ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN
UNIVERSIDAD POLITÉCNICA DE CARTAGENA

Proyecto Fin de Carrera

Diseño e implementación de un emulador de router con fines didácticos.

AUTOR: Justo Alcón Cazorla
DIRECTOR: Pablo Pavón Mariño

Octubre / 2.006

Autor	Justo Alcón Cazorla
E-mail del Autor	justo.alcon@gmail.com
Director(es)	Pablo Pavón Mariño
E-mail del Director	pablo.pavon@upct.es
Codirector(es)	
Título del PFC	Diseño e implementación de un emulador de router con fines didácticos
Descriptores	Java, Routers CBRA de Teldat, software de configuración

Resumen

El objetivo de este Proyecto Fin de Carrera es el desarrollo de un simulador del router CBRA de Teldat para fines didácticos.

La finalidad de este simulador será la de proporcionar una herramienta que, implementando tan solo la funcionalidad básica del router, permita practicar el proceso de configuración de un router CBRA real. El simulador proporcionará facilidades suficientes para el aprendizaje en la configuración del citado router comportándose como lo haría el router en la realidad, en una parte relevante de sus funciones.

Titulación	Ingeniero Técnico de Telecomunicación, Esp. Telemática
Intensificación	
Departamento	Tecnologías de la Información y las Comunicaciones
Fecha de Presentación	Octubre – 2.006

Índice de Contenidos

Capítulo 1
Introducción

1.1 Introducción.

El presente documento se encargará de describir una herramienta software desarrollada como Proyecto Fin de Carrera. El software en cuestión será un *simulador* del router CBRA de Teldat, el cual es usado con fines didácticos en diferentes universidades.

La finalidad de este *simulador* será la de proporcionar una herramienta que, implementando tan solo la funcionalidad básica del router, permita practicar el proceso de configuración de un router CBRA real. Atendiendo a esto, el *simulador* deberá proporcionar facilidades suficientes para el aprendizaje en la configuración del citado router comportándose como lo haría el router en la realidad, en una parte relevante de sus funciones.

Antes de continuar, es imprescindible explicar más detenidamente lo que es un router. No entraremos en detalle, pero sí dejaremos algunos conocimientos básicos claros para que el lector pueda entender más tarde el funcionamiento del router que utilizaremos para el *simulador* y entender la forma de configurarlo.

1.1.1 El router, definiciones y funcionamiento básico.

El **router (enrutador o encaminador)** es un dispositivo que aúna hardware y software para la interconexión de redes de ordenadores/computadoras que opera en la capa tres (nivel de red) del modelo OSI. Este dispositivo interconecta segmentos de red o redes enteras, dejando pasar paquetes de datos de una red a otra tomando como base la información de la capa de red.

El router toma decisiones lógicas con respecto a la mejor ruta para el envío de datos a través de una red interconectada y más tarde dirige los paquetes hacia el segmento y el puerto de salida adecuados. Sus decisiones se basan en diversos parámetros, una de las más importantes es decidir la dirección de la red hacia la que va destinado el paquete (En el caso del protocolo IP, ésta sería la dirección IP). Otras decisiones son la carga de tráfico de red en los distintos interfaces de red del router y establecer la velocidad de cada uno de ellos, dependiendo del protocolo que se utilice.

En el protocolo IP versión 4 basado en clase puro, los encaminadores toman decisiones basándose en la porción de red de las direcciones IP de los datagramas, por lo que el número de entradas necesario en las tablas de encaminamiento será igual al número de redes físicas accesibles. El direccionamiento no basado en clase, con la característica de agregación de rutas, elimina esta necesidad. En cualquier caso, para evitar que el administrador de la red tenga que estar cambiando las tablas manualmente cada vez que cambien las redes conectadas a un router, existen una serie de protocolos que rellenan automáticamente y periódicamente las tablas de encaminamiento.

Los protocolos de enrutamiento son aquellos protocolos que utilizan los routers o encaminadores para comunicarse entre sí y compartir información que les permita

tomar la decisión de cual es la ruta más adecuada en cada momento para enviar un paquete. Los protocolos mas usados son RIP (v1 y v2), OSPF (v1, v2 y v3), y BGP (v4), que se encargan de gestionar las rutas de una forma dinámica. Aunque no es estrictamente necesario que un router haga uso de estos protocolos, pudiéndosele indicar de forma estática las rutas (caminos a seguir) para las distintas subredes que están conectadas al dispositivo.

Comúnmente los routers se implementan también como puertas de acceso a Internet (por ejemplo un router ADSL), usándose normalmente en casas y oficinas pequeñas. Es correcto utilizar el término router en este caso, ya que estos dispositivos unen dos redes (una red de área local con Internet).

1.1.1.1 Definiciones básicas dentro de una red.

A continuación se definirán algunos conceptos básicos dentro de las redes:

- Red física: se trata de una estructura física que interconecta dispositivos IP.
- Direcciones físicas: En las redes no punto a punto es necesario identificar cada tarjeta con una dirección física (MAC). Se necesitará la traducción de una dirección IP a una dirección física mediante el protocolo ARP en las redes no punto a punto.
- Red IP: Conjunto de dispositivos IP interconectados por redes físicas, tal que se sigue un esquema de direccionamiento coherente
- Interfaz de red: Entidad abstracta que interconecta un dispositivo IP con una red física. Está identificada con una dirección IP.
- Un dispositivo IP, para cada interfaz configurada debe conocer tanto el MTU (Maximum Transfer Unit) de la red física para tomar las decisiones de fragmentación, como si la red física es punto a punto o no lo es. En las redes físicas punto a punto, los datagramas enviados por ese interfaz sólo pueden llegar a una interfaz concreta, siempre la misma, por lo que no es necesario el protocolo ARP. En las redes físicas que no son punto a punto existirá la necesidad de una dirección física para cada interfaz y será necesario utilizar el protocolo ARP para la obtener la dirección física a partir de la dirección IP.
- Tablas de encaminamiento: Existe en todos los routers y PCs, y contiene información con la ruta para alcanzar otros dispositivos. Los contenidos de las tablas de encaminamiento pueden ser fijos y variar automáticamente para adaptarse a los cambios de la red. Los campos necesarios para rellenar las tablas de encaminamiento son: la dirección IP destino, su máscara, el interfaz de salida y el gateway (GW). El gateway solo se usará en caso de que la red de salida sea del tipo no punto a punto y no esté directamente conectada al router.

Todos estos campos serán importantes, ya que serán parámetros a configurar en el router.

1.1.1.2 Tipos de routers.

Hay varios tipos de routers, a destacar:

- Si usamos un PC con Windows 98 o superior, Linux, FreeBSD o Solaris, con más de una tarjeta de red, para compartir una conexión a Internet, ese PC estará haciendo una funcionalidad de router básico. Tan solo se encargará de

ver si los paquetes de información van destinados al exterior o a otro PC del grupo.

- Los routers algo más sofisticados, y de hecho los más utilizados, hacen algo más, entre otras cosas protegen nuestra red del tráfico exterior, y son capaces de manejar bastante más tráfico. Es por ello que son la opción más típica en pequeñas redes, e incluso, en usuarios domésticos.
- Los routers más potentes, que están repartidos por todo Internet para gestionar el tráfico, manejan un volumen de millones de paquetes de datos por segundo y optimizan al máximo los caminos entre origen y destino.

En Internet, como hemos mencionado, hay miles de routers que trabajan, junto con el nuestro, para buscar el camino más rápido de un punto a otro. Si tenemos un router en nuestra conexión a Internet, este buscará la ruta óptima para llegar a un destinatario, y ese router óptimo, buscará a su vez el siguiente óptimo para llegar al destinatario. Digamos que es un gran trabajo en equipo.

Tanto los routers medianos como los más potentes (de clase alta) permiten configurar qué información deseamos que pueda entrar o salir de nuestro PC o red. En caso de que deseemos ampliar las posibilidades de control deberemos añadir un dispositivo llamado Firewall (cortafuegos).

El router CBRA de Teldat será un router de categoría pequeña, típicamente utilizado en pequeñas empresas u oficinas.

1.2 El router CBRA de Teldat. Características.

El router de acceso CBRA permite conectar oficinas o pequeñas empresas dotadas de redes LAN Ethernet a Internet, a redes IP públicas, a la Intranet o Red Corporativa propia de la compañía, así como la creación de Redes Privadas Virtuales, mediante líneas de acceso RDSI o Red Telefónica Básica.

Características más importantes:

- Permite el acceso simultáneo de todos los puestos de una red LAN a Internet y/ó redes IP públicas.
- Permite interconectar redes LAN remotas a través de Internet usando túneles IP (Redes Privadas Virtuales).
- Privacidad en las comunicaciones mediante el cifrado de los datos.
- Mecanismos de seguridad: controles de acceso, autenticación PAP/CHAP, funcionalidades de Firewall Filtering, passwords, etc.
- Todas las anteriores funciones utilizando una sola dirección IP para toda la red LAN.
- Flexibilidad en acceso WAN, vía línea telefónica y módem, o mediante línea RDSI.
- Conexiones RDSI conmutadas o permanentes (Servicio Novacom Multiplan).
- Conectividad garantizada mediante mecanismos de backup.
- Ancho de banda agregado de hasta 128 Kbps utilizando MLPPP por los dos canales B RDSI.
- Acceso a diferentes destinos: hasta 3 conexiones simultáneas.
- Configuración sencilla a través de un configurador en Windows 3.x, Windows 95 o Windows NT.
- Permite aprovechar los costes reducidos del acceso a redes IP públicas.

Para permitir a cualquier empresa aprovechar fácilmente la ventaja estratégica que supone la utilización de las nuevas tecnologías de la información, y en concreto las relacionadas con conexión a Internet y/ó redes IP públicas, Teldat presenta la familia de routers de acceso CBRA. El router CBRA consta de los siguientes interfaces:

- Ethernet 10 Base-T.
- Acceso básico RDSI (2 canales B).
- Línea serie V.24 para conexión a módem externo.
- Consola para configuración y actualización de software.

Ilustración 1: Imagen Router CBRA de Teldat

1.2.1 Configuraciones y características destacables.

En los siguientes apartados se describen algunas de las configuraciones de conexión y algunas características destacables que ofrece el router CBRA. Esto proporcionará una idea de algunas de las posibilidades más utilizadas que proporciona el router.

1.2.1.1 Acceso de una red LAN a Internet, Redes IP públicas o Intranets.

El router CBRA permite el acceso simultáneo de todos los puestos de una red de área local a Internet, mediante RDSI (Red Digital de Servicios Integrados) o conexión a la Red Telefónica Básica. Usando una única dirección IP, asignada dinámicamente por el proveedor de acceso a Internet, se conectan un número ilimitado de puestos de la red LAN a Internet mediante un único acceso WAN. Las conexiones se establecen o liberan en función de la presencia o ausencia de tráfico de forma transparente al usuario.

Si el acceso se hace a través de la RDSI, se pueden combinar el ancho de banda de los dos canales B, mediante protocolo MPPP, para acceso de alta velocidad (hasta 128 Kbps). También es posible acceder simultáneamente a Internet/redes IP públicas/Intranets, usando los dos canales B y la conexión RTB. Es decir, el CBRA soporta hasta tres conexiones simultáneas a destinos distintos.

Si el acceso se realiza mediante la RTB (Red Telefónica Básica), se debe conectar un módem externo al interfaz V.24.

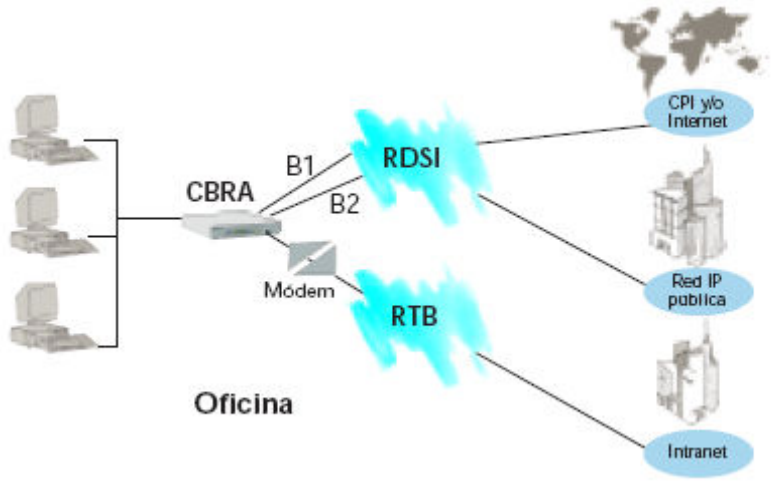


Ilustración 2: Acceso de LAN a Internet/Redes IP públicas/Intranets

1.2.1.2 Interconexión de LANs a través de Internet: VPN.

El router CBRA permite interconectar redes LAN remotas utilizando Internet a empresas que deseen crear una red privada virtual sobre una red pública IP de bajo coste.

Mediante procedimientos avanzados de túneles IP, los equipos de Teldat permiten encapsular transparentemente el tráfico de las redes LAN de oficina, sin necesidad de reservar direcciones fijas de Internet a las redes remotas. Esto permite tanto el acceso de terminales remotos a servidores de aplicaciones de oficina central, como la interconexión de las redes LAN remotas entre sí para aplicaciones distribuidas, e-mail, transferencia de ficheros, etc. Todo ello a tarifa de llamada local.

Para este servicio se utiliza acceso WAN RDSI. Esto permite una agregación dinámica de canales B, dependiendo de la demanda de tráfico saliente, usando MPPP de hasta 128 Kbps. Los túneles pueden ser dedicados a redes LAN específicas o reutilizables dinámicamente para cualquier LAN remota.

1.2.1.3 Visibilidad de subredes y servicios en Internet y/o en redes.

El router CBRA permite, opcionalmente, tener en su LAN servidores accesibles desde Internet y/o las redes IP públicas. Este funcionamiento es totalmente compatible con el funcionamiento normal del equipo, en el cual el usuario tiene una red con direccionamiento privado, desde la cual varios usuarios pueden acceder a Internet, Redes públicas IP o Intranets simultáneamente.

Otra posibilidad es la facilidad de tener servidores visibles en Internet utilizando una única dirección IP pública fija. Esta modalidad permite disponer de un número de máquinas con servicios distintos visibles en Internet y en las redes IP públicas, con la única condición de que las máquinas que deseen estar visibles no repitan servicios del mismo tipo.

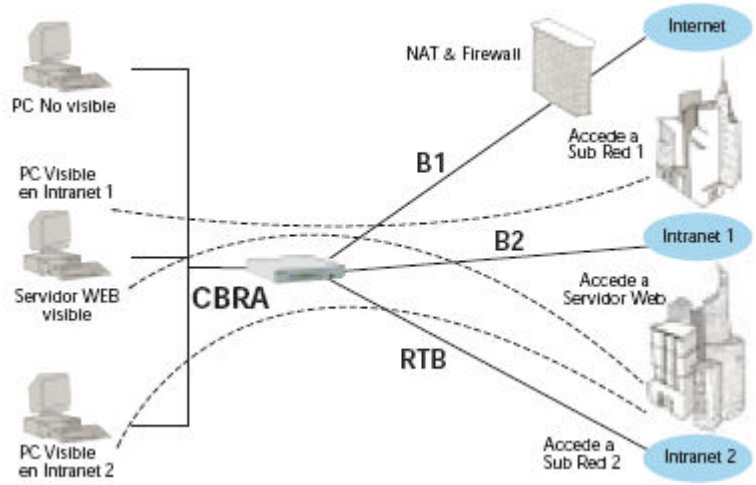


Ilustración 3: Visibilidad de subredes y servicios en Internet y/o en redes

1.2.1.4 Conectividad garantizada: backup de llamadas.

El router CBRA permite configurar conexiones de backup. Esta facilidad garantiza que, cuando una conexión definida como principal tiene problemas para establecerse, automáticamente intenta establecer la comunicación a través de una conexión alternativa definida para dar backup. La conexión de backup podrá establecerse vía RDSI o vía RTB, al mismo número llamante o a otro distinto.

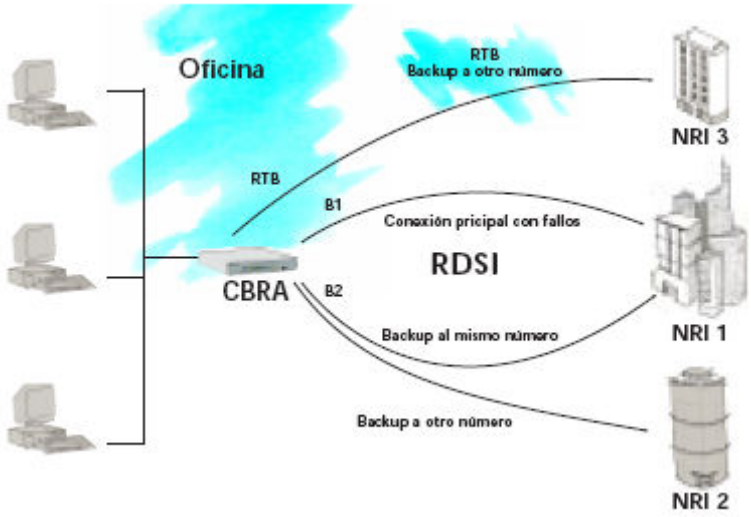


Ilustración 4: Conectividad garantizada: backup de llamadas.

1.2.1.5 Instalación y configuración inmediata.

El CBRA puede ser fácilmente instalado y configurado mediante la utilización de un programa Windows (WINCBRA), especialmente diseñado para facilitar al usuario esta labor accediendo al equipo a través del puerto de consola. Sucesivas actualizaciones de código pueden ser cargadas de manera análoga a través del mismo puerto.

A pesar de la existencia del programa WinCBRA, con la configuración por consola mediante HyperTerminal (en Windows) o Kermit (en Linux) se tendrá acceso a más opciones.

1.2.1.6 Controles de acceso a Internet.

Para garantizar el control de acceso a Internet, el CBRA dispone de filtros que imposibilitan a ciertos usuarios de la red local acceder a ciertas direcciones y servicios de Internet. Existe la posibilidad de permitir el acceso al exterior únicamente a determinadas horas del día y distinto para cada día de la semana. De esta manera las corporaciones pueden impedir el acceso fuera de las horas de trabajo, o limitarlo a un par de horas y solo a determinados usuarios (para aumentar la productividad de los trabajadores).

1.2.1.7 Seguridad frente al exterior.

El CBRA soporta mecanismos de seguridad adicionales que impiden que el tráfico externo indeseado pueda alcanzar nuestra LAN. Con el protocolo NAT (Network Address Translation) y la utilización de una única dirección IP dinámica en la WAN, las direcciones IP de la LAN resultan invisibles e inalcanzables al mundo exterior. Esto evita que los piratas informáticos puedan hacer telnet a las máquinas conectadas al router. Mecanismos de filtros IP, UDP, TCP evitan que determinado tráfico exterior pueda entrar en su corporación. Cualquier tipo de gestión y acceso al equipo (local o remota) están protegidos por password.

1.2.1.8 Privacidad: cifrado de datos.

Las comunicaciones establecidas entre el CBRA y los equipos remotos pueden ir cifradas (extremo a extremo dentro de los túneles IP) para garantizar la integridad y confidencialidad de los datos transmitidos. Para ello se emplea el algoritmo RC4 en los túneles GRE.

El sistema de cifrado implementado es independiente de los tipos de host y terminales y no requiere modificar las aplicaciones ni variar los protocolos de acceso a la red. Además, la clave de cifrado es configurable de manera independiente para cada uno de los túneles establecidos en el equipo.

1.2.1.9 Compresión de datos.

El CBRA soporta compresión de datos según diversos algoritmos, tanto propietarios como compatibles con otros fabricantes.

1.2.2 Familia de routers CBRA de Teldat.

En la familia de routers CBRA de Teldat se encuentran la serie 20 y la 21. Los routers CBRA son routers con características pequeñas, orientados para su utilización en oficinas y pequeñas empresas ya que destacan por su sencillez de utilización.

Externamente dispone de las conexiones mínimas necesarias: tres tomas RJ45 (una para su conexión a red Ethernet, otra para el enlace con la RDSI y por último la que mediante un conversor se permite configurar la unidad por puerto serie) y una DB-

25 que permite, en caso de ausencia de línea RDSI, conectar un módem analógico y emplear tal dispositivo para la conexión a las distintas redes.

Internamente incluye casi exclusivamente las funciones mínimas que se han de exigir a un router de este tipo: soporte para el protocolo IP, PPP y Multilink PPP, llamada bajo demanda, verificación PAP y CHAP de usuarios, translación NAT y translación de puertos .

El router CBRA es un router apropiado si sólo se desea conectar una red local a Internet, pero si las necesidades van un poco más allá, seguramente se eche en falta algunas opciones más avanzadas proporcionadas por otros routers de gama más alta.

En lo referente al entorno de configuración por consola, hay que destacar su sencillez y su similitud en toda la familia de routers CBRA. La forma de configurar esta familia de routers no cambia de una serie a otra, siendo los menús y submenús los mismos. Así, las series que tienen más funcionalidades tan solo añaden los comandos necesarios para configurar dicha funcionalidad, sin modificar los comandos que configuran las funcionalidades comunes en esta familia de routers.

1.3 Simuladores existentes.

Dentro de los simuladores de routers existentes, los más conocidos y más utilizados son los de los routers Cisco. Algunos de los nombres más conocidos entre emuladores/simuladores de los routers Cisco podemos encontrar los siguientes:

- Cert CCNA Router Simulator.

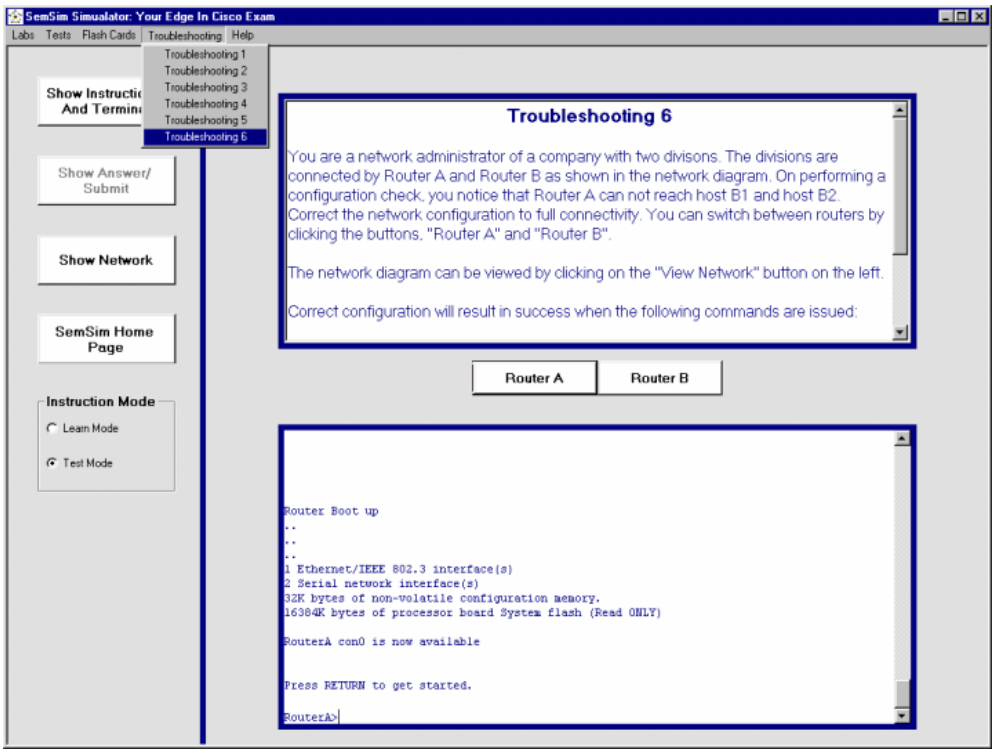


Ilustración 5: Captura Cert CCNA Router Simulator.

- Boson Router Simulator.

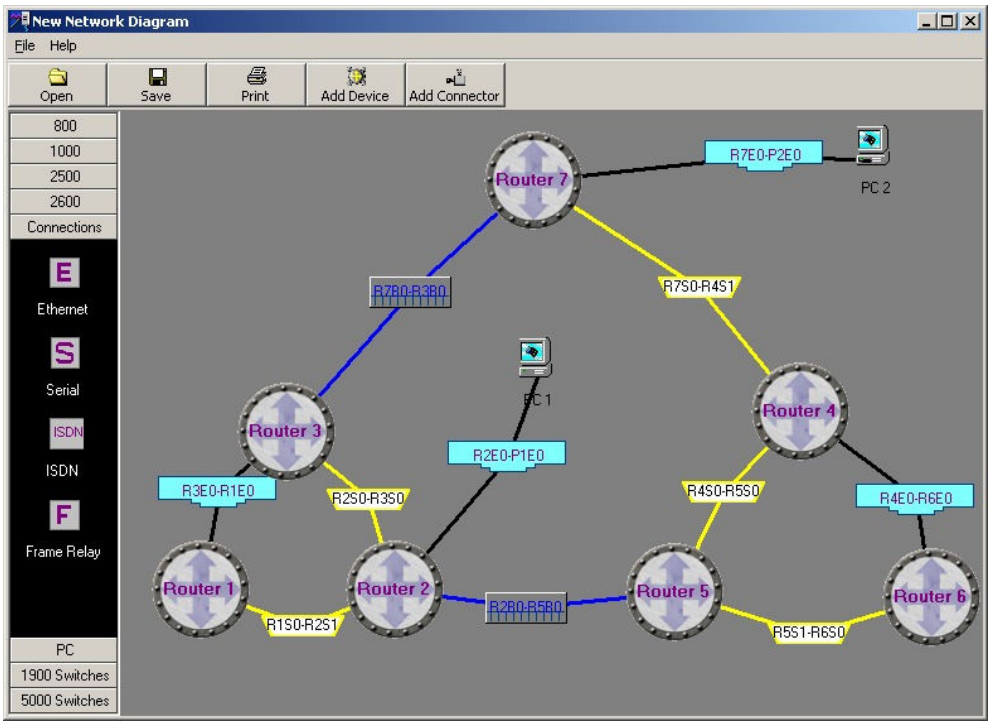


Ilustración 6: Captura Boson Router Simulator.

- RouterSim

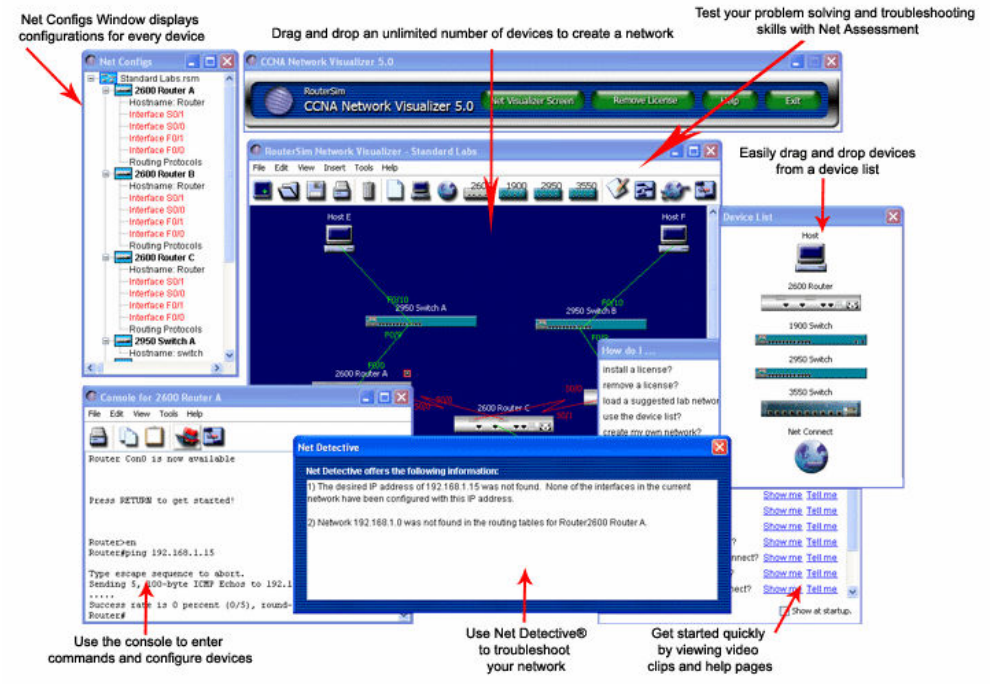


Ilustración 7: Captura RouterSim.

1.4 Diferencias entre emulador y simulador.

Un emulador es un software que permite ejecutar programas de ordenador en una plataforma (arquitectura hardware o sistema operativo) diferente de la cual fueron escritos originalmente. A diferencia de *un simulador,* que sólo *trata de reproducir el comportamiento del programa que se simula*, un emulador trata de modelar de forma precisa el dispositivo que se esta emulando.

Una vez leídas las definiciones de emulador y simulador, se llega fácilmente a la conclusión de que la herramienta software creada es un *simulador* y no un emulador, ya que no implementa la funcionalidad como tal, tan solo muestra el comportamiento que tendría el proceso de configuración del router.

Para tener esto más claro, un emulador de un router permitiría que un PC pudiese funcionar como un router, en este caso funcionar como lo hace el router CBRA. Un simulador de un router tan solo reproduce lo que haría el software del router sin implementar su funcionalidad interna.

1.5 Objetivos del Proyecto Fin de Carrera.

La finalidad principal de este Proyecto Fin de Carrera es la de proporcionar un *simulador* del router CBRA de Teldat con fines didácticas. La intención buscada es facilitar el aprendizaje en el proceso de configuración de un router de estas características, utilizando para ello una interfaz sencilla, aclaraciones y fácil acceso a documentación para su correcta configuración.

Debido al elevado coste que puede tener un router, se hace inviable para la mayoría de los alumnos el poder tener un router en casa para poder practicar el proceso básico de configuración. Tras analizar esto, surgió la idea de crear un *simulador* que pudiese ser utilizado por el alumnado tanto en casa como en simulaciones en el laboratorio.

Una vez se decidió la implementación de este *simulador*, dado que la implementación de todas las opciones no era necesario, se decidió implementar sólo aquellas que serían más necesarias para los casos más comunes de configuración. No se pretendía que esta herramienta se quedase sólo en esa funcionalidad básica, por lo que se optó por una implementación que permitiera su futura extensibilidad.

Gracias a Java y su programación orientada a objetos se consigue que el *simulador* pueda ser ampliado, permitiendo a futuros desarrolladores que implementan más funcionalidades o incluso la utilización de los interfaces creados para implementar *simuladores* de otros routers.

Además de lo mencionado, Java es un lenguaje adecuado para este tipo de fines por su portabilidad. La misma implementación podrá ser utilizada en cualquier Sistema Operativo, permitiendo ahorrar el tener que compilar el *simulador* para otra plataforma o implementarlo en otro lenguaje.

Una vez han quedado claros los objetivos perseguidos y las decisiones principales tomadas, se explicará a continuación el contenido de los próximos capítulos:

- Capítulo 2: Arquitectura del código. Se proporcionará toda la información necesaria para que todo aquél que desee implementar nuevas funcionalidad al *simulador* pueda hacerlo. En este capítulo se incluirán unos diagramas UML del código para una mayor comprensión.

- Capítulo 3: Manual del *simulador*. Información que necesitará cualquier persona que quiera utilizar el simulador:
    - Instalación.
    - Uso.
    - Ayuda.
    - Árbol de menús disponibles.
- Capítulo 4: Conclusiones y líneas futuras. Se explicarán brevemente las conclusiones, indicando los conocimientos obtenidos con este Proyecto Fin de Carrera así como comentar la experiencia tenida con Java

Capítulo 2
Arquitectura del sistema

2.1 Introducción a la implementación.

A la hora de implementar el código del simulador se ha buscado aprovechar las capacidades de Java, utilizando la Programación Orientada a Objetos. Esta propiedad consigue que sea una adecuada elección para la implementación de un *simulador* ya que facilita su extensibilidad.

En los siguientes puntos se explicará todo lo necesario para que cualquier programador pueda implementar nuevas funcionalidades al *simulador*. Para ello se seguirá el siguiente orden en los próximos puntos:

- Explicación de las principales características de Java para los que desconozcan este lenguaje.
- Ilustración de los diagramas UML del simulador para que los programadores conozcan la estructura del *simulador*.
- Pasos detallados para añadir nuevas funcionalidades al *simulador*.
- Recomendaciones para desarrolladores.

2.2 JAVA.

En este apartado se explicarán las principales características de Java para que cualquier programador que desconozca este lenguaje y conozca otros lenguajes similares pueda conocer sus posibilidades.

Entre las características de Java las más destacadas serán la herencia, el polimorfismo y los interfaces:

- La herencia es una propiedad esencial de la Programación Orientada a Objetos que consiste en la creación de nuevas clases a partir de otras ya existentes, lo que permite la reutilización del código.
- El polimorfismo es otra de las propiedades de la Programación Orientada a Objetos. Gracias al polimorfismo declararemos métodos abstractos en las clases padres que serán implementados en las clases hijos.
- Una interfaz es una colección de declaraciones de métodos (sin definirlos), que también puede incluir constantes.

2.2.1 ¿Por qué Java?

A continuación se nombrarán brevemente las principales ventajas de Java frente a otros lenguajes de programación a modo de justificación de la elección de este lenguaje para la programación del *simulador*:

- Fiabilidad del código y facilidad de desarrollo de nuevo software. C y C++ son lenguajes complejos que necesitan de un elevado coste en las

pruebas y en la depuración. Java soluciona los fallos que se pueden encontrar en el lenguaje de programación C++.

- Es un lenguaje sencillo y potente.
- Es un lenguaje orientado a objetos que permite abordar la resolución de cualquier tipo de problema.
- En Java el concepto de objeto resulta sencillo y fácil de ampliar conservando elementos "no objetos", como números y otros tipos simples. Esta capacidad proporciona a Java una gran capacidad de extensibilidad.
- La ejecución del código Java es segura y fiable, y los programas no acceden directamente a la memoria del ordenador.
- Capacidad multi-hilo, robustez e integración del protocolo TCP/IP.

Java se creó con la intención de que fuese un lenguaje lo más sencillo posible, permitiendo su adaptación a cualquier entorno de ejecución. Fue creado cogiendo las mejores características de otros lenguajes de programación ya existentes y eliminando aquellas que no eran imprescindibles.

2.2.2 Sobre el lenguaje.

Java fue desarrollado basándose en C++, pero eliminando rasgos del mismo poco empleados. Básicamente, encontramos las siguientes diferencias con C++ a tener en cuenta por aquellos programadores familiarizados con C++:

- Java no soporta los tipos *struct*, *union* y *pointer*.
- No soporta *typedef* ni *#define*.
- Se distingue por su forma de manejar ciertos operadores y no permite una sobrecarga del operador.
- No soporta herencia múltiple.
- Java maneja argumentos en la línea de comandos de forma diversa a como lo hacen C o C++.
- Java cuenta con un sistema automático para asignar y liberar memoria, con lo que no es necesario utilizar las funciones previstas con este fin en C y C++ (recolectores de basura).
- En Java podemos crear los siguientes tipos de construcciones:
    - Miniaplicaciones (“applets”).
    - Aplicaciones. Se ejecutan sin necesidad de un navegador.
    - Manipuladores de protocolo. Interpretan protocolos.
    - Manipuladores de contenido. Interpretan archivos.
    - Métodos nativos. Métodos declarados en una clase Java que están implementados en C.

2.2.3 Herencia y Polimorfismo en Java.

Java permite heredar a las clases, características y conductas de una clase denominada base (Java no permite herencia múltiple). Las clases que heredan de clases base se denominan derivadas; éstas a su vez pueden ser clases bases para otras clases derivadas.

La herencia ofrece una ventaja importante: permite la reutilización del código. Una vez que una clase ha sido depurada y probada, el código fuente de dicha clase no necesita modificarse. Su funcionalidad se puede cambiar derivando una nueva clase que herede la funcionalidad de la clase base y le añada otros comportamientos. Reutilizando el código existente, el programador ahorra tiempo y dinero, ya que solamente tiene que verificar la nueva conducta que proporciona la clase derivada. (Esto será muy útil en nuestro caso para la implementación de los diferentes menús, pero lo veremos en detalle más adelante).

En el lenguaje Java, todas las clases derivan implícitamente de la clase base "Object", por lo que heredan las funciones miembro definidas en dicha clase. Las clases derivadas pueden redefinir algunas de sus funciones y definir otras nuevas.

El polimorfismo se implementa por medio de las funciones abstractas; en las clases derivadas se declara y se define una función que tiene el mismo nombre, el mismo número de parámetros y del mismo tipo que en la clase base, pero que da lugar a un comportamiento distinto, específico de los objetos de la clase derivada. Enlace dinámico significa que la decisión sobre la función a llamar se demora hasta el tiempo de ejecución del programa.

2.2.4 Interfaces.

Un interfaz es simplemente una lista de métodos no implementados. Una clase abstracta puede incluir métodos implementados y no implementados o abstractos, miembros dato constantes y otros no constantes.

Una clase solamente puede derivar ("extends") de una clase base, pero puede implementar varios interfaces. Los nombres de los interfaces se colocan separados por una coma después de la palabra reservada "implements".

2.3 Esquema de funcionalidad implementada.

Mediante un esquema se mostrarán las funcionalidades implementadas del *router* en el *simulador*. La disposición del esquema está basada en la ubicación del comando (el menú en el que se encuentra). Los comandos y menús/submenús que no aparezcan en el esquema no están implementados.

1. GESTCON
2. CONFIG
    i. ADD DEVICE FR-DIAL
    ii. ADD DEVICE FR-ISDN
    iii. ADD DEVICE PPP-DIAL
    iv. ADD DEVICE X25-DIAL
    v. ADD DEVICE ATPPP-DIAL
    vi. ADD DEVICE MPPP
    vii. ADD DEVICE TNIP
    viii. ADD DEVICE 270

ix. CLEAR ALL
x. CLEAR DEVICE
xi. CLEAR IP
xii. DELETE DEVICE
xiii. LIST DEVICE
xiv. NETWORK
    1. AT COM
        a. DISABLE AUTO-ANSWER
        b. DISABLE RING-PATTERN
        c. ENABLE AUTO-ANSWER
        d. ENABLE RING-PATTERN
        e. LIST
        f. SET CONNECTION
        g. SET CTS-CONTROL
        h. SET DIAL
        i. SET DCD-CONTROL
        j. SET DSR-CONTROL
        k. SET DTR-CONTROL
        l. SET FLOW-CONTROL
        m. SET NUMBER-RINGS
        n. SET T1-RINGS
        o. SET T2-SILENCE
        p. SET V42-CONTROL
    2. B CHANNEL: PPP
        a. DISABLE ACCESS
        b. DISABLE INCOMING
        c. DISABLE OUTGOING
        d. ENCAPSULATOR
            i. ADD USERS
            ii. DISABLE AUTHENTICATION
            iii. DISABLE CALLBACK
            iv. DISABLE CRTP
            v. DISABLE NAT
            vi. DISABLE MPPP
            vii. DISABLE RAIDUS
            viii. DISABLE RIP-NO-DIAL
            ix. DELETE USERS
            x. ENABLE AUTHENTICATION CHAP
            xi. ENABLE AUTHENTICATION PAP
            xii. ENABLE CALLBACK
            xiii. ENABLE CRTP
            xiv. ENABLE NAT
            xv. ENABLE MPPP
            xvi. ENABLE RADIUS
            xvii. ENABLE RIP-NO-DIAL
            xviii. LIST ALL
            xix. LIST LINE
            xx. LIST LCP
            xxi. LIST NCP
            xxii. LIST IPCP
            xxiii. LIST AUTHENTICATION
            xxiv. LIST FACILITY
            xxv. LIST USERS
            xxvi. LIST INTERVAL-OF-CONNECTION
            xxvii. LIST CCP

xxviii. SET AUTHENTICATION
xxix. SET CCP CHECK
xxx. SET CCP HISTORY
xxxi. SET CCP PROCESS
xxxii. SET CCP TYPE
xxxiii. SET INTERVAL-OF-CONNECTION
xxxiv. SET IPCP
xxxv. SET LCP OPTIONS
xxxvi. SET LCP PARAMETERS
xxxvii. SET LINE ENCODING NRZ
xxxviii. SET LINE ENCODING NRZI
xxxix. SET LINE IDLE MARK
xl. SET LINE FRAME-SIZE
xli. SET LINE LINE-SPEED
xlii. SET LINE TRANSMIT-DELAY
xliii. SET NCP

e. ENABLE ACCESS
f. ENABLE INCOMING
g. ENABLE OUTGOING
h. LIST
i. SET BASE-INTERFACE
j. SET DESTINATION-ADDRESS
k. SET INACTIVE-TIME
l. SET PERMITTED-CALLER
m. SET NAME-CIRCUIT

3. ISDN
   a. LIST
   b. SET CONNECTION
   c. SET LOCAL
   d. SET MAXIMUM
4. MULTILINK PPP
   a. SET DIRECTION
   b. SET FRAGMENTATION VOICE
   c. SET INTERVAL ACTIVATION
   d. SET INTERVAL DESACTIVATION
   e. SET PRE-EMPTION
   f. SET THRESHOLD ACTIVATION
   g. SET THRESHOLD DESACTIVATION
5. PPP AT COM
   a. ENCAPSULATOR
      i. ADD USERS
      ii. DISABLE AUTHENTICATION
      iii. DISABLE CALLBACK
      iv. DISABLE CRTP
      v. DISABLE NAT
      vi. DISABLE MPPP
      vii. DISABLE RAIDUS
      viii. DISABLE RIP-NO-DIAL
      ix. DELETE USERS
      x. ENABLE AUTHENTICATION CHAP
      xi. ENABLE AUTHENTICATION PAP
      xii. ENABLE CALLBACK
      xiii. ENABLE CRTP
      xiv. ENABLE NAT
      xv. ENABLE MPPP

xvi. ENABLE RADIUS
xvii. ENABLE RIP-NO-DIAL
xviii. LIST ALL
xix. LIST LINE
xx. LIST LCP
xxi. LIST NCP
xxii. LIST IPCP
xxiii. LIST AUTHENTICATION
xxiv. LIST FACILITY
xxv. LIST USERS
xxvi. LIST INTERVAL-OF-CONNECTION
xxvii. LIST CCP
xxviii. SET AUTHENTICATION
xxix. SET CCP CHECK
xxx. SET CCP HISTORY
xxxi. SET CCP PROCESS
xxxii. SET CCP TYPE
xxxiii. SET INTERVAL-OF-CONNECTION
xxxiv. SET IPCP
xxxv. SET LCP OPTIONS
xxxvi. SET LCP PARAMETERS
xxxvii. SET LINE ENCODING NRZ
xxxviii. SET LINE ENCODING NRZI
xxxix. SET LINE IDLE MARK
xl. SET LINE FRAME-SIZE
xli. SET LINE LINE-SPEED
xlii. SET LINE TRANSMIT-DELAY
xliii. SET NCP

b. LIST
c. SET DESTINATION-ADDRESS
d. SET INACTIVE-TIME

6. QUICC ETHERNET
    a. IP IEEE-802.3
    b. IP ETHERNET
    c. LIST
    d. MAC

xv. PROTOCOL IP
1. ADD ACCESS-CONTROL
2. ADD ADDRESS
3. ADD AGGREGATION-ROUTE
4. ADD FILTER
5. ADD ROUTE
6. CHANGE ADDRESS
7. CHANGE ROUTE
8. DELETE ACCESS-CONTROL
9. DELETE ADDRESS
10. DELETE ROUTE
11. LIST ACCESS-CONTROLS
12. LIST ADDRESSES.
13. LIST ROUTES
14. SET ACCESS-CONTROL OFF
15. SET ACCESS-CONTROL ON
16. SET DEFAULT NETWORK-GATEWAY
17. SET DEFAULT SUBNET-GATEWAY

xvi. SAVE

2.4 Diagramas UML de la implementación.

En este apartado se mostrarán los diagramas UML que describen la estructura del código y se explicará la finalidad básica de cada una de las clases. El diagrama UML elegido ha sido el "Diagrama de Clases" ya que con un único diagrama se pueden representar las clases que forman el *simulador*, las relaciones que existen entre ellas, los atributos y los métodos de cada clase, proporcionando toda la información necesaria para la comprensión del programa de forma general.

Debido a que algunas clases tienen un gran número de atributos y métodos, la forma de representar el "Diagrama de Clases" será mostrando primero el diagrama con todas las clases contraídas (sin mostrar los atributos y los métodos, mostrando tan solo los nombres de las clases), dando una vista general de las relaciones existentes, y posteriormente mostrando cada una de las clases expandidas por separado.

En este apartado no se proporcionará información sobre cada uno de los métodos ya que la funcionalidad de cada uno de ellos se puede ver en el API del *simulador*. El API se puede encontrar en el directorio "doc" del *simulador*, siendo el índice de dicho API el archivo "index.html".

En la implementación se ha construido una clase para cada uno de los menús/submenús existentes, por lo que al pasar de un menú/submenú a otro se crea un nuevo objeto perteneciente al menú/submenú al que se quiere ir y se destruye el objeto perteneciente al menú/submenú del que se viene.

A continuación se muestra el "Diagrama de Clases" contraído. La clase principal es "RouterTeldat", todos los menús/submenús extenderán de la clase "Menú" y todos los interfaces existentes extenderán de la clase "Interfaz".

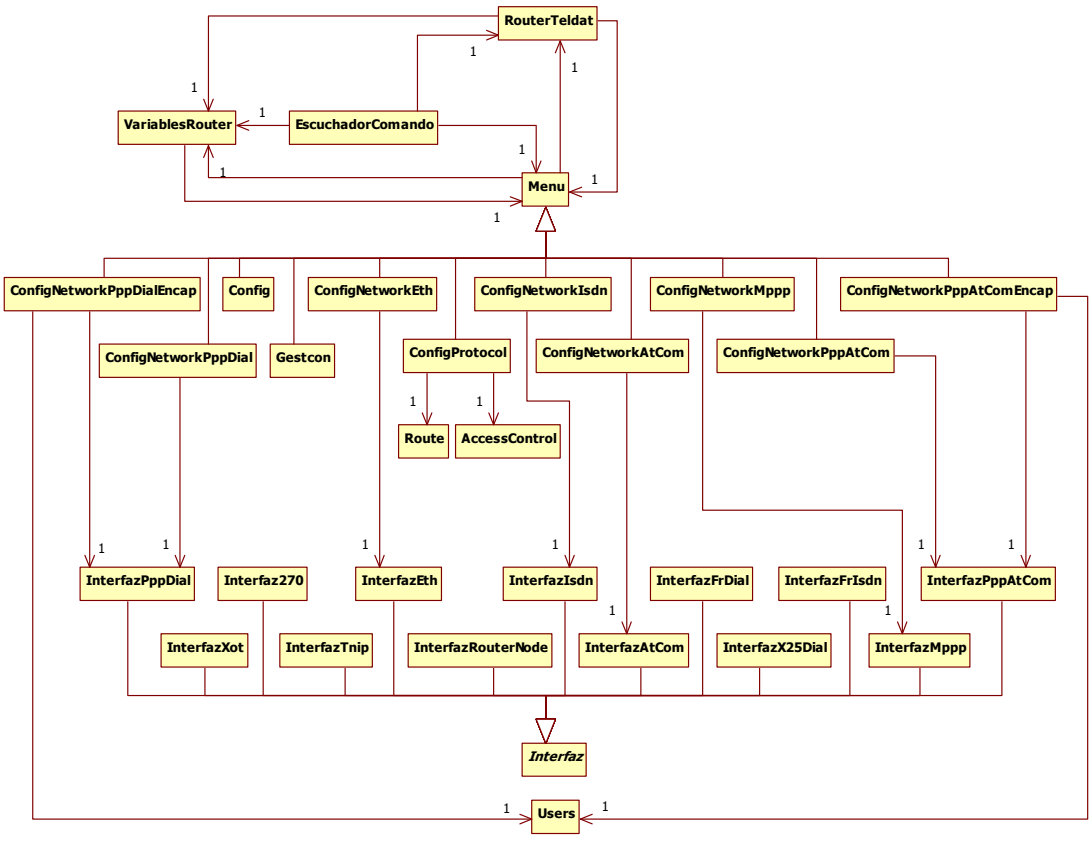


Ilustración 8: Diagrama de Clases contraído.

Una vez vista la estructura principal del *simulador*, se irán mostrando cada una de las clases de modo expandido (mostrando los atributos y los métodos de cada una de las clases implementadas). Las clases expandidas se irán mostrando por orden alfabético del nombre de la clase.

La primera clase por orden alfabético es “AccessControl”. Esta clase representa cada una de las entradas presentes en la tabla “Access Control” (relacionada con las opciones de filtrado) del menú “Config”. Cada entrada tendrá un tipo (inclusivo o exclusivo), una dirección origen, una máscara origen, una dirección destino, una máscara destino, intervalo de protocolos a los que afecta e intervalo de puertos a los que afecta tanto en el origen como en el destino.

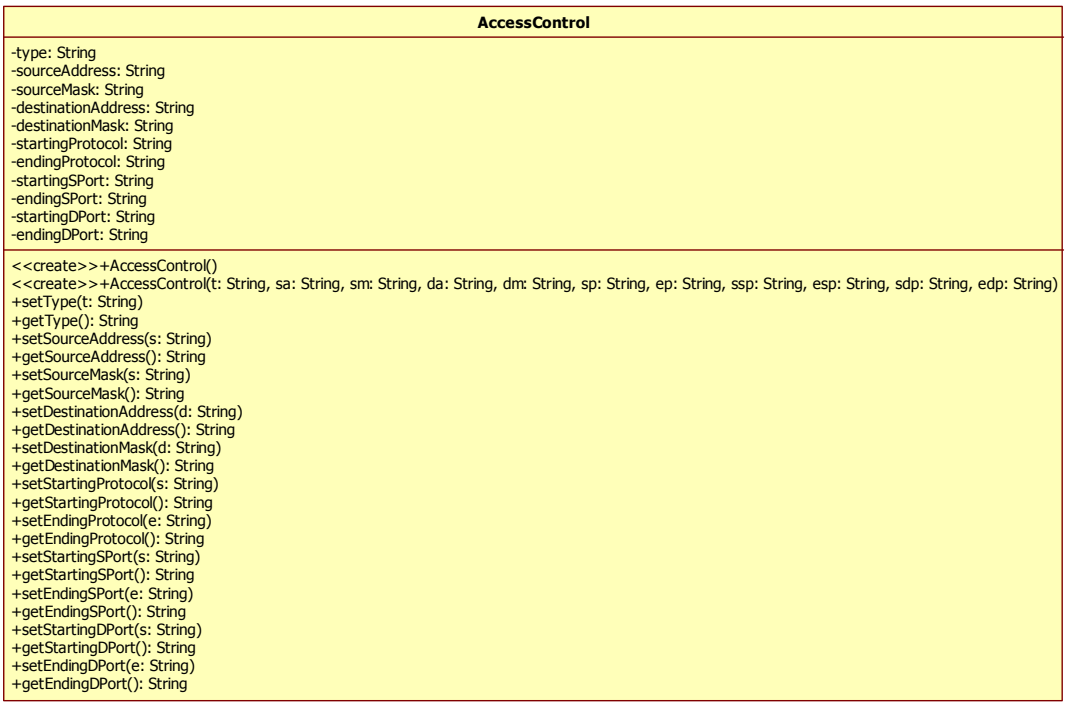


Ilustración 9: Diagrama expandido de la clase “AccessControl”.

La siguiente clase por orden alfabético es “Config”. Esta clase representa el menú “Config” (P 4 del proceso “Gestcon”) o proceso de Configuración del *router*. Permite configurar parámetros del *router* como: Interfaces y Protocolos.

Mediante el menú “Config” podremos modificar la configuración almacenada. Hay que recordar que para que los cambios tengan efecto en el *router* (en la realidad) se debe ejecutar el comando “SAVE” y después el comando “RESTART”.

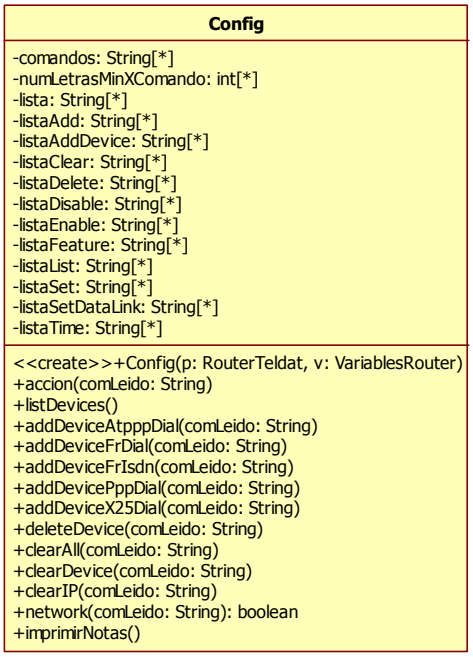


Ilustración 10: Diagrama expandido de la clase "Config".

La clase "ConfigNetworkAtCom" representa el menú desde el cual se podrá acceder a la configuración del interfaz del tipo "AT COM" seleccionado, introduciendo los comandos adecuados. En esta clase se podrá encontrar una referencia a un interfaz del tipo "AT COM", el cual tendrá una serie de métodos que permitirán modificar sus parámetros y que serán los invocados por los comandos de esta clase.

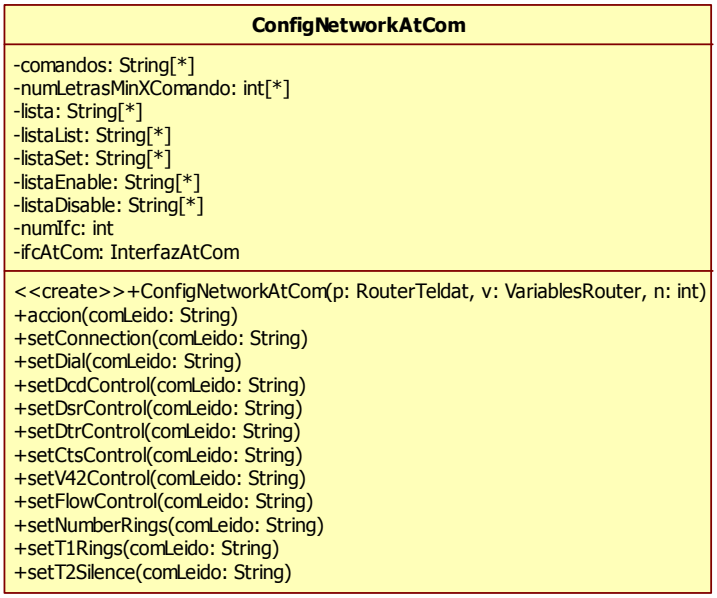


Ilustración 11: Diagrama expandido de la clase "ConfigNetworkAtCom".

La clase "ConfigNetworkEth" representa el menú desde el cual se puede modificar la configuración del interfaz del tipo "Ethernet" seleccionado. , introduciendo los comandos adecuados. En esta clase se podrá encontrar una referencia a un interfaz del tipo "Ethernet", el cual tendrá una serie de métodos que permitirán modificar sus parámetros y que serán los invocados por los comandos de esta clase.

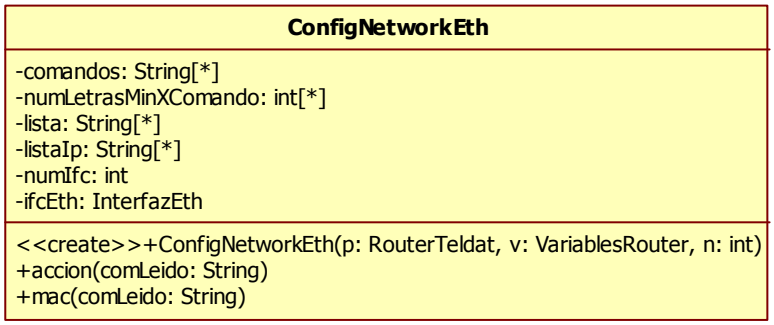


Ilustración 12: Diagrama expandido de la clase "ConfigNetworkEth".

La clase "ConfigNetworkIsdn" representa el menú desde el cual se puede acceder a la configuración del interfaz del tipo "ISDN" (RDSI) seleccionado, introduciendo los comandos adecuados. En esta clase se podrá encontrar una referencia a un interfaz del tipo "ISDN", el cual tendrá una serie de métodos que permitirán modificar sus parámetros y que serán los invocados por los comandos de esta clase.

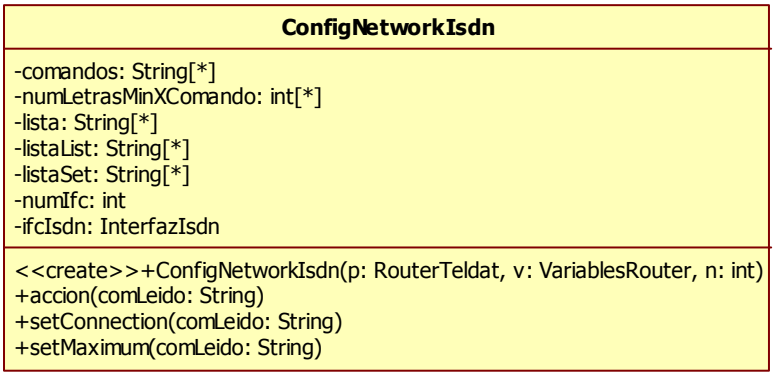


Ilustración 13: Diagrama expandido de la clase "ConfigNetworkIsdn".

La clase "ConfigNetworkMppp" representa el menú desde el cual se puede acceder a la configuración del interfaz "MPPP" seleccionado, introduciendo los comandos adecuados. En esta clase se podrá encontrar una referencia a un interfaz del tipo "MPPP", el cual tendrá una serie de métodos que permitirán modificar sus parámetros y que serán los invocados por los comandos de esta clase.

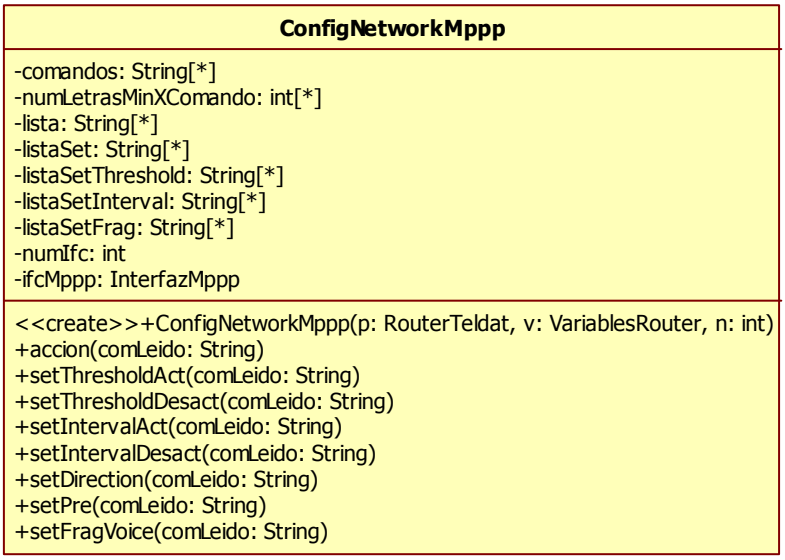


Ilustración 14: Diagrama expandido de la clase "ConfigNetworkMppp".

La clase "ConfigNetworkPppAtCom" representa el menú desde el cual se puede acceder a la configuración del interfaz "PPP AT COM" seleccionado, introduciendo los comandos adecuados. En esta clase se podrá encontrar una referencia a un interfaz del tipo "PPP AT COM", el cual tendrá una serie de métodos que permitirán modificar sus parámetros y que serán los invocados por los comandos de esta clase.

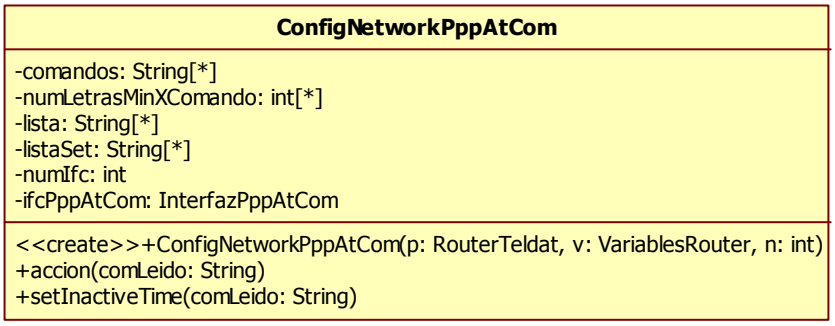


Ilustración 15: Diagrama expandido de la clase "ConfigNetworkPppAtCom".

La clase "ConfigNetworkPppAtComEncap" representa el menú desde el cual se puede acceder a la configuración de encapsulado del interfaz "PPP AT COM" seleccionado.

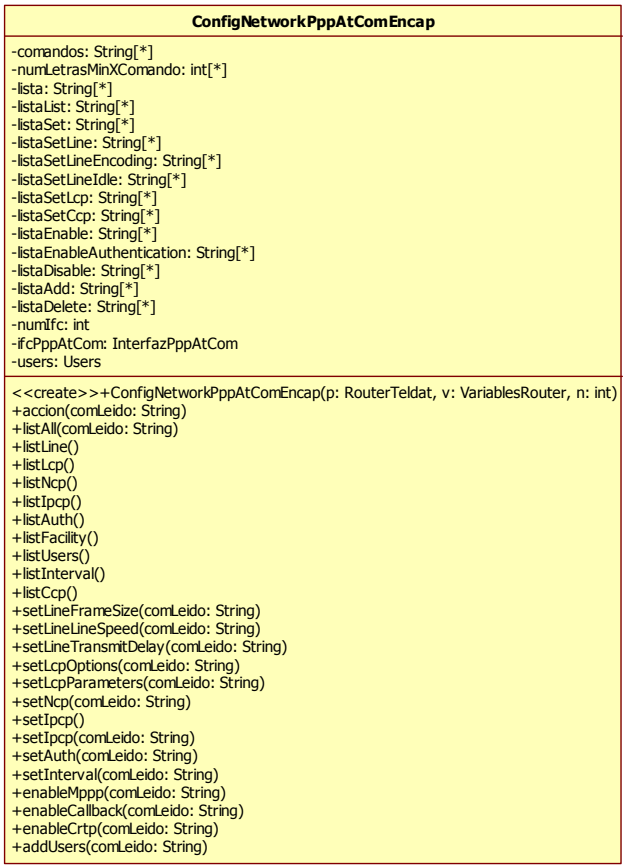


Ilustración 16: Diagrama expandido de la clase "ConfigNetworkPppAtComEncap".

La clase "ConfigNetworkPppDial" representa el menú desde el cual se puede acceder a la configuración del interfaz "PPP DIAL" seleccionado, introduciendo los comandos adecuados. En esta clase se podrá encontrar una referencia a un interfaz del tipo "PPP DIAL", el cual tendrá una serie de métodos que permitirán modificar sus parámetros y que serán los invocados por los comandos de esta clase.

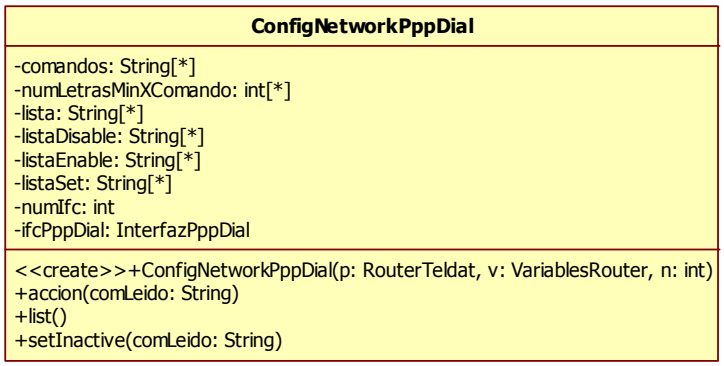


Ilustración 17: Diagrama expandido de la clase "ConfigNetworkPppDial".

La clase “ConfigNetworkPppDialEncap” representa el menú desde el cual se puede acceder a la configuración de encapsulado del interfaz “PPP DIAL” seleccionado.

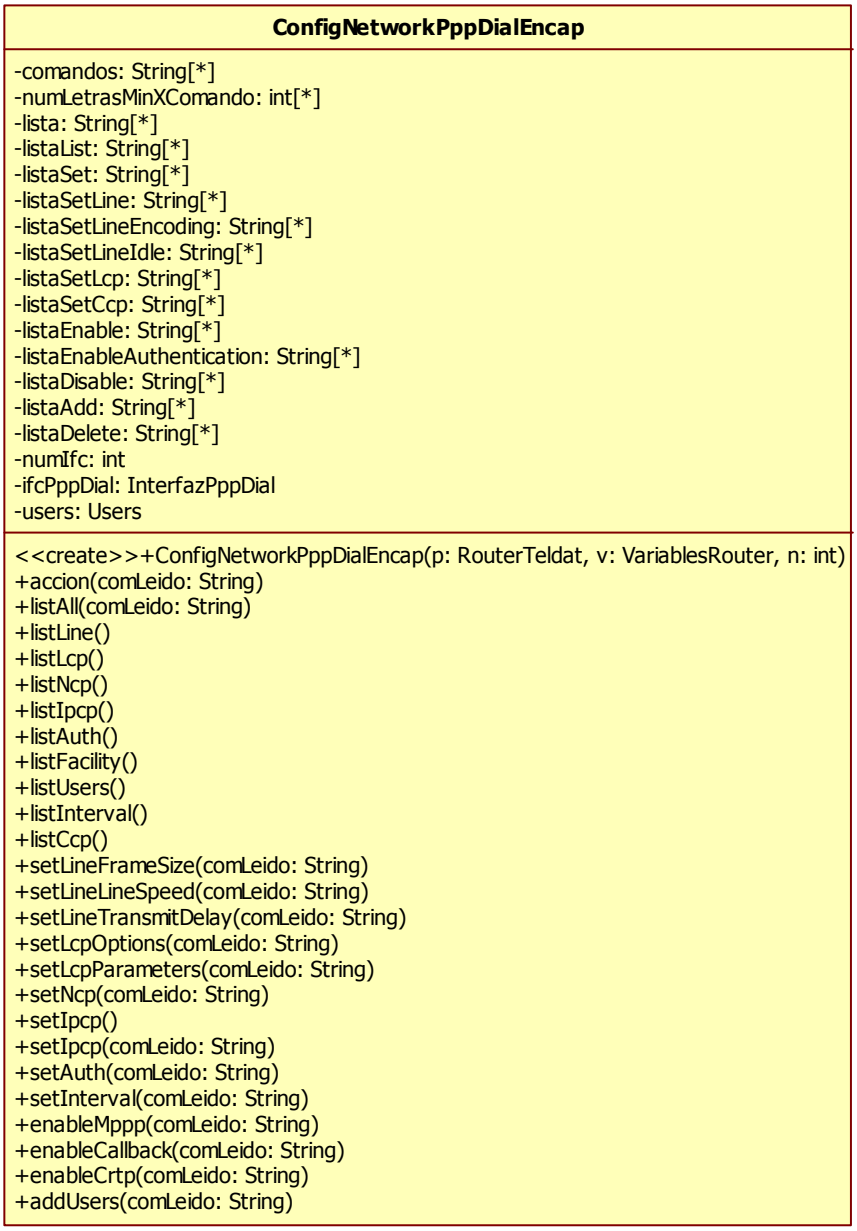


Ilustración 18: Diagrama expandido de la clase “ConfigNetworkPppDialEncap”.

La clase “ConfigProtocol” representa el menú desde el cual se puede acceder a la configuración del protocolo Internet. Desde este menú se podrán configurar todas las opciones del protocolo IP.

Las configuraciones más destacables a las que se puede acceder desde este menú son: la asignación/eliminación/modificación de direcciones IP a los diferentes interfaces, la agregación/eliminación/modificación de rutas a la tabla de rutas y la creación de la tabla de control de acceso para las opciones de filtrado.

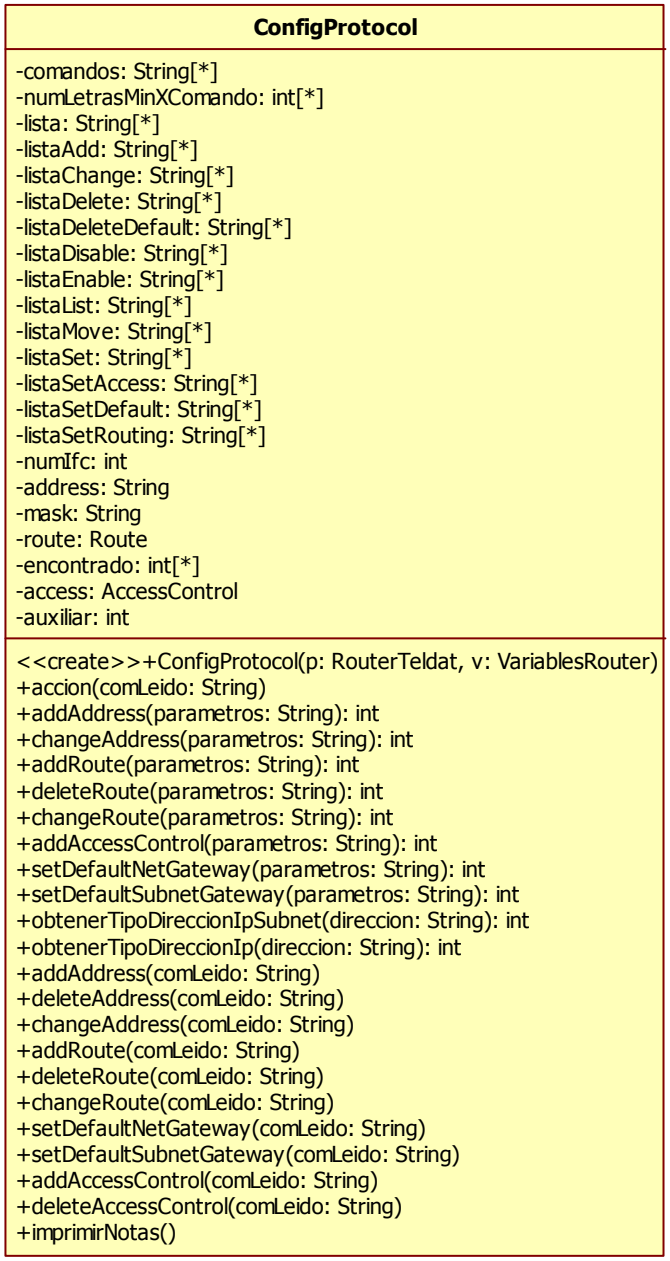


Ilustración 19: Diagrama expandido de la clase "ConfigProtocol".

La clase "EscuchadorComando" es la encargada de capturar todos los eventos que tengan lugar en la consola donde se introducen los comandos. Estos eventos serán básicamente los nuevos comandos introducidos y la combinación Ctrl+P.

Cada vez que se introduce un nuevo comando será capturado y el String obtenido será mandado al método "accion(comLeido:String)" del objeto que represente el menú/submenú en el que nos encontremos. El "EscuchadorComando" conocerá el menú/submenú al que mandar el String obtenido, gracias a la variable "actual" de la clase principal.

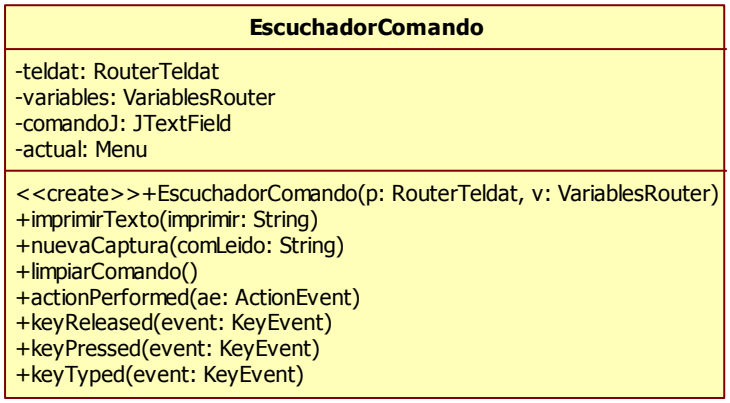


Ilustración 20: Diagrama expandido de la clase "EscuchadorComando".

La clase "Gestcon" representa el menú o proceso principal. Es el primer menú en mostrarse al iniciar el *simulador* (equivalente a iniciar el *router*). Desde el proceso principal (Gestcon) se podrá acceder a todos los demás procesos. De todos los procesos existentes, solo queda implementado el proceso "Config" ya que los demás procesos no son necesarios en un *simulador* (se necesita del hardware que viene en el *router* para los demás procesos).

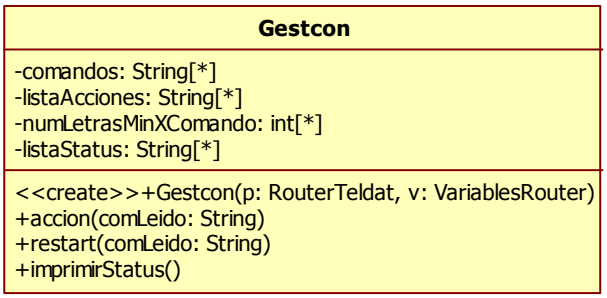


Ilustración 21: Diagrama expandido de la clase "Gestcon".

La clase "Interfaz" es la clase desde la que heredarán los diferentes tipos de interfaces. Esta clase contiene todos aquellos atributos y métodos comunes a todos los interfaces como son:

- "con": conector físico al que corresponde el interfaz.
- "ifc": identificador del interfaz.
- "type of interface": tipo de interfaz programado.
- "addresses": lista de direcciones IP asignadas al interfaz.
- "mask": lista de máscaras asociadas a cada una de las direcciones IP de la lista "addresses".

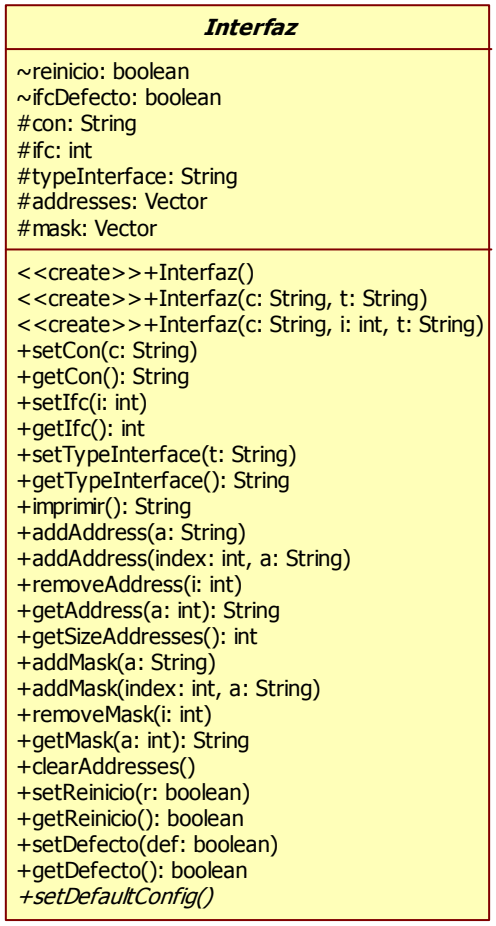


Ilustración 22: Diagrama expandido de la clase "Interfaz".

La clase "Interfaz270" representa un interfaz del tipo "270" y no está completamente implementada, tan solo se permite añadir interfaces de este tipo pero no se permite modificar su configuración.

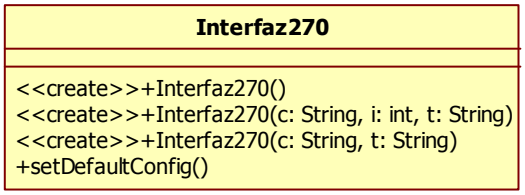


Ilustración 23: Diagrama expandido de la clase "Interfaz270".

La clase "InterfazAtCom" representa un interfaz del tipo "AT COM". Esta clase contendrá todos los atributos y métodos necesarios para implementar la funcionalidad de un interfaz de este tipo en el *router* y para contener todos los parámetros necesarios para un interfaz de este tipo.

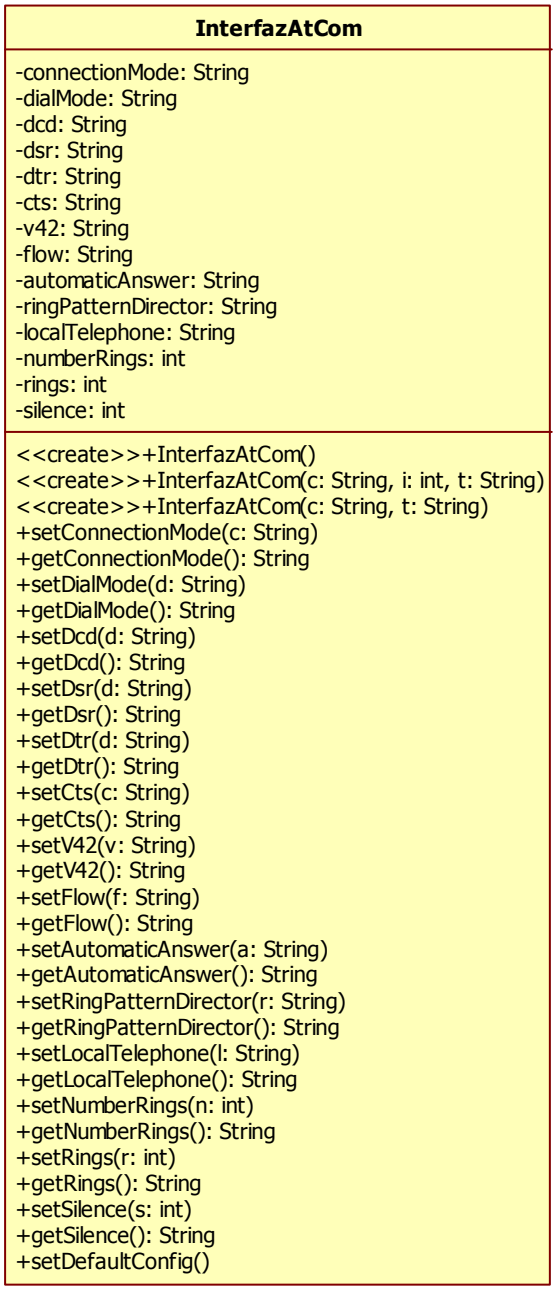


Ilustración 24: Diagrama expandido de la clase "InterfazAtCom".

La clase "InterfazEth" representa un interfaz del tipo "Ethernet". Esta clase contendrá todos los atributos y métodos necesarios para implementar la funcionalidad de un interfaz de este tipo en el *router* y para contener todos los parámetros necesarios para un interfaz de este tipo.

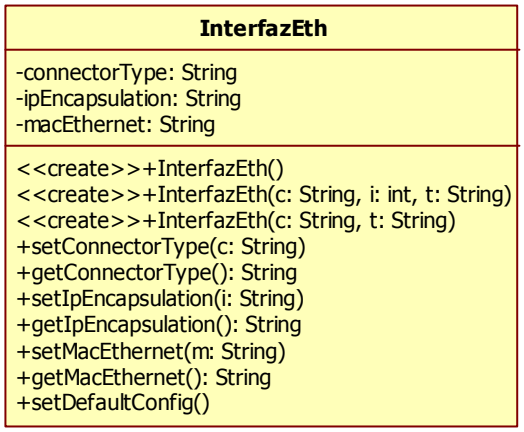


Ilustración 25: Diagrama expandido de la clase "InterfazEth".

La clase "InterfazFrDial" representa un interfaz del tipo "FR DIAL" y no está completamente implementada, tan solo se permite añadir interfaces de este tipo pero no se permite modificar su configuración.

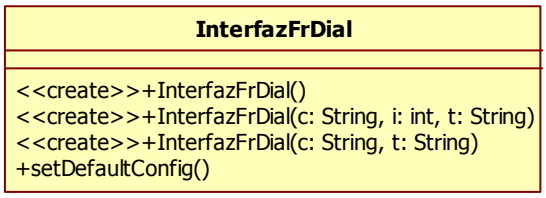


Ilustración 26: Diagrama expandido de la clase "InterfazFrDial".

La clase "InterfazFrIsdn" representa un interfaz del tipo "FR ISDN" y no está completamente implementada, tan solo se permite añadir interfaces de este tipo pero no se permite modificar su configuración.

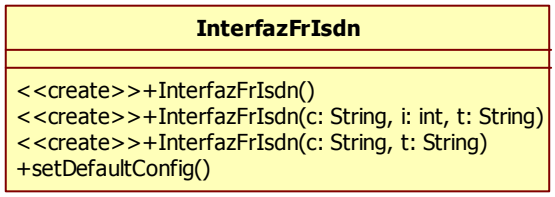


Ilustración 27: Diagrama expandido de la clase "InterfazFrIsdn".

La clase "InterfazIsdn" representa un interfaz del tipo "ISDN". Esta clase contendrá todos los atributos y métodos necesarios para implementar la funcionalidad de un interfaz de este tipo en el *router* y para contener todos los parámetros necesarios para un interfaz de este tipo.

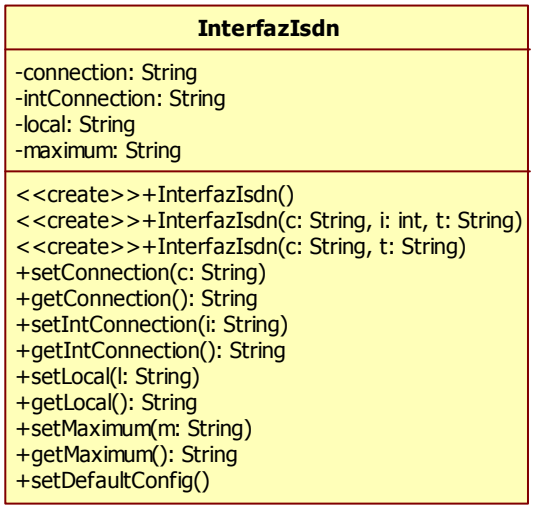


Ilustración 28: Diagrama expandido de la clase "InterfazIsdn".

La clase "InterfazMppp" representa un interfaz del tipo "MPPP". Esta clase contendrá todos los atributos y métodos necesarios para implementar la funcionalidad de un interfaz de este tipo en el *router* y para contener todos los parámetros necesarios para un interfaz de este tipo.

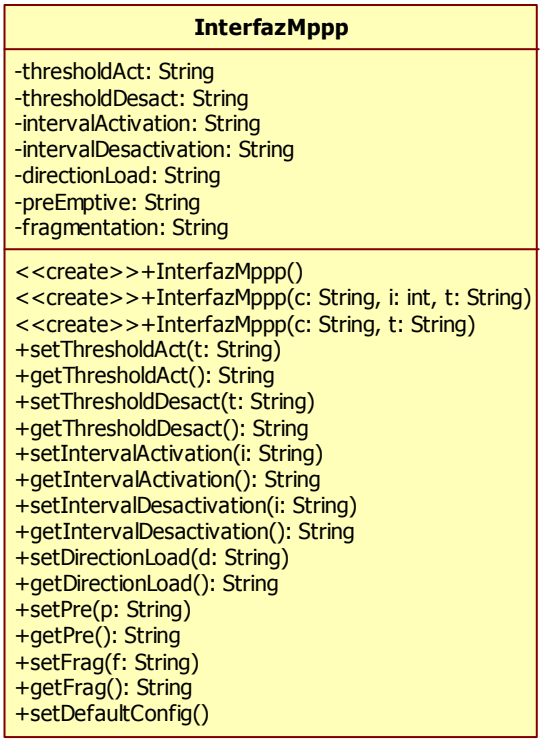


Ilustración 29: Diagrama expandido de la clase "InterfazMppp".

La clase “InterfazPppAtCom” representa un interfaz del tipo “PPP AT COM”. Esta clase contendrá todos los atributos y métodos necesarios para implementar la funcionalidad de un interfaz de este tipo en el *router* y para contener todos los parámetros necesarios para un interfaz de este tipo.

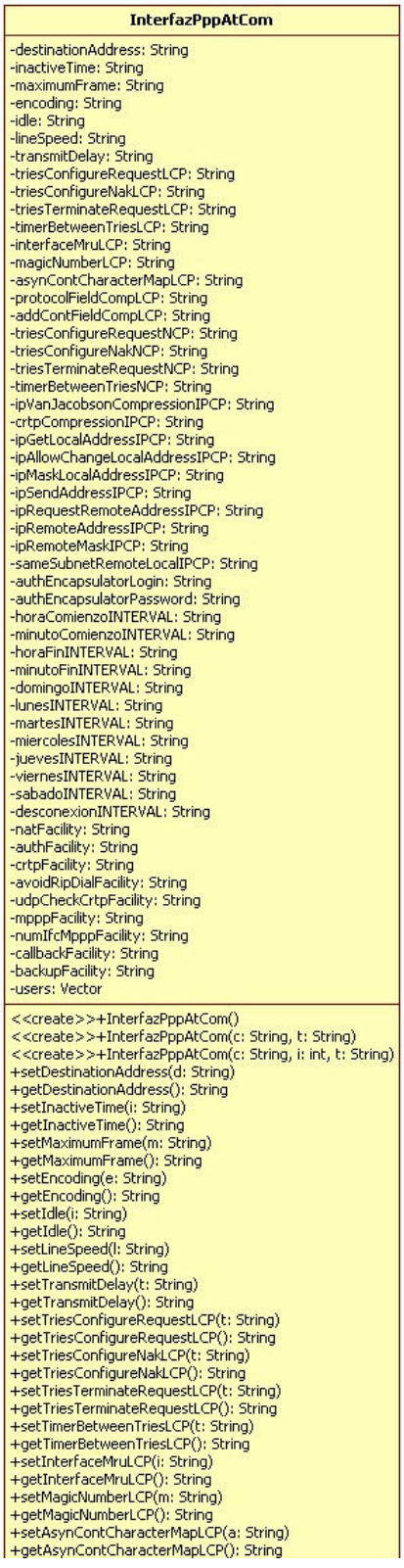


Ilustración 30: Diagrama expandido de la clase "InterfazPppAtCom".

La clase “InterfazPppDial” representa un interfaz del tipo “PPP DIAL”. Esta clase contendrá todos los atributos y métodos necesarios para implementar la funcionalidad de un interfaz de este tipo en el *router* y para contener todos los parámetros necesarios para un interfaz de este tipo.

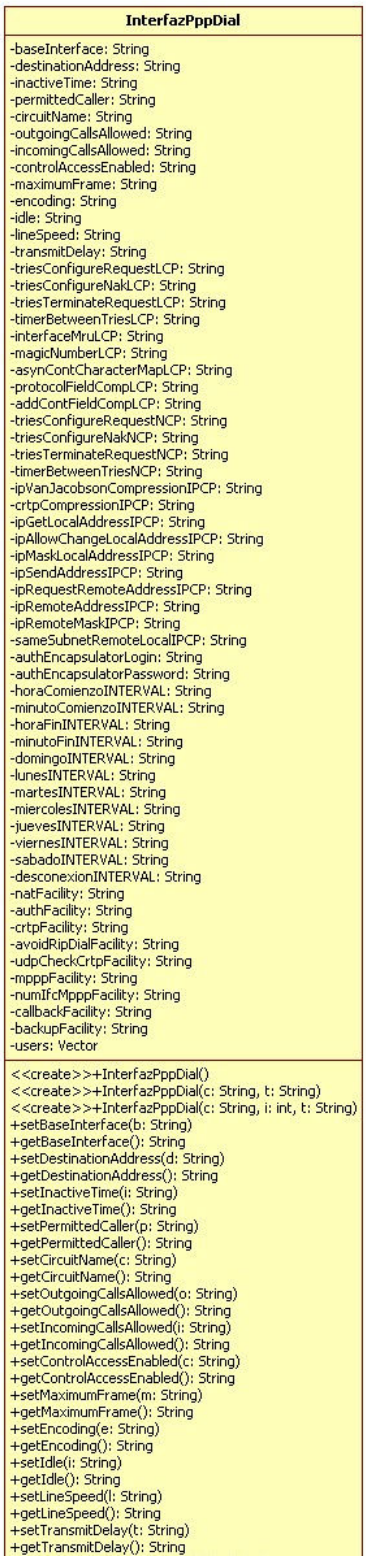


Ilustración 31: Diagrama expandido de la clase “InterfazPppDial”.

La clase “InterfazRouterNode” representa un interfaz del tipo “ROUTER->NODE” y este interfaz permite unir los equipos virtuales “Router” y “Node”. Recordar que el *router* estará formado por dos equipos virtuales “Router” y “Node” (para obtener más información sobre este tema se deberá visitar la documentación del *router* o ver el fichero “ayuda.html”).

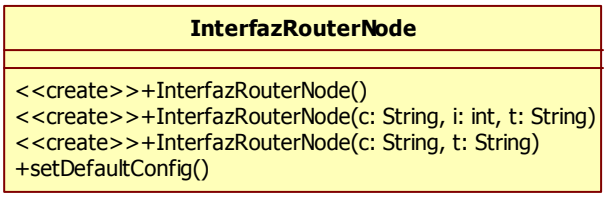


Ilustración 32: Diagrama expandido de la clase “InterfazRouterNode”.

La clase “InterfazTnip” representa un interfaz del tipo “TNIP” y no está completamente implementada, tan solo se permite añadir interfaces de este tipo pero no se permite modificar su configuración.

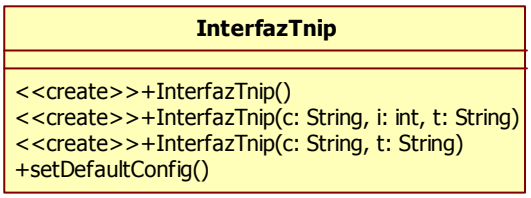


Ilustración 33: Diagrama expandido de la clase “InterfazTnip”.

La clase “InterfazX25Dial” representa un interfaz del tipo “X25 DIAL” y no está completamente implementada, tan solo se permite añadir interfaces de este tipo pero no se permite modificar su configuración.

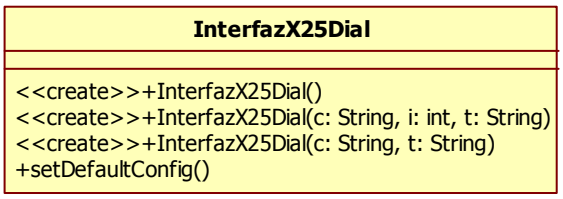


Ilustración 34: Diagrama expandido de la clase “InterfazX25Dial”.

La clase “InterfazXot” representa un interfaz del tipo “XOT” y no está completamente implementada, tan solo se permite añadir interfaces de este tipo pero no se permite modificar su configuración.

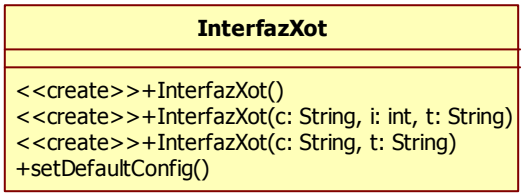


Ilustración 35: Diagrama expandido de la clase "InterfazXot".

La clase "Menu" es la clase de la que heredarán todas las demás clases que representen a los diferentes menús y submenús del router. Esta clase tiene como principal característica el método "comprobarComando(comando:String)" que es el encargado de encontrar si el comando introducido es válido (incluido si se introduce de forma abreviado con sus letras mínimas). Además tiene otros métodos muy útiles que podrán utilizarse en los nuevos menús/submenús que se puedan implementar en el futuro (consultar el API para más información de los demás métodos).

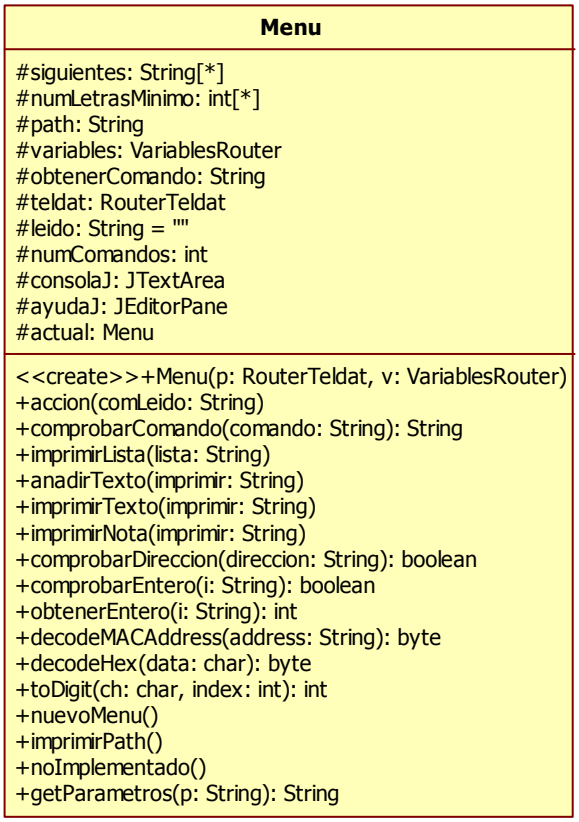


Ilustración 36: Diagrama expandido de la clase "Menu".

La clase "Route" representa cada una de las entradas presentes en la tabla de rutas del menú "Config". Cada entrada estará formada por la dirección IP de la red de destino, su máscara de red, el gateway necesario para llegar a esa red y el coste necesario.

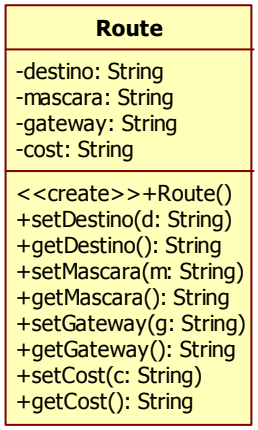


Ilustración 37: Diagrama expandido de la clase "Route".

La clase "RouterTeldat" será la clase principal que contendrá el método "public static void main (String args[])" y todos los elementos del entorno gráfico. Esta clase será la que inicialice todas las variables y la que se encargue de la gestión del interfaz gráfico, conteniendo los métodos adecuados para ello.

Entre los métodos, encontraremos algunos con las siguientes finalidades:

- Imprimir texto en el cuadro de texto de la salida del *simulador*.
- Imprimir texto en el cuadro de texto de las notas.
- Cargar todas las variables del fichero de configuración.
- Mostrar el fichero de ayuda en una nueva pestaña.
- Mostrar el fichero de ayuda en el navegador del sistema operativo.

Esta clase, además, creará la primera referencia de un menú, que será el proceso "Gestcon". Cada vez que se cambie de menú/submenú se notificará a esta clase y todas las demás clases podrán obtener el menú/submenú en el que nos encontremos.

Ilustración 38: Diagrama expandido de la clase "RouterTeldat" (Clase Principal).

La clase "Users" representa cada una de las entradas para los casos en los que haya que introducir usuarios autorizados para acceder al router por el interfaz correspondiente. Cada entrada tendrá un "login" y un "password".

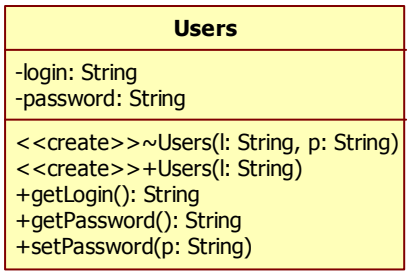
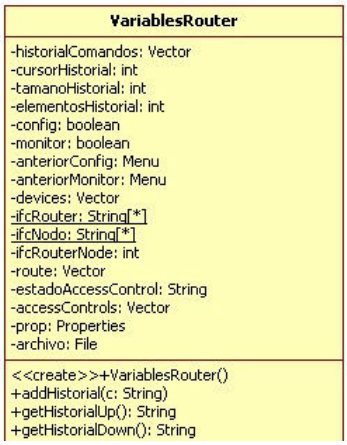


Ilustración 39: Diagrama expandido de la clase "Users".

La clase "VariablesRouter" contendrá todas las variables representativas de todos los parámetros del router y los métodos correspondientes para obtener y modificar dichas variables. Dispone, además, de métodos para permitir cargar y guardar la configuración.



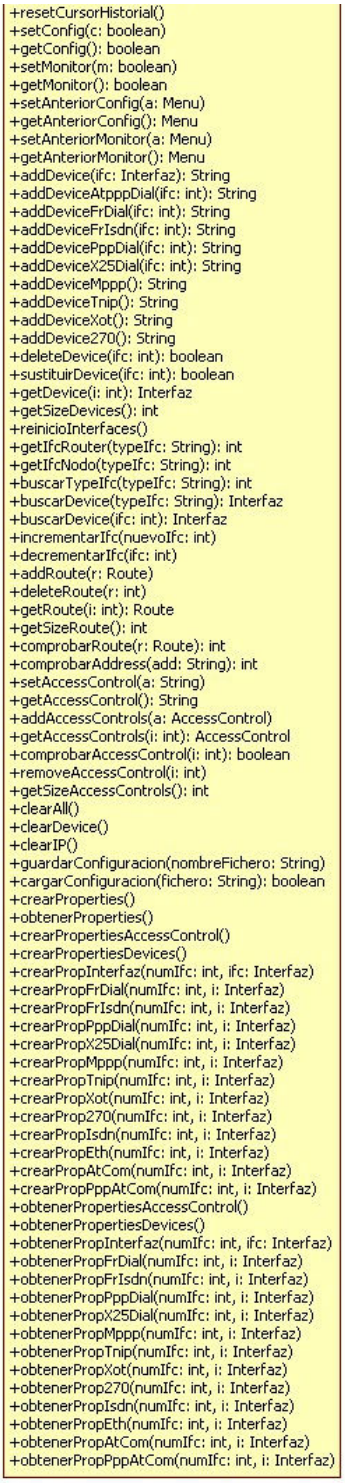


Ilustración 40: Diagrama expandido de la clase "VariablesRouter".

NOTA: Si se desean ver los diagramas UML con mejor claridad, se pueden encontrar en el directorio "Diagramas UML" del CD del proyecto.

2.5 Instrucciones para desarrolladores.

Para los desarrolladores que deseen ampliar la funcionalidad del "*simulador"* se proporcionarán unas instrucciones básicas. En estas instrucciones no se describirá el código a utilizar (ya implementado, y que se puede leer en el API), tan solo se proporcionarán unos pasos generales a seguir a modo de guía.

En el *simulador* hay muchas funcionalidades que agregar o modificar. Se explicarán los tres tipos más importantes de funcionalidades a implementar:

1. Implementar funcionalidades en interfaces.
2. Implementar comandos no implementados en menús o submenús.
3. Implementar nuevos menús o submenús.

2.5.1 Implementar funcionalidades en interfaces.

Todos los tipos de interfaces que se pueden configurar en el *router* están representados mediante una clase. De todos los tipos existentes, no todos los interfaces están completamente implementados. Para poder añadir nueva funcionalidad a los diferentes interfaces se recomienda seguir los siguientes pasos:

1. Añadir los parámetros (atributos) deseados en la clase que represente el interfaz a ampliar (se aconseja seguir el principio de privacidad de la información).
2. Completar el constructor para los nuevos parámetros.
3. Añadir los métodos necesarios para poder obtener y modificar los parámetros introducidos en el paso anterior, implementado aquella funcionalidad que se desee añadir en dichos métodos.
4. En la clase "VariablesRouter", añadir las líneas necesarias en los métodos correspondientes al interfaz elegido para poder cargar y guardar los parámetros añadidos en el fichero de configuración (por ejemplo observar los métodos "crearPropMppp(numIfc:int,i:Interfaz)" y "obtenerPropMppp(numIfc:int,i:Interfaz)" para entender como cargar y guardar los diferentes parámetros).

Para entender mejor como realizar los pasos anteriores, puede ser muy aconsejable revisar el código de otros interfaces ya implementados y así comprender bien la pauta seguida.

2.5.2 Implementar comandos.

Antes de explicar cómo implementar comandos, es necesario explicar la forma de capturar los comandos y cómo ir pidiendo los diferentes parámetros necesarios tras introducir un comando.

Como ya se ha comentado anteriormente en este capítulo, en cada menú/submenú se puede encontrar un método ("accion(comLeido:String)") al que se mandarán todas las cadenas de texto introducidas por teclado. Este método será el encargado de realizar las acciones oportunas para que el comando introducido se comporte como lo haría al introducirlo en el *router*, mostrando los mismos mensajes y errores.

Lo primero que hará el método “accion(comLeido:String)” al llegarle un comando será comprobar que dicho comando es correcto llamando al método “comprobarComando(comando:String)”. Si no es correcto se mostrará un mensaje de error diciendo que el comando no es correcto, si está incompleto se mostrará un mensaje diciendo que el comando no está completamente especificado y si es correcto se devolverá el comando completo y los parámetros (introducidos a continuación del comando, ya que los parámetros también se pueden introducir junto al comando sin que se vayan pidiendo uno a uno) si se introdujo alguno.

Una vez se obtiene el comando completo, mediante unas estructuras de selección “if/else” se ejecutarán las líneas de código adecuadas, correspondientes al comando introducido.

En ocasiones, tras introducir un comando, se piden una serie de parámetros para poder ejecutar el comando introducido con éxito (si estos parámetros no se introdujeron junto al comando). Esto implica que para cada comando se deban añadir unas líneas de código que vayan pidiendo cada uno de los parámetros necesarios (por ejemplo, tras introducir el comando “add route” se irá pidiendo la dirección IP de destino, la máscara, el gateway, etc).

En el caso en que se necesiten pedir diferentes parámetros, se asignará la cadena total del comando a la variables “obtenerComando” y el número de parámetros a pedir en al variable “numComandos”. Los parámetros se irán pidiendo en otra estructura “if/else”, y esto ocurrirá porque la variable “obtenerComando” tendrá un valor distinto de “NULL” que será la condición para saber que se esperan parámetros y no comandos. Conforme vayamos introduciendo parámetros, la variable “numComandos” irá decreciendo, de esta forma se sabrá qué parámetro será el que se tenga que pedir a continuación y cuántos parámetros quedan por pedir.

Una vez explicado el funcionamiento básico de captura de comandos, enumeraremos los pasos para implementar nuevos comandos:

1. Añadir a la variable “comandos[][]”, del menú/submenú en cuestión, el comando completo separado en cada una de sus palabras.
2. Añadir a la variable “numLetrasMinXComando[][]” el número de letras mínimo que hay que introducir para cada una de las palabras que forman el comando. La posición, en este vector, del número de letras mínimos debe de ser la misma que ocupe el comando en el vector “comandos[][]”,
3. Añadir el comando completo a la estructura de selección “if/else” del método “accion(comLeido:String)” junto con las líneas de código que se deseen ejecutar cuando se introduzca el comando.
4. En el caso de que se necesiten pedir uno o varios parámetros, tras ejecutar el comando, se colocará la cadena del comando completo en las variables “obtenerComando” y el número de parámetros a pedir en la variables “numComandos”. A continuación se implementará la estructura “if/else” para los parámetros.

Es importante destacar que en ocasiones algunos de los parámetros (no todos los necesarios) se introducirán junto al comando, por lo que, para un mismo comando, el valor de la variable “numComandos” variará dependiendo de los parámetros que ya se hayan adjuntado.

NOTA: Para comprender mejor como implementar nuevos comandos, se recomienda observar cómo se han implementado en los diferentes menús los comandos.

2.5.3 Implementar nuevos menús o submenús.

Todos los menús y submenús del *router* extenderán de la clase padre "Menú" por herencia. Gracias a esto, heredaremos todos aquellos métodos y variables comunes a todos los menús/submenús y que nos será de gran utilidad.

Para implementar nuevos menús o submenús habrá que seguir los siguientes pasos:

1. Crear una nueva clase con el nombre del nuevo menú/submenú a implementar.
2. A continuación del nombre de la clase, añadir "extends Menu" para extender de dicha clase (herencia).
3. Crear como mínimo las variables: "comandos[][]", "numLetrasMinXComando[][]" y "lista[][]". Estas variables serán para lo siguiente:
    a. "comandos[][]": Contendrá la lista de comandos que se pueden introducir en este menú. Cada comando estará separado en sus diferentes palabras, formando un array bidimensional.
    b. "numLetrasMinXComando[][]": Contendrá las letras mínimas de cada una de las palabras de los diferentes comandos. La posición debe ser la misma que la posición del comando en la variable "comandos[][]".
    c. "lista[][]": Será la lista que se muestre al introducir el comando "?" en el *router*.
4. Crear el constructor, que tendrá que tener como mínimo lo siguiente:
    a. Llamada al constructor de la clase padre, pasando la referencia de la clase principal y la referencia de las variables del *router*.
    b. Asignación de la variable "comandos[][]" a la variable "siguientes[][]".
    c. Asignación de la variable "numLetrasMinXComando[][]" a la variable "numLetrosMinimo".
    d. Asignar a la variable "path" el valor que tendrá que tener el path.
    e. Llamar al método "nuevoMenu()" que realizará el resto de acciones necesarias.
5. Todos los menús/submenús tendrá un menú/submenú antecesor menos el proceso "GESTCON" que es el menú principal. Por ello, tras implementar el menú/submenú nuevo, en el menú/submenú anterior hay que implementar el comando correspondiente que, al introducirlo, permitirá crear una instancia del nuevo menú/submenú. Esta instancia del nuevo menú/submenú se asociará a la variable "actual" de la clase principal (la que lleva el "main") que permitirá obtener el menú/submenú en el que nos encontramos a todas las clases que lo necesiten.

2.6 Recomendaciones.

Cuando se desee implementar nuevas funcionalidades al *simulador*, se recomiendo la utilización de dos programas: "Eclipse SDK" y "StarUML":

- "Eclipse SDK" es un entorno de programación para Java gratuito que, mediante plugins, contiene todas las herramientas que pueda necesitar un programador.
- "StarUML" es una herramienta gratuita que permite la creación de diagramas UML y permitirá representar los cambios que se realicen mediante diagramas UML.

Para poder modificar el código del *simulador* es muy recomendable utilizar la versión de Java proporcionada por SUN MICROSYSTEMS, ya que al utilizar otras versiones pueden ocurrir diversas incompatibilidades. Además, para poder tener todas las herramientas de desarrollo de Java, será imprescindible descargar la versión JDK.

Antes de implementar nuevas funcionalidades o modificar alguna de las funcionalidades ya implementadas, se recomienda encarecidamente entender adecuadamente el código proporcionado, especialmente el código que puede servir de ejemplo para la implementación de la funcionalidad que se desee agregar.

Los nombres de las clases, las variables y los métodos, es recomendable que sean fácilmente reconocibles, de manera que permita hacerse una idea de su finalidad a cualquier programador sólo por su nombre.

Capítulo 3
Manual del usuario

3.1 Introducción al manual de usuario.

La finalidad de este capítulo será proporcionar al usuario las instrucciones necesarias para la utilización del *simulador*, así como de las partes de las que está compuesto. Gracias a esta documentación, cualquier usuario podrá utilizar cualquiera de las facilidades proporcionadas por el entorno del *simulador*.

El manual del usuario estará dividido en los siguientes apartados:

- Requisitos del simulador.
- Ejecución y entorno de ejecución.
- Árbol de funcionalidad implementada.
- Ayudas proporcionadas.

3.2 Requisitos del simulador.

Para la ejecución del simulador se necesitará tener instalada la máquina virtual de Java y correctamente configuradas las variables de entorno para el sistema operativo. En los próximos apartados se hará una breve descripción de cómo se instala y configuran las variables de entorno tanto en Windows como en GNU/Linux.

En este documento solo se proporciona información de cómo instalar Java correctamente en Windows y Linux ya que son los más utilizados, pero el *simulador* podrá ser ejecutado desde cualquier Sistema Operativo en el que se pueda instalar Java. Tan solo habrá que ejecutar el comando “java RouterTeldat.class” en el directorio “classes” del *simulador.*

3.2.1 Instalación de Java en Windows 2000 y XP.

La ejecución de programas realizados en Java requiere la instalación de la JRE (Java Runtime Environment), el cual se puede encontrar en la dirección http://java.sun.com.

Al descargar Java, encontraremos diversas versiones, de las cuales se recomiendan las versiones JDK y JRE. La versión JDK incluye el JRE y una serie de herramientas para el desarrollo de programas en Java, mientras que la versión JRE tan solo permite la ejecución de programas en Java.

Atendiendo a lo anterior, para un usuario normal que no desee modificar el código del *simulador*, se recomienda descargar la versión JRE. Tanto esta versión como la JDK son de descarga gratuita y pertenecen a SUN MICROSYSTEMS.

Tras descargar la versión de Java escogida, se procederá a su instalación.

3.2.2 Configuración de las variables de entorno en Windows 2.000 y XP.

Una vez descargada la versión de Java elegida e instalada, hay que comprobar que las variables de entorno funcionan correctamente, y en caso de que no funcionen configurarlas.

Para comprobar que están correctamente configuradas las variables de entorno, abriremos una nueva instancia del "Símbolo del sistema", que se puede encontrar en "accesorios", y escribiremos el comando "java". Si tras escribir este comando obtenemos la lista de opciones del comando "java" significará que están correctamente configuradas las variables de entorno; si por el contrario se obtiene el mensaje *"java no se reconoce como un comando interno o externo, programa o archivo por lotes ejecutable"* querrá decir que no están correctamente configuradas las variables de entorno.

Para configurar las variables de entorno en Windows 2.000 y XP, se hará click con el botón derecho encima de "Mi PC" para seleccionar "Propiedades". Esto nos permitirá acceder a las "Propiedades del sistema". También se puede llegar hasta las "Propiedades del sistema" a través del "Panel de control", haciendo doble click sobre el icono "Sistema".

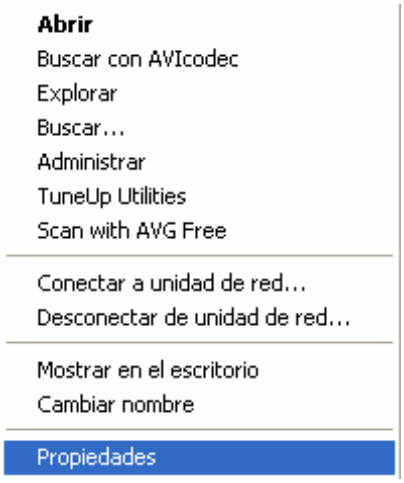


Ilustración 41: Captura botón derecho sobre "Mi PC".

En la ventana que aparece ("Propiedades del sistema") se seleccionará la pestaña de "Opciones avanzadas" y a continuación se hará click sobre el botón de "Variables de entorno".

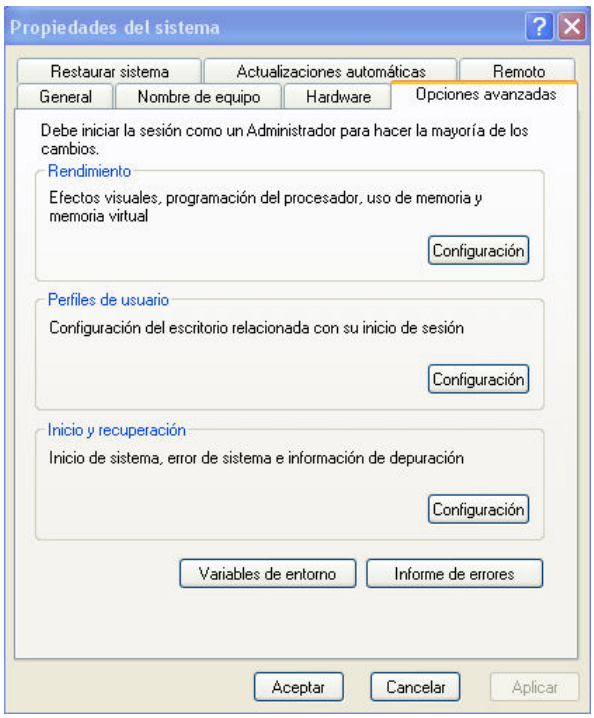


Ilustración 42: Captura de "Propiedades del sistema" en Windows.

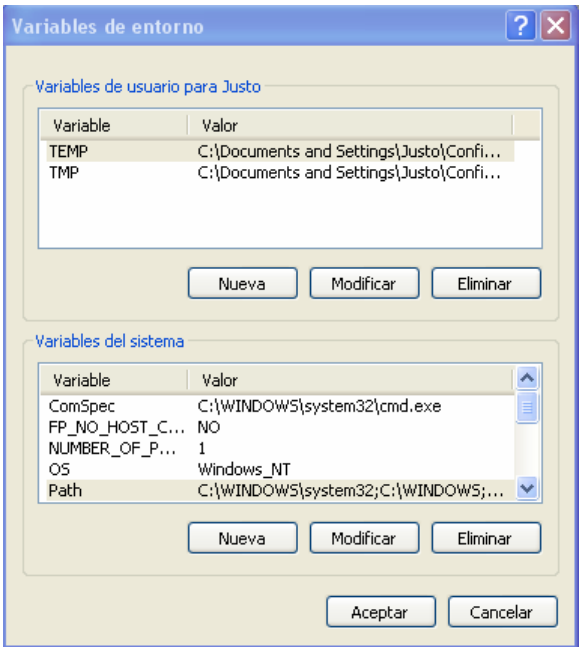


Ilustración 43: Captura de la ventana de "Variables de entorno" en Windows.

En la nueva ventana que aparece, en el apartado de “Variables del sistema” se hará doble click en “Path” y en “Valor de variable” se añadirá la ruta del directorio “bin” de la versión de Java instalada tras un “;” (por ejemplo: “C:\Archivos de programa\Java\jre1.5.0_07\bin”).

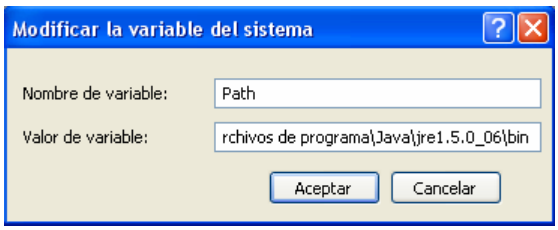


Ilustración 44: Captura de la ventana para modificar el "Path" en Windows.

3.2.3 Instalación de Java en GNU/Linux.

Al igual que para Windows, en GNU/Linux tendremos dos versiones de Java, la JDK y la JRE (Máquina Virtual de Java), las cuales se podrán encontrar en el mismo enlace de antes, http://java.sun.com.

Hay versiones de GNU/Linux que permiten instalar Java desde los CDs de instalación o desde sus repositorios. En el caso de SUSE, se podrá encontrar la JRE en los CDs de instalación y en el caso de UBUNTU se podrá encontrar la JRE añadiendo los repositorios "Universe" y "Multiverse".

Para la instalación de la JRE con SUSE se accederá al gestor de paquetes que se puede encontrar en el "yast2". Para la instalación en distribuciones basadas en Debian como UBUNTU, se utilizará el "Synaptic".

En el caso de que tengamos que descargar Java de la página oficial, se encontrarán dos tipos de archivos: un archivo ".rpm" y un archivo ".sh"; de los cuales habrá que elegir uno. La elección de uno u otro dependerá de la distribución que el usuario esté utilizando.

Una vez descargado Java, se procederá a su instalación mediante el comando "rpm –i" o el comando "sh", dependiendo del tipo de archivo descargado.

3.2.4 Configuración de las variables de entorno en GNU/Linux.

Tras la instalación de Java, habrá que comprobar que las variables del PATH en Linux están correctamente configuradas para poder ejecutar aplicaciones de Java. El tener correctamente configuradas las variables de entorno ahorrará problemas al ejecutar aplicaciones, ya que no tendremos que indicar la ruta del ejecutable "java".

Para comprobar que está correctamente configurado el PATH, abriremos una instancia de un Terminal (como puede ser "gnome-terminal" en GNOME o "konsole" en KDE) y se introducirá el comando "java". Si aparecen todas las opciones del comando significará que el PATH está correctamente configurado; si por el contrario se obtiene un mensaje de error indicando que el comando "java" no existe es porque no está correctamente configurado Java.

En el caso de que no esté correctamente configurado el PATH, se procederá a configurarlo correctamente:

1. Se creará el fichero /etc/profile.d/java.sh a fin de incluir en este una línea que añadirá la ruta de binarios de Java (/usr/java/jre1.5.0_06/bin, o lo que

corresponda según la versión del paquete) siempre antes de las rutas predeterminadas de ejecutables del sistema.

```
export PATH=/usr/java/jre1.5.0_06/bin:$PATH
JAVA_HOME="/usr/java/jre1.5.0_06/"                    export
JAVA_HOME
```

2. Haga ejecutable /etc/profile.d/java.sh:

```
chmod 755 /etc/profile.d/java.sh
```

Cuando la JRE (Máquina Virtual de Java) se encuentre en los CDs de instalación o en los repositorios de la distribución utilizada, no será necesario configurar las variables ya que automáticamente serán configuradas en la mayoría de los casos.

3.3 Ejecución y entorno del *simulador*.

Una vez cumplidos los requisitos, ya se podrá proceder a la ejecución del *simulador*. En Windows bastará con hacer doble click sobre el archivo "Simulador-Windows.bat" y en GNU/Linux bastará con introducir el comando "sh Simulador-Linux.sh" en el directorio del *simulador* desde un Terminal.

El aspecto que presentará el entorno del *simulador* será el siguiente:

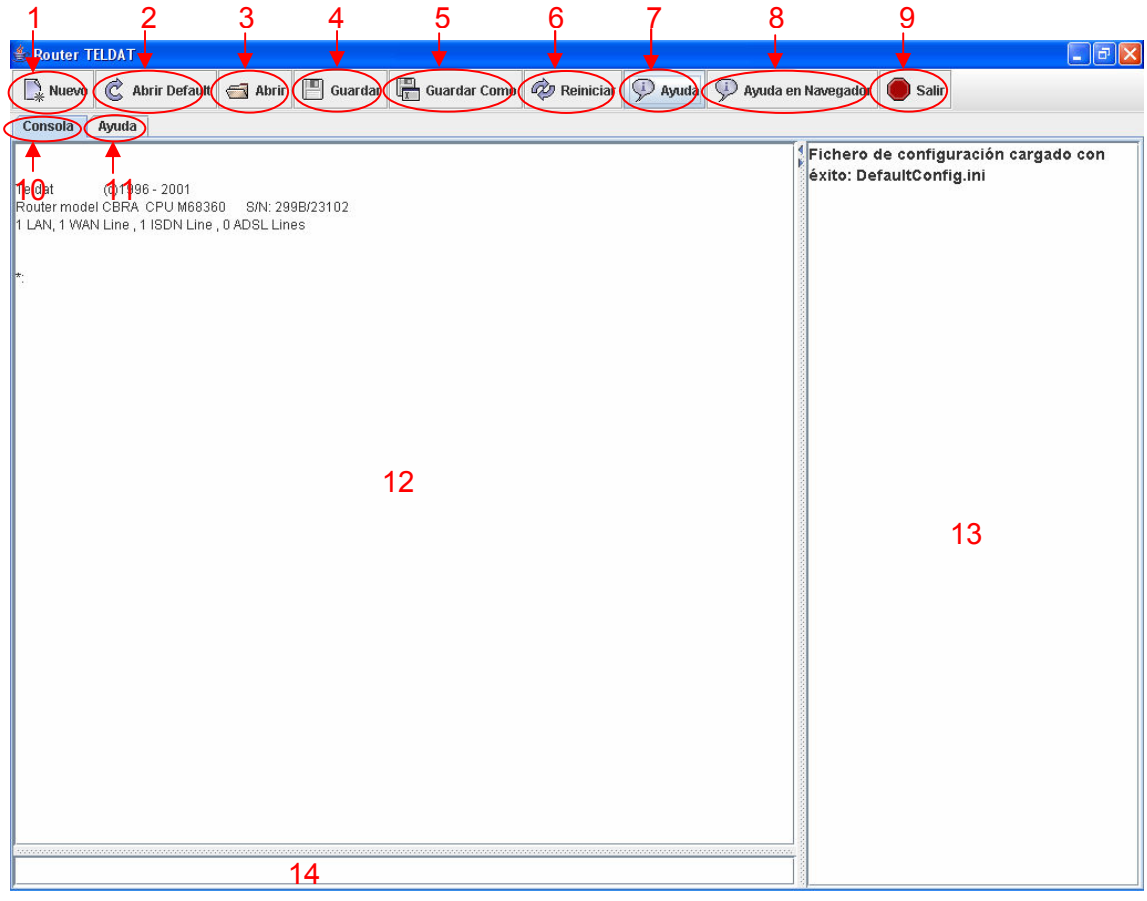


Ilustración 45: Captura del Simulador.

3.3.1 Descripción del entorno gráfico del simulador.

Observando la **ilustración 45** (Captura del Simulador), se describirán cada uno de los elementos del entorno gráfico:

1. Botón de “Nuevo”: Pulsando sobre este botón se borrará toda la configuración y todas las variables pasarán a tener su valor por defecto. Pulsar este botón será equivalente a introducir el comando “clear all” en el menú “CONFIG”.
2. Botón de “Abrir Default”: Pulsando sobre este botón se cargará el archivo de configuración por defecto (“DefaultConfig.ini”) situado en el directorio principal del *simulador*.
3. Botón de “Abrir”: Pulsando sobre este botón se abre una ventana de diálogo que permite seleccionar cualquier archivo de configuración que haya sido guardado en otras sesiones. Si el archivo abierto es incorrecto, el *simulador* cargará automáticamente los valores por defecto y en caso de guardar la configuración se guardará en el último archivo que se intentó abrir, ya que automáticamente se cambiará la ubicación del fichero de configuración a utilizar.
4. Botón de “Guardar”: Pulsando sobre este botón se guardará la configuración en el último fichero abierto. En el caso de que no se haya abierto ningún fichero diferente, la configuración se guardará en el fichero de configuración por defecto (“DefaultConfig.ini”). Pulsar este botón equivale a introducir el comando “SAVE” del proceso “CONFIG”.
5. Botón de “Guardar Como”: Pulsando sobre este botón se podrá guardar la configuración en un fichero con cualquier nombre y en cualquier ubicación. Este fichero de configuración podrá ser cargado más tarde pulsando el botón “Abrir”.
6. Botón de “Reiniciar”: Pulsando sobre este botón se podrá simular la reiniciación del *router*. Este comando equivale a introducir el comando “RESTART” en el proceso “GESTCON”.
7. Botón de “Ayuda”: Pulsando sobre este botón, la primera vez nos aparecerá una nueva pestaña que cargará un fichero HTML con documentación sobre el *router*. Al pulsar la segunda vez, la pestaña desaparecerá. Esta forma de ver el fichero de ayuda es muy lenta debido a que Java tarda mucho en procesar archivos grandes de tipo HTML, por esta razón se recomienda utilizar el botón de “Ayuda en Navegador” con el que se abrirá la ayuda en un Navegador.
8. Botón de “Ayuda en Navegador”: Pulsando sobre este botón se abrirá el fichero de ayuda en el navegador. El navegador utilizado para Windows será el Internet Explorer, mientras que el navegador utilizado para GNU/Linux será Firefox.
9. Botón de “Salir”: Pulsando sobre este botón se saldrá del programa. Pulsar este botón será equivalente a cerrar la ventana del *simulador*.
10. Pestaña de “Consola”: Pestaña principal en la que se irán introduciendo los comandos y en la que se irán mostrando los diferentes mensajes que devolvería el *router* en la realidad.
11. Pestaña de “Ayuda”: Pestaña en la que aparecerá la ayuda proporcionada sobre el *router*.

12. Campo de texto para mensajes de salida: En este espacio irán apareciendo todos los mensajes que irá devolviendo el *simulador* como respuesta a los comandos introducidos.
13. Campo de texto para notas de ayuda: En este espacio irán apareciendo diferentes mensajes de ayuda a modo de consejo o recordatorio para ayudar al usuario en el aprendizaje.
14. Campo de texto para la introducción de comandos: En este espacio se irán introduciendo los diferentes comandos para la configuración simulada del *router*. Una vez escrito el comando, se deberá pulsar la tecla "Intro" para que sea introducido.

Hay que añadir que cada botón mostrará una pequeña ayuda al dejar el puntero sobre el botón un breve tiempo.

3.4 Árbol de funcionalidad implementada.

Mediante un esquema se mostrarán las funcionalidades implementadas del *router* en el *simulador*. La disposición del esquema está basada en la ubicación del comando (el menú en el que se encuentra). Los comandos y menús/submenús que no aparezcan en el esquema no están implementados.

1. GESTCON
2. CONFIG
    i. ADD DEVICE FR-DIAL
    ii. ADD DEVICE FR-ISDN
    iii. ADD DEVICE PPP-DIAL
    iv. ADD DEVICE X25-DIAL
    v. ADD DEVICE ATPPP-DIAL
    vi. ADD DEVICE MPPP
    vii. ADD DEVICE TNIP
    viii. ADD DEVICE 270
    ix. CLEAR ALL
    x. CLEAR DEVICE
    xi. CLEAR IP
    xii. DELETE DEVICE
    xiii. LIST DEVICE
    xiv. NETWORK
        1. AT COM
            a. DISABLE AUTO-ANSWER
            b. DISABLE RING-PATTERN
            c. ENABLE AUTO-ANSWER
            d. ENABLE RING-PATTERN
            e. LIST
            f. SET CONNECTION
            g. SET CTS-CONTROL
            h. SET DIAL
            i. SET DCD-CONTROL
            j. SET DSR-CONTROL
            k. SET DTR-CONTROL
            l. SET FLOW-CONTROL
            m. SET NUMBER-RINGS
            n. SET T1-RINGS
            o. SET T2-SILENCE

p. SET V42-CONTROL

2. B CHANNEL: PPP
    a. DISABLE ACCESS
    b. DISABLE INCOMING
    c. DISABLE OUTGOING
    d. ENCAPSULATOR
        i. ADD USERS
        ii. DISABLE AUTHENTICATION
        iii. DISABLE CALLBACK
        iv. DISABLE CRTP
        v. DISABLE NAT
        vi. DISABLE MPPP
        vii. DISABLE RAIDUS
        viii. DISABLE RIP-NO-DIAL
        ix. DELETE USERS
        x. ENABLE AUTHENTICATION CHAP
        xi. ENABLE AUTHENTICATION PAP
        xii. ENABLE CALLBACK
        xiii. ENABLE CRTP
        xiv. ENABLE NAT
        xv. ENABLE MPPP
        xvi. ENABLE RADIUS
        xvii. ENABLE RIP-NO-DIAL
        xviii. LIST ALL
        xix. LIST LINE
        xx. LIST LCP
        xxi. LIST NCP
        xxii. LIST IPCP
        xxiii. LIST AUTHENTICATION
        xxiv. LIST FACILITY
        xxv. LIST USERS
        xxvi. LIST INTERVAL-OF-CONNECTION
        xxvii. LIST CCP
        xxviii. SET AUTHENTICATION
        xxix. SET CCP CHECK
        xxx. SET CCP HISTORY
        xxxi. SET CCP PROCESS
        xxxii. SET CCP TYPE
        xxxiii. SET INTERVAL-OF-CONNECTION
        xxxiv. SET IPCP
        xxxv. SET LCP OPTIONS
        xxxvi. SET LCP PARAMETERS
        xxxvii. SET LINE ENCODING NRZ
        xxxviii. SET LINE ENCODING NRZI
        xxxix. SET LINE IDLE MARK
        xl. SET LINE FRAME-SIZE
        xli. SET LINE LINE-SPEED
        xlii. SET LINE TRANSMIT-DELAY
        xliii. SET NCP
    e. ENABLE ACCESS
    f. ENABLE INCOMING
    g. ENABLE OUTGOING
    h. LIST
    i. SET BASE-INTERFACE
    j. SET DESTINATION-ADDRESS

   k. SET INACTIVE-TIME
   l. SET PERMITTED-CALLER
   m. SET NAME-CIRCUIT
3. ISDN
   a. LIST
   b. SET CONNECTION
   c. SET LOCAL
   d. SET MAXIMUM
4. MULTILINK PPP
   a. SET DIRECTION
   b. SET FRAGMENTATION VOICE
   c. SET INTERVAL ACTIVATION
   d. SET INTERVAL DESACTIVATION
   e. SET PRE-EMPTION
   f. SET THRESHOLD ACTIVATION
   g. SET THRESHOLD DESACTIVATION
5. PPP AT COM
   a. ENCAPSULATOR
      i. ADD USERS
      ii. DISABLE AUTHENTICATION
      iii. DISABLE CALLBACK
      iv. DISABLE CRTP
      v. DISABLE NAT
      vi. DISABLE MPPP
      vii. DISABLE RAIDUS
      viii. DISABLE RIP-NO-DIAL
      ix. DELETE USERS
      x. ENABLE AUTHENTICATION CHAP
      xi. ENABLE AUTHENTICATION PAP
      xii. ENABLE CALLBACK
      xiii. ENABLE CRTP
      xiv. ENABLE NAT
      xv. ENABLE MPPP
      xvi. ENABLE RADIUS
      xvii. ENABLE RIP-NO-DIAL
      xviii. LIST ALL
      xix. LIST LINE
      xx. LIST LCP
      xxi. LIST NCP
      xxii. LIST IPCP
      xxiii. LIST AUTHENTICATION
      xxiv. LIST FACILITY
      xxv. LIST USERS
      xxvi. LIST INTERVAL-OF-CONNECTION
      xxvii. LIST CCP
      xxviii. SET AUTHENTICATION
      xxix. SET CCP CHECK
      xxx. SET CCP HISTORY
      xxxi. SET CCP PROCESS
      xxxii. SET CCP TYPE
      xxxiii. SET INTERVAL-OF-CONNECTION
      xxxiv. SET IPCP
      xxxv. SET LCP OPTIONS
      xxxvi. SET LCP PARAMETERS
      xxxvii. SET LINE ENCODING NRZ

xxxviii. SET LINE ENCODING NRZI
xxxix. SET LINE IDLE MARK
xl. SET LINE FRAME-SIZE
xli. SET LINE LINE-SPEED
xlii. SET LINE TRANSMIT-DELAY
xliii. SET NCP
b. LIST
c. SET DESTINATION-ADDRESS
d. SET INACTIVE-TIME
6. QUICC ETHERNET
a. IP IEEE-802.3
b. IP ETHERNET
c. LIST
d. MAC

xv. PROTOCOL IP
1. ADD ACCESS-CONTROL
2. ADD ADDRESS
3. ADD AGGREGATION-ROUTE
4. ADD FILTER
5. ADD ROUTE
6. CHANGE ADDRESS
7. CHANGE ROUTE
8. DELETE ACCESS-CONTROL
9. DELETE ADDRESS
10. DELETE ROUTE
11. LIST ACCESS-CONTROLS
12. LIST ADDRESSES.
13. LIST ROUTES
14. SET ACCESS-CONTROL OFF
15. SET ACCESS-CONTROL ON
16. SET DEFAULT NETWORK-GATEWAY
17. SET DEFAULT SUBNET-GATEWAY

xvi. SAVE

3.5 Ayudas proporcionadas.

En el directorio principal del *simulador* se puede encontrar un fichero llamado “ayuda.html”. Este fichero contiene información sobre conocimientos básicos necesarios y sobre cada uno de los comandos implementados en el *simulador*. Esta ayuda explica detalladamente los procedimientos a seguir para realizar las diferentes configuraciones que permite el *simulador*.

Lo primero que se podrá ver en la ayuda será el árbol de la funcionalidad implementada. Tras el árbol aparecerá la descripción del proceso principal (GESTCON), desde el cual mediante hiperenlaces se podrá acceder a los diferentes apartados de la documentación.

La estructura de la ayuda en HTML se basa en ir recorriendo cada una de las ramas del árbol de la funcionalidad implementada. Dentro de un menú/submenú, la ayuda mostrará la sintaxis adecuada de cada comando y algunos ejemplos de utilización. Además, las diferentes secciones que puedan estar relaciones y que se encuentren en sitios diferentes dentro del documento, estarán enlazadas mediante hiperenlaces.

Capítulo 4
Conclusiones y líneas futuras

4.1 Conclusiones.

4.1.1 Conclusiones sobre los *routers* CBRA.

Los *routers* CBRA destacan por su facilidad de configuración. Esta facilidad de configuración los convierte en una buena herramienta para la docencia.

Los diferentes menús y las diferentes capacidades están ordenadas de manera "lógica" en una especie de árbol de menús, lo que facilita la búsqueda. Dentro de cada menú/submenú se encuentran unos comandos básicos muy claros, cuyo nombre permite fácilmente asociar el comando a la acción que se desea hacer.

El punto negativo que puede tener esta familia de *routers* es que no proporcionan muchos servicios. Esta familia de *routers* sólo proporciona los servicios mínimos de un *router* de sus características, comparándolos con, por ejemplo, los *routers* CISCO.

A parte de lo comentado antes, la experiencia con la familia de *routers* CBRA ha sido positiva y el desarrollo de un *simulador* ha permitido conocer más a fondo la forma de comportarse un *router* internamente.

Durante la implementación del *simulador* fue necesario buscar y entender en qué se basaba el *router* para simular las decisiones que tomaba, lo cual supuso grandes dificultades ya que no se podía tener acceso al código del software del *router*. El llegar a comprender mejor el funcionamiento interno permitió entender más sobre el funcionamiento de un *router*, ya que a menudo ese funcionamiento interno es transparente al usuario.

4.1.2 Conclusiones de experiencia con Java.

La implementación de un *simulador* de esta complejidad en Java ha permitido consolidar las bases de programación, adquiridas durante las diferentes asignaturas de programación de la carrera, y aprender cosas nuevas sobre un lenguaje de programación que cada día es más popular y más utilizado.

Durante la implementación, Java ha demostrado ser uno de los lenguajes de programación más sencillos y potentes. Gracias a su compilador, muchos de los errores, que en otros lenguajes serían errores de ejecución, se han podido solucionar.

Como alumno, han sido varios los lenguajes de programación utilizados a lo largo de la carrera y no hace falta mencionar los problemas que pueden dar otros lenguajes de programación como C o C++.

Otros temas importantes en Java son la gran cantidad de librerías que tiene y la documentación tan completa que se puede encontrar. Java contiene un elevado número de librerías que permiten realizar casi cualquier cosa y todas estas librerías están correctamente documentadas en su API.

4.2 Líneas futuras.

El proceso de configuración de un *router* es una de las tareas típicas de un Ingeniero Técnico de Telecomunicaciones, especialidad Telemática. A menudo, un alumno no dispone de los recursos para poder practicar con *routers* de verdad en horario no lectivo. Este hecho hace que sea importante tener un medio por el que un alumno pudiese practicar con un *router* en horario no lectivo, como por medio de un *simulador*.

Atendiendo a lo explicado, sería importante que existieran *simuladores* para todos los *routers* utilizados en docencia. Esto facilitaría el aprendizaje de los alumnos y permitiría, además, su utilización en sesiones de prácticas.

El código fuente del *simulador* implementado para este Proyecto Fin de Carrera podrá ser utilizado para implementar nuevas funcionalidades y crear futuros *simuladores* de *routers* con una estructura similar a los *routers* CBRA de Teldat.

Como futuro software a implementar, también sería muy interesante que el *simulador* implementado pasase a ser un *emulador*. Un *emulador* permitiría utilizar un PC normal como *router*, permitiendo durante las prácticas utilizar PCS normales. Con esto no sería necesario tener un *router* por cada banco de trabajo cuando las prácticas no fuesen de una cierta complejidad y necesitasen de un *router* real obligatoriamente.

Bibliografía

BIBLIOGRAFÍA DE JAVA

[1] Página oficial de Java
http://java.sun.com

[2] JavaTM 2 Platform, Standard Edition, v 1.5 API Specification
http://java.sun.com/j2se/1.5.0/docs/api/index.html

[3] Introducción a Java
http://www.programacion.com/java/tutorial/intjava/

[4] I/O: Leer y Escribir en Java
http://www.programacion.com/java/tutorial/io/

[5] Manejo de errores usando Excepciones
http://www.programacion.com/java/tutorial/excepciones/

[6] Tutorial de Java
http://java.sun.com/docs/books/tutorial/index.html

[7] Eclipse
http://www.eclipse.org/

BIBLIOGRAFÍA DE UML

[8] Introducción a UML
http://www.programacion.com/tutorial/uml/

[9] Desarrollo Orientado a Objetos con UML
http://www.clikear.com/manuales/uml/

[10] Tutorial UML 2.0
http://www.sparxsystems.com.au/UML_Tutorial.htm

[11] StarUML

http://staruml.sourceforge.net/en/

BIBLIOGRAFÍA DEL ROUTER TELDAT

[12] Manuales de configuración

En el directorio “Documentación -> Documentación Router” adjunto en el CD del proyecto. (La página de Teldat requiere registro previo)