivo propuesto dándole cumplimiento a todas las tareas planteadas en la investigación.

Recomendaciones.

En el presente trabajo se alcanzaron los objetivos planteados pero como toda investigación se puede seguir perfeccionando para lo cual se recomienda:

1. Adicionar al módulo algunas funcionalidades que permitan al usuario personalizar las interfaces que se le muestran.
2. Profundizar en las temáticas de compiladores en el tema de las redes informáticas para implementar un compilador que permita el trabajo con distintas gramáticas para así no implementar uno diferente por cada servicio.
3. Se sugiere tomar en cuenta la aplicación de los procesos de compilación en otras temáticas de los Laboratorios Virtuales donde se requiera hacer validaciones.

Bibliografía

1. **Aho, Alfred V. y Ullman, Jefferey**. Compiladores Principios Técnicas y Herramientas. 2000.
2. **Pichastor, R. Peris, Nieto, S. Agut, y Navarro, Gimeno M.** Entornos virtuales de aprendizaje: El papel del valor del entorno. Castellon, España. 2006.
3. **Roa, Pablo F., Loyarte, Horacio.** Implementación de un laboratorio virtual de redes por intermedio de software de simulación. Facultad de Ingenieria y Ciencias Hídricas. Universidad Nacional del Litoral. Argentina. 2007.
4. **Rodríguez, Lilier Pérez, Puerto, Janet Bárbara Viera y Grao, Yordan Danilo.** Laboratorio Virtual de Redes. Universidad de las Ciencias Informáticas, Facultad 4. La Habana. 2007.
5. **Salas, Calos Vázques.** Laboratorios Virtuales. 20 Julio del 2009. ISSN 1988 - 6047. [En línea] *http://*www.csi-csif.es.
6. **Nodarse, Dr.C. Francisco A. Fernández, González, Dr.C. Edwin Pedrero Lima, MSc. Francisco Fernández.** Laboratorios virtuales en la Universidad virtual del CITMA. Ministerio de Ciencia, Tecnología y Medio Ambiente. La Habana, Cuba. [En línea] *http://www.cursosenlínea.cu.*
7. **L. Rosado, Herreros, J. R.** Nuevas aportaciones didácticas de los laboratorios virtuales y remotos en la enseñanza de la Física. Universidad Carlos III de Madrid, España. 2005. [En línea] *http://*www.uv.es/eees/archivo/286.pdf.
8. **Rivas, Federico Reina Toranzo, Ruiz, Juan Antonio.** Redes de área local. 2006. [En línea] *http://*www.forpas.us.es/aula/hardware/dia4_**redes**.pdf.
9. **Salinas, Alejandro Castán.** Introducción a TCP/IP. [En línea] *http://www.*ownz.despai.es/Documentos/Protocolos/**Introduccion**%20a%20**tcpip**.pdf.
10. **Bísaro, Mauricio Eduardo Danizio.** Laboratorio de Redes Diseño y Configuración de Redes de Computadoras. Direccionamiento IP.
11. **Zuleta, José Vicente Núñez.** DNS (Domain Name System). Venezuela. 2007. Email: *jose@ing.ula.ve.*

12. **Alvarez, Miguel Angel, Juszkiewicz, Leo, Alvarez, Sara, Lazaro, Juliana Monteiro, Garay, William Wong.** Tutorial de FTP. Venezuela. [En línea] *http://www.desarrolloweb.com/manuales/72/.*

13. **Pérez, Javier Ayllón.** Seguridad y Alta Disponibilidad Instalación y Configuración de servidores proxy. Escuela Superior de Informática Universidad de Castilla-La Mancha,España. [En línea] *http://www.*arco.esi.uclm.es/~david.villa/**seguridad**/**servidores_proxy**.2x4.pdf

14. **Faría, Daniela A. Torres.** IPTABLES y Squid. Laboratorio Docente de Computación. Venezuela. Junio de 2010. Email: *daniela@ldc.usb.ve.*

15. **Kiracofe, Daniel.** Proxy transparente con Squid. Abril 2000.

16. **Eckstein, Robert, Collier−Brown, David, Kelly, Peter.** Usando Samba. Noviembre 2002, Vol. Primera Edición.

17. **Ángel, Miguel.** Del Funcionamiento y comportamiento de los Compiladores.

18. **Robaina, Abdiel Cruz, García, Yisnier Guerra.** Herramienta de Compilación para Métodos Numéricos. La Habana. Julio 2008.

19. **Conferencia 1. Programación IV.** Introducción a las Técnicas de Compilación. Universidad de las Ciencias Informáticas, La Habana. 2011.

20. **Conferencia 4. Programación IV.** Fase de Análisis Léxico. Universidad de las Ciencias Informáticas, La Habana. 2011.

21. **Conferencia 5. Programación IV.** Fase de Análisis Sintáctico. Universidad de las Ciencias Informáticas, La Habana. 2011.

22. **Conferencia 8. Programación IV.** Fase de Análisis Semántico. Universidad de las Ciencias Informáticas, La Habana .2011.

23. **Sanchez, María A Mendoza.** Metodologías de Desarrollo de Software. 2004.

24. **Zarzuela, Jorge Ferrer.** Metodologías Ágiles. [En línea] *http://libresoft.dat.escet.urjc.es/html/downloads/ferrer-20030312.pdf.*

25. **Grady Booch, Ivar Jacobson.** El Proceso Unificado del Desarrollo del Software.

Anexos

A continuación se muestran imágenes del módulo desarrollado ya integrado en el Laboratorio Virtual de Administración y Configuraciones.

Figura A.1 Interfaz Principal.

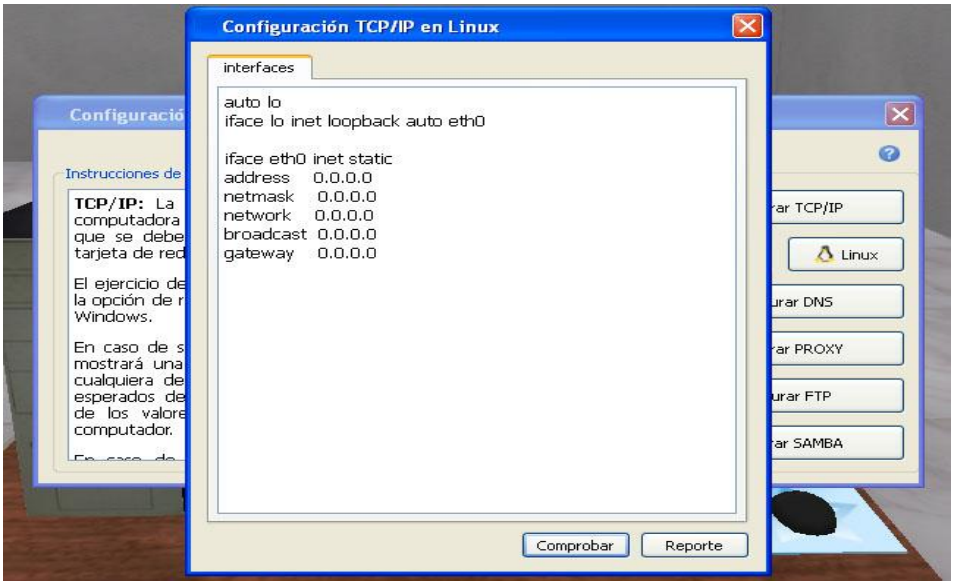

Figura A.2 Interfaz de Configuración.

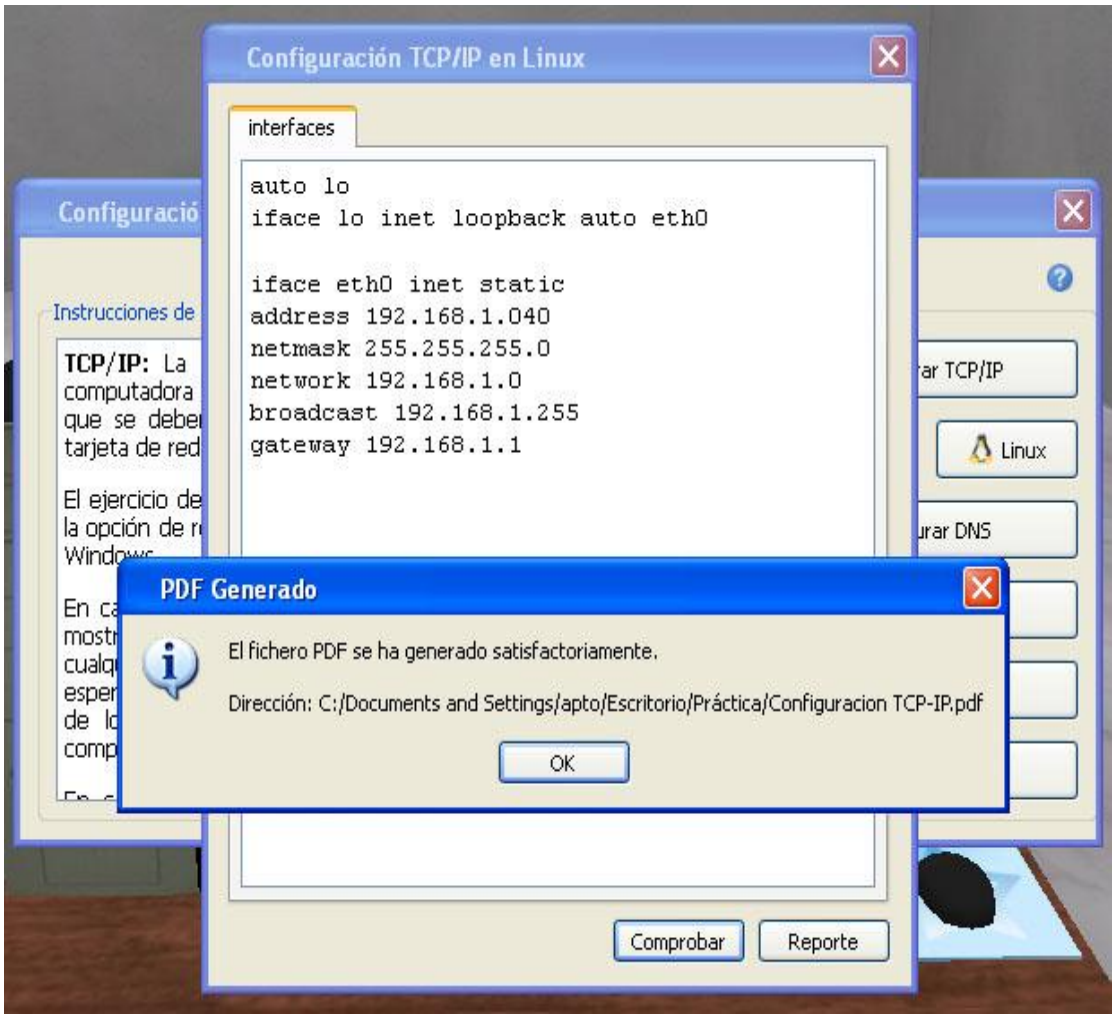

Figura A.3 Configuración Salvada.

Glosario de Términos.

DNS: Sistema de Nombres de Dominio. Permite el mapeo de las direcciones IP a nombres de dominio y viceversa.

FORTRAN: Lenguaje de programación informática de alto nivel, ha sido ampliamente adoptado por la comunidad científica para escribir aplicaciones de cálculos intensivos.

FTP: Protocolo de Trasferencia de Ficheros. Protocolo que permite la trasferencia de archivos en la red entre los ordenadores.

GNU/GPL: Licencia Pública General de GNU o más conocida por su nombre en inglés GNU General Public License o simplemente su acrónimo del inglés GNU GPL, es una licencia creada por la Free Software Foundation a mediados de los 80, y está orientada principalmente a proteger la libre distribución, modificación y uso de software.

IP: Protocolo de Internet. Permite a los usuarios obtener un identificador para su ordenador, el cual identifica al mismo en su red.

Laboratorios Virtuales: Herramientas virtuales que simulan al laboratorio tradicional que conocemos en la realidad. Estos pueden ser de distintos tipos y para diferentes asignaturas o materias.

Módulo: Es una parte de un programa, o más general un componente de un sistema que tiene una interfaz bien definida para interactuar con otros módulos.

Multiplataforma: Se refiere a los programas, sistemas operativos, lenguajes de programación, u otra clase de software, que puedan funcionar en diversas plataformas.

PROXY: Es un servicio que permite el filtrado de la conexión desde una intranet hacia una Internet.

SAMBA: Es una implementación libre del protocolo de archivos compartidos que permite la convivencia en la red de ordenadores bajo diferentes sistemas operativos para que los mismos puedan compartir información.

Servicios de Redes: Permite que se cumpla la finalidad de una red de sistemas informáticos, mediante estos los usuarios pueden realizar conexiones, compartir información, navegar en la red entre otros.

TIC: Tecnologías de la Información y de la Comunicación conforman el conjunto de recursos necesarios para manipular la información, particularmente los ordenadores, programas informáticos y redes necesarias para convertirla, almacenarla, administrarla, transmitirla y encontrarla.