**UNIVERSIDAD NACIONAL AUTÓNOMA DE NICARAGUA**

**UNAN – LEÓN**

**Facultad De Ciencias y Tecnología**

**Departamento de Computación**

# Creación de un front-end para VirtualBox que permita la gestión de escenarios de redes virtuales internas desarrollado en Java con las librerías JUNG Y JespXML

**Tesis para optar al título de**

**INGENIERO EN SISTEMAS DE INFORMACIÓN**

**Presentado por:**

Br. Herwin Alexander Méndez Mendoza

**Tutor:**

Msc. Denis Leopoldo Espinoza Hernández.

# DEDICATORIA

Dedico este trabajo principalmente a DIOS, por haberme dado el don de la vida y permitirme el haber llegado hasta este momento tan importante de mi formación profesional.

De igual forma, dedico esta tesis en memoria de mi padre: Santos Domingo Méndez Mitchell (Q.E.P.D), quien con sus consejos supo formarme con buenos sentimientos, hábitos y valores, los cuales me han ayudado a salir adelante en los momentos más difíciles.

# AGRADECIMIENTO

En primer lugar doy infinitamente gracias a DIOS, por haberme dado fuerza, sabiduría y valor para culminar esta etapa de mi vida.

Agradezco también la confianza y el apoyo brindado por parte de mi madre, que sin duda alguna en el trayecto de mi vida me ha demostrado su amor, corrigiendo mis faltas y celebrando mis triunfos.

A mis hermanos, que con sus consejos me han ayudado a afrontar los retos que se me han presentado a lo largo de la vida.

Agradezco especialmente a mi novia Gemma Milagro Espinoza Toruño, por su apoyo incondicional, su amor y comprensión han sido parte fundamental en mi vida.

Al Msc. Denis Leopoldo Espinoza Hernández por toda la colaboración brindada durante la elaboración de este proyecto.

Finalmente a Jesús Trinidad Herrera Centeno por cada una de sus valiosas aportaciones que hicieron posible este proyecto y por la gran calidad humana que me ha demostrado con su amistad.

# Tabla de contenido

# 1. INTRODUCCIÓN

En la actualidad los estudiantes de las carreras de ingeniería en sistema de información e ingeniería en telemática de la Universidad Nacional Autónoma de Nicaragua "UNAN-LEÓN", han tenido dificultades en la elaboración de las prácticas de la materia "administración de servicios de redes".

Las prácticas plantean la creación de escenarios de redes y configuración de los equipos virtuales con la herramienta virtualizadora "VirtualBox".Al realizar las interconexiones entre las máquinas virtuales el docente y/o estudiante no puede mostrar de manera gráfica el escenario de red creado, generando sobreesfuerzo ya que se diseña la topología de red con alguna herramienta como por ejemplo Cisco Packet Tracer o en ocasiones en hoja de papel.

Debido a esta problemática es de vital importancia la existencia de una herramienta que permita la administración de escenarios de red utilizando máquinas virtuales en VirtualBox, visualizando el diagrama de red formado por las máquinas virtuales seleccionadas por el usuario así como la manipulación de los adaptadores de red y la creación, clonación y eliminación de dichas máquinas.

La aplicación está desarrollada en el lenguaje Java que permite sea portable y multiplataforma, ya que el docente y/o estudiante puede utilizarla en los sistemas operativos windows o linux. La librería utilizada para la visualización del escenario de red se llama JUNG (Java Universal Network Graph), esta es una biblioteca de código abierto que permite gestionar grafos. Además otra librería incorporada es JespXML, la cual sirve de intermediario entre la aplicación y VirtualBox, debido a que el fichero de configuración de las máquinas virtuales con extensión .vbox, está escrito en el lenguaje XML. Ésta librería permite la lectura y escritura de archivos escritos en un lenguaje de etiquetado extensible.

## 1.1 Antecedentes

En la actualidad existe software con los cuales se pueden construir escenarios de red, tal es el caso de la NETGUI con la cual se puede generar escenarios de red en base de selección/ arrastre y suelte de objetos teniendo el inconveniente que no muestra un escenario de red a partir de la creación de máquinas virtuales en un virtualizador y solo puede ser ejecutada bajo el sistema operativo Linux.

También es el caso de la aplicación GNS3, la diferencia entre NETGUI y GNS3 es que NETGUI posee integrado los servidores y switches en la misma aplicación, GNS3 ocupa imágenes ISO para realizar la simulación de dichos servidores y switches siendo esta una de las más notables diferencias entre estos generadores de escenarios de red, remarcando el hecho que este último puede ser ejecutado en linux y windows.

Existen otro software que se utilizan para la creación de escenarios de red tales son el caso de:

1. Cisco Packet Tracer
2. Netsim
3. Netsimk
4. WebNms Simulation Toolkit
5. Shunra NV Desktop
6. Jimsim

Corriendo bajo un sistema operativos en específico.

## 1.2 Planteamiento del problema

Debido a la gran demanda de realizar escenarios de redes de ordenadores virtuales sin necesidad de disponer de dispositivos de comunicaciones ni de ordenadores reales se hizo de vital importancia la utilización de una herramienta de virtualización la cual soportara varios sistemas operativos y que fuera multiplataforma.

VirtualBox es el principal software que cumple con los requisitos descritos y por ende se pueden crear las máquinas virtuales realizando las conexiones de los adaptadores en dichas redes internas pero no logrando aún visualizar las interconexiones de estas.

La problemática surge en las materias:

1. Administración de servicios de red I. (SEMESTRE VIII)
2. Administración de servicios de red II. (SEMESTRE IX)

De las carreras correspondientes que ofrece la UNAN-LEÓN:

1. Ingeniería en sistemas de información
2. Ingeniería en telemática

Dando lugar a las siguientes interrogantes:

1. ¿Cómo visualizar dicha interconexión de máquinas virtuales?
2. ¿puedo establecer una topología de red partiendo de las máquinas virtuales creadas?
3. ¿Solo desde VirtualBox puedo interactuar con las máquinas virtuales?

## 1.3 Justificación

Dando solución a la problemática se hace necesaria la creación de una aplicación multiplataforma y con una interfaz amigable al usuario, desarrollada en lenguaje Java, que permite la administración de escenarios de red utilizando máquinas virtuales en VirtualBox; beneficiando así, mayormente a los estudiantes y docentes que requieran gestionar los escenarios de red modificando las interfaces a fin de visualizar su interconexión de red.

### 1.3.1 Originalidad

Anteriormente no se ha desarrollado ningún software que permita gestionar escenarios de redes partiendo de máquinas virtuales creadas con una herramienta virtualizadora siendo esta aplicación la primera en su género.

### 1.3.2 Alcance

El alcance del proyecto está dedicado al ámbito educacional específicamente para docentes y estudiantes de las carreras de ingeniería en sistema de información e ingeniería en telemática que necesiten utilizar VirtualBox para la creación de máquinas virtuales generando así escenarios de red.

***Esta aplicación permitirá a sus usuarios:***

- Diseñar escenarios de red.
- Crear, clonar y eliminar máquinas virtuales
- Configurar los adaptadores de cada máquina virtual.
- Maximizar y minimizar el escenario de red.
- Agregar elementos (pc, switch) a un escenario de red previamente diseñado.
- Ejecutar, salvar, detener máquinas virtuales.
- Dibujar y agrupar zonas de redes con figuras geométricas coloreadas (rectángulo, elipse, rectángulo con bordes redondeado).
- Guardar como imagen el escenario de red diseñado.
- Generar reporte PDF de las máquinas virtuales contenidas en la topología.

### 1.3.3 Producto

El resultado de este proyecto es una aplicación sencilla al usuario, que incorpora las funcionalidades necesarias para el gestionamiento de escenarios de red. Además de permitirle un mayor entendimiento de la interconexión que estas poseen entre sí, pero siempre tomando en cuenta los siguientes aspectos:

**Escalabilidad**: Dejar siempre abierta la posibilidad de agregar más funcionalidad sin afectar a lo que ya estará desarrollado.

**Estable**: Porque el usuario debe poder visualizar y manipular las interfaces de red sin alterar el funcionamiento propio de la máquina virtual.

### 1.3.4 Impacto

El desarrollo de esta aplicación además de proporcionar el diseño de escenarios de red a partir de las máquinas virtuales creadas con VirtualBox, permitirá al docente y/o estudiante configurar, guardar imagen del escenario de red, ejecutar máquinas virtuales y obtener información en tiempo real. Permitiéndole así una mayor utilidad con respecto a la gestión de los escenarios de redes y evitara la creación de escenarios de red de una manera arcaica, es decir, a partir de las máquinas virtuales creadas en VirtualBox, diseñar la topología en una hoja de papel, ya es cosa del pasado.

## 1.4 Objetivos

### 1.4.1 Objetivos Generales

Desarrollar una aplicación que permita la administración de escenarios de red utilizando máquinas virtuales en VirtualBox.

### 1.4.2 Objetivos Específicos

1. Visualizar el diagrama de red formado por las máquinas virtuales seleccionadas por el usuario.

2. Editar la configuración de los adaptadores de red de las máquinas virtuales para crear la topología deseada por el usuario.

3. Crear, clonar y eliminar máquinas virtuales.

4. Permitir generar reporte en formato PDF de las máquinas virtuales contenidas en la topología.

# 2. MARCO TEÓRICO

La Virtualización es una tecnología de software acreditada que rápidamente está transformando el panorama de las tecnologías de la Información TI (Information Technology) y cambiando fundamentalmente la manera en que las personas utilizan las computadoras.

El poderoso hardware de las computadoras x86 de hoy en día fue originalmente diseñado para ejecutar un solo sistema operativo y una sola aplicación, pero la virtualización rompe esa limitación, haciendo posible que múltiples sistemas operativos y múltiples aplicaciones se ejecuten en la misma computadora y al mismo tiempo, incrementando la utilización y flexibilidad del hardware.

## 2.1 Tipos de virtualización

Entre los métodos más populares de virtualización de servidores podemos citar: Emulación, Virtualización completa, Paravirtualización y Virtualización a nivel de sistema operativo.

### 2.1.1 Emulación

La emulación se basa en crear máquinas virtuales que emulan el hardware de una o varias plataformas de hardware distintas. Este tipo de virtualización es la más costosa y la menos eficiente, ya que obliga a simular completamente el comportamiento de la plataforma de hardware a emular e implica también que cada instrucción que se ejecute en estas plataformas sea traducida al hardware real.

Sin embargo la emulación tiene características interesantes: como permitir que un sistema operativo creado para una arquitectura en particular sea ejecutado sobre una arquitectura diferente sin tener que modificarlo, o cuando se desarrolla firmware para hardware, se puede comenzar el desarrollo aún sin tener el dispositivo de hardware real.

### 2.1.2 Virtualización completa

La virtualización completa (full virtualization), también llamada virtualización nativa, está diseñada para proveer una abstracción completa del hardware subyacente y crear un sistema virtual completo (hypervisor) en el cual el SO huésped puede ser ejecutado.

Algunas instrucciones protegidas deben capturarse y manejarse dentro del hypervisor ya que el hardware subyacente no es propiedad de un sistema operativo sino que es compartido a través del hypervisor. No se requiere ninguna modificación del SO huésped; el SO huésped no es consciente de que se está ejecutando en un entorno virtualizador por lo que tienen la capacidad de comportarse como si el hardware emulado fuera su propio hardware "real". Esto puede ser ventajoso, ya que permite una completa independencia entre el software y el hardware, pero puede llegar a penalizar el rendimiento.

Para que el SO huésped funcione correctamente, el software virtualizador deberá emular el hardware a la perfección, haciendo creer al SO huésped que tiene acceso directo al equipo. Cada vez que el huésped quiera hacer uso de los recursos del hardware real (acceso al disco, memoria, vídeo) lo hará por medio de la aplicación emuladora.

La virtualización completa es una técnica que asegura que el SO invitado funcione de manera casi transparente, sin ninguna modificación, pero tiene un precio muy alto a pagar y es el nivel de rendimiento. En líneas generales se habla de un 20 a 30% de pérdida de rendimiento debido a esta técnica. Lo que significa que nuestro SO invitado va a estar utilizando un hardware menos potente de lo que realmente es el hardware.

### 2.1.3 Paravirtualización

La paravirtualización es una técnica para incrementar el rendimiento de los sistemas virtualizados reduciendo la cantidad de recursos de hardware que el monitor de máquina virtual (VMM por sus siglas en inglés) debe emular mediante software. La emulación tradicional generalmente involucra traducción binaria, en la cual un proceso dentro del VMM atrapa las llamadas de hardware hechas por el sistema operativo y las traduce, haciéndolas compatibles con el sistema operativo anfitrión. Dicha traducción, sin embargo, incrementa la capacidad de procesamiento y por ende reduce el rendimiento del sistema.

La paravirtualización elimina la necesidad de la traducción binaria, ya que los sistemas operativos huéspedes son modificados para que éstos sepan que se están ejecutando sobre una máquina virtual; esto ayuda al VMM a eludir instrucciones de procesador difíciles de virtualizar, reemplazándolas con una llamada a un proceso que proporciona esa funcionalidad. El resultado es un rendimiento mejor que el de la virtualización completa.

### 2.1.4 Virtualización a nivel de sistema operativo.

En la virtualización a nivel de sistema operativo, en lugar de que el hardware engañe al sistema operativo, como se hace en los modelos de virtualización de hardware, nos vamos un nivel más arriba y ahora es el sistema operativo que engaña a las aplicaciones. Básicamente se tiene una sola instancia de hardware y una sola imagen base de sistema operativo que se puede multiinstanciar creando así particiones aisladas o entornos virtuales (VEs) donde van a correr las aplicaciones.

El modelo de virtualización a nivel de sistema operativo es mejorado para conseguir un mejor rendimiento, administración y eficiencia. En la base reside un sistema operativo estándar. Luego se encuentra la capa de virtualización con un sistema de archivos y una capa de abstracción de los servicios del kernel que garantiza el aislamiento y la seguridad de los recursos entre los diferentes entornos virtuales. La capa de virtualización hace que cada máquina virtual se vea como un servidor independiente. Finalmente, los entornos virtuales albergan las aplicaciones y la carga de trabajo.

## 2.1.5 Estructura de XML

XML son las siglas del lenguaje de etiquetado extensible. La expresión se forma a partir del acrónimo de la expresión inglesa eXtensible Markup Language. Se trata también de un lenguaje estándar que posee una recomendación del World Wide Web Consortium: Extensible Markup Languajes (XML) (http://www.w3.org/TR/REC-xml/). Con la palabra "Extensible" se alude a la no limitación en el número de etiquetas, ya que permite crear aquellas que sean necesarias.

***Los principales usos de XML son los siguientes:***

- XML aplicado a los **sitios web**: permite separar contenido y presentación, y que los mismos datos se puedan mostrar de varias formas distintas sin demasiado esfuerzo.
- XML para la **comunicación** entre aplicaciones: representación de los datos muy simple, fácil de transmitir por la red, estándar. En los últimos tiempos este uso se está haciendo muy popular con el surgimiento de los *servicios web*.
- XML para la **configuración** de programas: representación de los datos simple y estándar, en contraposición con los crípticos formatos propietarios.

Los documentos XML pueden ser de 2 tipos: documentos bien formados y documentos válidos. Un documento está bien formado si, considerado en su conjunto, encaja con la especificación XML de producción. Un documento bien formado es, además, válido, si tiene DTD o esquema. Un archivo XML válido comienza con una DTD o Declaración de Tipo de Documento:

```
<?xml version="1.0"?>

<!DOCTYPE noticia SYSTEM "http://www.hipertexto.info/ad.dtd">

<noticia>

<titulo>...<foto/>...</titulo>

<texto>...</texto>

</noticia>
```

### 2.1.6 Terminología XML

- **XML:** Es el metalenguaje sobre el que se construye todo. Por sí mismo tiene un valor limitado, pero cuando se habla de XML, normalmente no se refiere sólo al lenguaje XML, sino también a todos los lenguajes, especificaciones y herramientas relacionadas.
- **Esquema XML:** Otro lenguaje para restringir los documentos XML, que soluciona diversos problemas de las DTDs y que tiene mayor potencia que éstas ya que permite especificar el contenido de los documentos en función del tipo de datos empleado.
- **XSLT:** Lenguaje para transformar documentos de un formato XML a otro formato (por ejemplo, a HTML, otro vocabulario XML, texto plano, PDF, etc.)
- **DOM:** Otro tipo de API (*Application Programming Interface*) que proporciona una representación de los documentos XML en forma de árbol, permitiendo el recorrido y manipulación de los datos.
- **XHTML:** Reformulación del HTML como formato de datos XML.
- **SOAP:** Lenguaje que especifica la forma de enviar contenido XML a través de Internet.
- **Servicios web:** Aplicaciones que se publican en la web y pueden ser accedidas de forma estándar desde cualquier lugar de Internet.

## 2.2 Tipos de conexiones de red en VirtualBox

### 2.2.1 Modo bridge

Modo bridge o puente es la configuración por defecto cuando creamos una máquina virtual, ya que es la forma más sencilla de otorgar acceso a la red a una máquina virtual.

Cuando está en modo bridge, la red local es extendida desde el equipo anfitrión hacia la máquina virtual. Aunque el equipo se conecte a la red local usando el hardware del ordenador físico, la máquina virtual será totalmente independiente de la red. De esta forma, una máquina virtual puede usar todos los servicios disponibles en la red a la que está conectada, incluidos los servidores de archivos, las impresoras y las puertas de enlace. Del mismo modo, todo equipo físico u otra máquina virtual que esté conectada de la misma forma, podrá usar los recursos de la máquina virtual como si fuera un equipo físico en la misma red.

De esta forma si el equipo físico está configurado para recibir una dirección IP por un servidor DHCP, la máquina virtual recibirá una IP del mismo servidor DHCP.

### 2.2.2 Modo NAT

El modo NAT es un modo de conexión fácil de utilizar pero algo complicado de entender.Para situarse, NAT (Network Address Translation) fue pensado para solucionar el problema de la escasez de direcciones IP de forma que redes de ordenadores utilicen un rango de direcciones especiales (IP privadas) y se conecten a internet usando una única dirección IP (IP pública), de esta forma varios equipos se conectan a internet con una única IP pública.

En las máquinas virtuales lo que sucede es que ésta recibirá una dirección IP de un servidor DHCP virtual, sin embargo el que pide la IP será el firewall dentro de la aplicación de virtualización, que sustituye a la máquina virtual. Así, el que se encarga de comunicarse con la red fuera del equipo será el firewall, no la máquina virtual.

### 2.2.3 Modo red interna

El modo red interna es una forma de conectar varias máquinas virtuales entre ellas creando una red privada, de esta forma, las máquinas virtuales no podrán comunicarse con el equipo anfitrión ni viceversa.

Con esta opción podemos crear más de una red interna de forma sencilla, de esta forma se puede trabajar con varias redes internas independientes de forma muy sencilla.

### 2.2.4 Modo host-only

El modo host-only, como su propio nombre indica, solo se conecta con el host anfitrión.

Cuando está en modo host-only, la máquina virtual está totalmente aislada de la red de área local ya que la red de la máquina virtual está dentro del propio equipo y es invisible e inaccesible para cualquier equipo de la red del equipo.

### 2.2.5 Modo no conectado

Esta opción indica que hay una tarjeta de red instalada pero no está conectada a ningún otro lugar. Esta configuración se utiliza para que no se pierda la configuración, ya que aunque este en modo no conectado, la tarjeta de red guarda la configuración especificada.

## 2.3 JUNG

El JUNG (Java Universal Network Graph) es un software libre de código abierto; biblioteca que proporciona un lenguaje común y extensible para la manipulación, análisis, y la visualización de datos que pueden ser representados como un grafo o red.

Está escrito en el lenguaje de programación Java, permitiendo a las aplicaciones basadas en JUNG hacer uso de amplias capacidades incorporadas de **JAVA Aplicattion Programming Interface** (API), así como las de otras bibliotecas existentes Java de terceros. Se describe el diseño, y algunos detalles de la aplicación, de la arquitectura JUNG, y proporcionar ilustrativos ejemplos de su uso.

Un conjunto de datos de red normalmente se compone de un conjunto de entidades y relaciones conocidas entre éstas entidades. Por ejemplo, un conjunto de datos de red social podría consistir en una lista de personas y una lista de las relaciones binarias pares que indican los pares de individuos que se sabe que son amigos.

A menudo es conveniente representar formalmente datos como un gráfico con vértices representando entidades y bordes que representan sus relaciones. Ejemplos de redes de amplio interés incluir redes bibliográficas de citas (documentos que citan otros documentos), redes biológicas (por ejemplo, proteínas y sus interacciones), redes de telecomunicaciones y la web.

Conjuntos de datos de la red también pueden incluir información adicional sobre ambas entidades y la relación.

Una red de co-autoría, por ejemplo, se podría definir en autores y co-autor relaciones. Información adicional sobre cada autor puede incluir el nombre del autor, afiliaciones institucionales, y áreas de interés. Del mismo modo, los documentos subyacente a la co-relaciones de autor tienen palabras clave, el contenido del texto, y citas de otros documentos. Además, autores en esa red también pueden estar relacionados en otros aspectos (relaciones de asesoramiento, membresía barcos en los comités de conferencia, amistades, etc.). Este tipo de relaciones de varias capas no son necesariamente susceptibles de los mismos tipos de técnicas de análisis de datos conocidas que aplicará a los datos "planas" o no relacionales.

***Las principales características de JUNG incluyen los siguientes:***

- Soporte para una variedad de representaciones de entidades y sus relaciones, incluyendo grafos dirigidos y grafos no dirigidos, gráficos multi-modales (gráficos que contienen más de un tipo de vértice o borde), gráficos con bordes paralelos (también conocidos como multigrafos), y hiper-graficas (que contienen hiper-bordes, cada uno de los cuales puede conectar cualquier número de vértices).

- Mecanismos para la anotación de gráficos, las entidades y las relaciones con los metadatos. Estas capacidades facilitan la creación de herramientas de análisis de conjuntos de datos complejos que pueden examinar las relaciones entre las entidades, así como los metadatos unido a cada entidad y relación.

- Las implementaciones de una serie de algoritmos de la teoría de grafos, análisis exploratorio de datos, análisis de redes sociales y de aprendizaje automático. Estos incluyen rutinas para la agrupación, descomposición, optimización, generación de gráficos al azar, análisis estadístico, y cálculo de las distancias de redes, flujos, y las medidas de clasificación (centralidad, **PageRank** , HITS, etc.)

- Un marco de visualización que hace que sea fácil de construir herramientas para la exploración interactiva de datos de la red. Los usuarios pueden elegir entre el diseño proporcionado y renderizados algoritmos, o utilizar el marco para crear sus propios algoritmos personalizados.

- Los mecanismos de filtrado que extraen subconjuntos de una red; esto permite a los usuarios centrarse su atención, o sus algoritmos, sobre partes específicas de una red.

Estas capacidades hacen a **JUNG** una buena plataforma para el análisis exploratorio de datos relacional en conjuntos de datos.

JUNG es un marco en el que las aplicaciones y herramientas para la manipulación de gráficos y los datos de red pueden ser construidas. Se puede utilizar en fragmentos simples de código para probar ideas o para ayudar en el desarrollo de una sofisticada herramienta con una interfaz gráfica de usuario. JUNG no es en sí mismo un stand-herramienta solo, sino más bien una biblioteca que se puede utilizar para apoyar la construcción de herramientas específicas.

La construcción de una herramienta que utiliza JUNG requiere algún conocimiento de programación Java. La distribución JUNG incluye muestras de pequeñas aplicaciones que utilizan JUNG para lograr ciertas tareas, pero que están destinados a ser ejemplos de cómo se podría utilizar JUNG, no herramientas en su derecho propio.

JUNG se ofrece libremente bajo la licencia BSD de código abierto, lo que permite a cualquiera crear obras derivadas de JUNG, siempre y cuando ellos reconocen la contribución de JUNG a su trabajo.

La naturaleza de código abierto del proyecto hace que sea más fácil para los usuarios escribir su propio código, y para entender cómo funciona la misma JUNG; como resultado, los miembros de la comunidad de usuarios de JUNG tienen podido contribuir una serie de extensiones, algoritmos y correcciones.

## 2.3.1 Teoría de Grafos

Gráficos (también conocidas como redes) consisten en un conjunto de vértices, **V**, y un conjunto de aristas, **E**; el número de vértices se denota por | **V** | y el número de bordes por | **E** |.

Los vértices (también conocido como nodos) representan entidades y bordes (también conocidos como arcos, enlaces o vínculos), que conectan los vértices, representan relaciones o acontecimientos que implican las entidades que los vértices representan. El número de aristas incidentes a un vértice se llama el grado de ese vértice.

Los gráficos en que cada borde tiene un valor numérico asociado (por ejemplo, el número de trabajos en coautoría) se llaman ponderada o gráficos valorado. Un subtipo de gráfico común es un gráfico k-partita (llamada grafico bipartito cuando k = 2), en el que los vértices se dividen en k subconjuntos disjuntos, y cada borde conecta vértices en particiones distintas.

La mayoría de los gráficos contienen bordes que conectan exactamente dos vértices; a menos que se especifique lo contrario, todos los gráficos en este trabajo tienen esta propiedad. (Gráficos en el que los bordes se pueden conectar cualquier número de vértices se llaman hiper-grafos, y sus bordes se llaman hiper-bordes.) Un borde que tiene una fuente definida y de destino (por ejemplo, uno en representación de "A ha citado B") son llamados **bordes dirigidos**; un borde que no es así (como uno en representación de "A y B son co-autor de un documento" se llaman **bordes no dirigidos**. Los gráficos que contienen ambos bordes dirigidos y no dirigidos son llamados gráficos de tipo mixto. Dos bordes se dice que son paralelas si se conectan el mismo conjunto de vértices y tienen la misma dirección / pedido.

Un gráfico se dice que está conectada si cada vértice es accesible desde cada otro vértice; muchos algoritmos (como algoritmos de centralidad) sólo están bien definidos en los gráficos conectados. Definimos la distancia entre dos vértices sea la longitud de la trayectoria más corta (en la ONU grafo subyacente ponderado) que los une; en otros contextos, la distancia puede referirse a la más corta camino ponderado.

Una red puede contener entidades de diferentes tipos, o con diferentes roles; también puede incluir diferentes tipos de relaciones o acontecimientos. Estos roles y tipos de interacción son colectivamente se hace referencia como modos. Una red que tiene un tipo de entidad y un tipo de relación es llamada una red de un solo modal; si la red tiene más de un tipo de entidad, relación, o ambos, se llama una red multimodal.

### 2.3.2 Programación y Java orientado a objetos

Java es un lenguaje de programación orientado a objetos. Esto generalmente significa que la programación implica **(a)** la definición de los tipos de objetos y sus capacidades, y **(b)** la construcción de objetos y utilizando sus capacidades en beneficio de las tareas deseadas.

Los tipos de objetos se definen por interfaces y clases (que puede poner en práctica una o más internacional caras, y puede extenderse (heredar el comportamiento de, o ser una subclase de) una sola otra clase). El objeto particular se dice que es una instancia de los tipos que aplique y se extiende. Java define una clase llamada Object, que es una superclase de todas las clases de Java (incluyendo, de manera implícita, cualquier clase definida por el usuario). Los comportamientos y las capacidades específicas de una clase se definen por la aplicación de los métodos de esa clase. Las clases pueden ser declaradas para ser abstracto, en el caso de que necesiten no suministrar implementaciones de cada método que ellos declaran, y una instancia de la clase no puede ser creada; esto puede ser útil para proporcionar implementaciones que son lo suficientemente general aplicar a las extensiones más esperadas de dicha clase. Los objetos pueden contener referencias a otros objetos.

API (Application Programming Interface) es un término común para una biblioteca de software, especialmente uno cuya filosofía de diseño hace hincapié en la compatibilidad hacia atrás a medida que evoluciona la biblioteca. Ambas bibliotecas estándar de Java y JUNG son APIs.

Hay algunas API de Java diferentes que se pueden usar para crear interfaces gráficas de usuario; dos de los más populares son Swing ( Sun Microsystems ( 2004 )) Y SWT (Fundación Eclipse( 2001 )).

### 2.3.3 Principios de diseño

JUNG hace uso de interfaces Java, las clases abstractas y clases de implementación en su tipo definiciones. Hay algunas razones que JUNG utiliza combinaciones de estas capas de abstracción.

En primer lugar, separa las especificaciones de sus implementaciones; esto hace JUNG más útil como lengua general para la manipulación de gráficos como datos. Así, por ejemplo, el ArchetypeGraph Interface especifica las capacidades de todos los gráficos, pero no restringe cómo esas capacidades están provistas.

En segundo lugar, combinamos esta noción de abstracción con la idea orientada a objetos de la herencia.

Por ejemplo, definimos el gráfico, la interfaz para todos los gráficos cuyos bordes conectan exactamente dos vértices, como una sub-interfaz de ArchetypeGraph, lo que significa que tiene todas las propiedades de ArchetypeGraph, pero con algunas propiedades adicionales que son específicos de tales gráficos; Esta refleja la relación real entre "gráficos" arquetipo y gráficos. Del mismo modo, la clase abstracta AbstractArchetypeGraph proporciona implementaciones de los métodos de ArchetypeGraph que no son específicos de cualquier variedad particular de gráfico.

En tercer lugar, es ventajoso para diseñar firmas de método, cuando sea posible, de modo que los usuarios no pueden suministrar argumentos del tipo equivocado; esto permite a los usuarios descubrir este tipo de errores cuando compilan su código, en lugar de cuando se les acaba la misma. Al mismo tiempo, los métodos deberían en general no ser tan limitados que sólo pueden aceptar los argumentos que utilizan una determinada implementación interna. Logramos estos dos objetivos mediante la especificación de firmas de método en términos de interfaces (que sólo especificar la capacidad) en lugar de las clases. Así, por ejemplo, métodos generalmente de JUNG operan en los gráficos, en lugar de SparseGraphs; esto permite a los usuarios utilizar estos métodos con sus propias implementaciones, si así lo desean, así como con cualquier implementación de gráfico que JUNG puede ofrecer en el futuro

Tradicionalmente, los conjuntos de datos de la red y el gráfico a menudo se han descrito matemáticamente como matrices. Matrices se implementan comúnmente como dos dimensiones arrays; esta representación facilita la rápida recuperación del borde (s), si los hay, que se conectan (s) dos vértices especificados (que operación se llama findEdge en JUNG). (Por ejemplo, en el contexto de la cooperación CiteSeer red de autoría, se podría utilizar a.findEdge (b) para recuperar el borde que mantiene una lista de los trabajos para los que los investigadores a y b han aparecido como coautores.) Sin embargo, esto representación generalmente no es factible para las redes a gran escala.
Por lo tanto, una representación 2D matriz de la red CiteSeer, que cuenta con unos 100.000 autores, requeriría 10 millones de entradas para representar las posibles conexiones; esto excede la capacidad de memoria de cualquier computadora de escritorio disponible en la actualidad.

Los algoritmos existentes para el análisis de redes que operan en matrices no escala bien a las matrices de este tamaño; muchos algoritmos de análisis de redes basado en matriz implican la multiplicación de matrices, o inversión de la matriz, que generalmente requieren O (| V |3) "Tiempo en matrices 2D". Además, esta representación es problemática de redes dinámicas (aquellas cuyo vértice conjunto puede hacerse más grande o más pequeño) y para redes con bordes paralelos, y no proporciona una forma natural para representar metadatos para entidades. Finalmente, las redes a gran escala son casi invariablemente muy escasas, por lo que casi todo el espacio en una matriz 2D que representa este tipo de red se pierde en la representación de vínculos ausentes.

Una representación alternativa común para gráficos y redes dispersas es la lista de adyacencia, en el que cada vértice mantiene una lista de aristas incidentes (o vértices adyacentes); esto requiere O (| V | + | E |) espacio. En el caso de la red CiteSeer, esto representa una ahorros en espacio de un factor de aproximadamente $10^4$.

A diferencia de la representación array 2D, esta representación no permite una implementación eficiente de findEdge; sin embargo, JUNG proporciona una implementación de esta representación, por las circunstancias en las que el ahorro de espacio es supremo.

La mayoría de las implementaciones actuales de vértices JUNG emplean una variante de la lista de adyacencia, que llamamos una representación mapa adyacencia: cada vértice mantiene una mapa de cada vértice adyacente al borde de conexión (o conectar borde conjunto, en el caso de gráficos que permitan aristas paralelas). (Mapas separados se mantienen, en su caso, por entrante aristas dirigidas, bordes dirigidos salientes y bordes no dirigidos.) Esto utiliza ligeramente más memoria que la representación lista de adyacencia, pero hace aproximadamente findEdge tan rápido como la operación correspondiente en la representación de matriz 2D. Esta representación hace que las estructuras de datos y algoritmos de Jung, en general, bien adaptado para el uso en gran escasa (redes O'Madadhain, Smyth y adánica ( 2005 ), Blanca y Smyth ( 2003 )).

### 2.3.4 Usando JUNG con otras bibliotecas

Al igual que los usuarios pueden escribir programas Java que utilizan JUNG proporcionar métodos para la manipulación de datos de la red, JUNG utiliza otras bibliotecas de Java (que son también independiente de la plataforma), para realizar ciertas funciones y por lo tanto reducir la duplicación de esfuerzos.

COMMONS COLLECTIONS ( Proyecto Jakarta Apache ( 2004 )) Es una biblioteca que mejora la básica API de Java para las colecciones de objetos (por ejemplo, al proporcionar operaciones básicas de teoría de conjuntos).

JUNG hace un amplio uso de esta biblioteca; en particular, commons colecciones proporciona la Interfaz predicado, que JUNG utiliza para implementar soporte para predicados, subconjuntos, y restricciones.

COLT ( CERN ( 2004 )) Es un conjunto de bibliotecas de alto rendimiento científica y computacionalmente. JUNG proporciona métodos para convertir los gráficos JUNG a matrices colt, y viceversa; esta permite a los usuarios de JUNG aplicar la extensa biblioteca de colt de las rutinas de álgebra lineal y de la matriz a los gráficos Jung. JUNG también hace uso ocasional de algunas de las funciones de álgebra matricial de colt.

XERCES ( Proyecto XML de Apache ( 2004 )) Es una biblioteca para el análisis sintáctico XML, que se utiliza para implementar la GraphML ( Brandes, Eiglsperger, Kaufmann, y Lerner ( 2004 )) capacidades entrada salida.

También es posible utilizar JUNG en conjunción con el código escrito en otros idiomas: tal código puede hacer uso de JUNG, ampliar las capacidades de Jung, o ambos. El Java Native Interface ( Sun Microsystems ( 2003 )) Permite código Java para acceder y ser visitada por, llamado código "nativo" (es decir, código que está escrito para una plataforma de programación en particular, como Windows o Linux). El paquete SJava ( Lang ( 2004 )) Proporciona una forma más especializada de llamar R y código S de Java y viceversa.

## 2.3.5 Los gráficos, vértices y aristas

### *Propiedades básicas y Operaciones*

Los gráficos, vértices y aristas cada uno tienen varias propiedades que se pueden extraer y operaciones que puedan realizar (o han realizado sobre ellos).

Dependiendo del tipo específico de gráfica, vértice, o borde, y sobre la aplicación utilizada, un objeto dado puede tener otras propiedades disponibles y / u operaciones. Cada uno de estos tipos básicos incluye operaciones para crear una copia de sí mismo; vértices y aristas también incluyen operaciones para identificar dicha copia en un gráfico especificado

Por definición, los vértices y aristas JUNG pueden ser elementos de no más de un gráfico; esto permite a los usuarios pregunten, por ejemplo, para los vértices vecinos de un vértice sin que tengan que especificar el gráfico como parte de la consulta. Esto implica que cada vértice y el borde deben ser capaces de identificar el gráfico del cual es un elemento; esta información puede ser de utilidad, por ejemplo, por escrito un método para determinar la longitud del camino más corto entre dos vértices especificados (en los que no se especifica explícitamente el propio gráfico).

### *Gráficos*

ArchetypeGraph define un gráfico para ser un contenedor de vértices y aristas, con varios métodos para acceder y modificar estos conjuntos, para la definición de las limitaciones de vértices y bordes, y especificando los oyentes.

Las interfaces hiper-gráficos y gráfico extienden de la interfaz ArchetypeGraph, y son especializadas para hiper-gráficos y gráficos de quien cuyos bordes conectan exactamente dos vértices, respectivamente.

Estas interfaces especifican firmas de método para añadir vértices y aristas; estos métodos son definido aquí en vez de en ArchetypeGraph a fin de permitir para los controles de tiempo de compilación de vértice y el tipo marginal. Los gráficos a su vez tienen sub-interfaces para los gráficos que sólo han dirigido bordes, sólo bordes no dirigidos, o que son k-partito.

Los tipos de gráficos escasos proporcionan implementaciones de los métodos gráfico que son adecuados para un gráfico escaso.

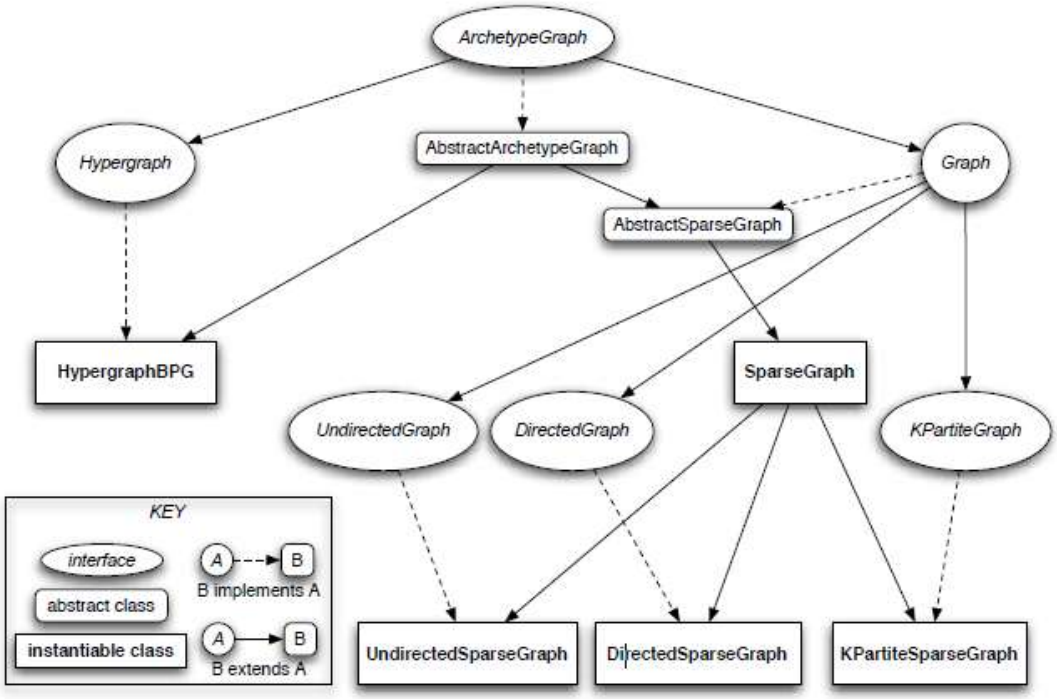



**Ilustración 1. Tipo de jerarquía para grafos.**

### *Vértices*

La interfaz ArchetypeVertex define un vértice como una entidad, que reside en una especificada gráfica, con el borde incidente y vecinos conjuntos de vértices; también define diferentes conveniencias y métodos de utilidad (como findEdge).

La interfaz **Vertex** amplía la interfaz ArchetypeVertex, y especifica método adicional para vértices de gráficos cuyos bordes conectan exactamente dos vértices; éstas incluyen métodos para acceder a los sucesores y predecesores de un vértice, o de su borde de salida y de entrada. En tales gráficos, un vértice **V** que está conectado a otro vértice **W** a través de un borde no dirigido es el sucesor tanto de w y su predecesor (y viceversa), y es considerado tanto un borde entrante y un borde de salida de v y w.

La interfaz **Hypervertex** también amplía la interfaz ArchetypeVertex; que sirve para identificar un vértice que puede añadirse a un hiper-gráfica.

La clase abstracta **AbstractSparseVertex** implementa algunos de los métodos básicos de vértices, y a su vez extendió por varias clases de vértices diferentes, cada uno de los cuales está diseñado para diferentes combinaciones de propiedades de gráfico (por ejemplo, si el gráfico sólo acepta bordes dirigidos, sólo bordes no dirigidos, o ambos, y si acepta bordes paralelo). Esta multiplicidad de tipos permite implementaciones Vertex más eficientes para tipos de gráficos más restringidos (tales como gráficos dirigidos que no permiten bordes paralelos).

En algunos casos, puede ser útil para tratar un gráfico dirigido g como un grafo no dirigido (por ejemplo, la hora de decidir cómo dibujar una foto de ella). Desde JUNG considera como sucesores y predecesores sean casos especiales de vecinos y saliente y bordes entrantes sean casos especiales de bordes incidentes, uno puede mirar de manera implícita en el subyacente grafo no dirigido de g por acceder a los vecinos e incidentes conjuntos de borde para cada vértice.

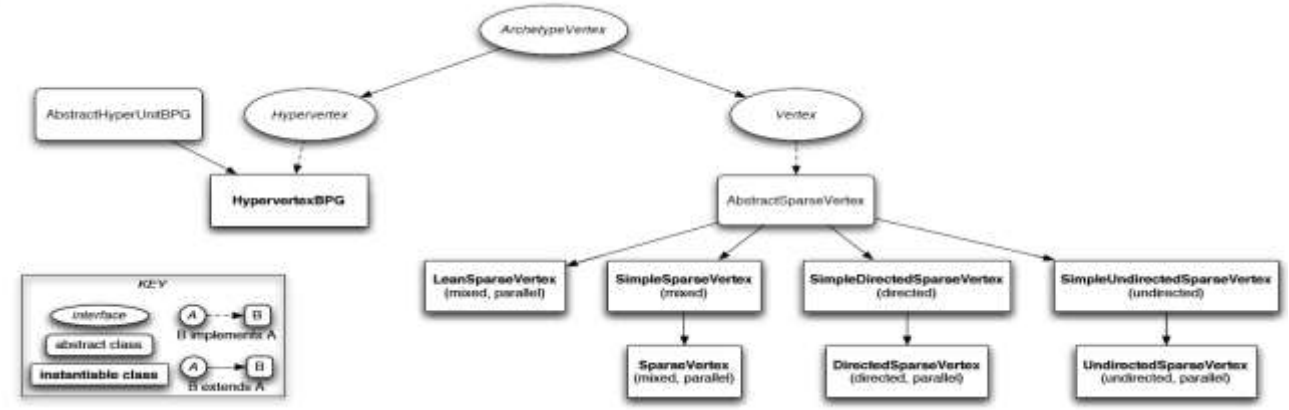



**Ilustración 2. Tipo de jerarquía para vértices.**

### *Bordes*

La interfaz ArchetypeEdge define un borde de ser una entidad, que reside en un determinado gráfico, con un conjunto de vértices incidente; también define varios métodos de conveniencia y utilidad (tal como uno que devuelve verdadero si un vértice especificado es incidente a este borde).

La interfaz **Edge** amplía la interfaz **ArchetypeEdge** para los bordes que conectan exactamente dos vértices; que es a su vez extendió por las interfaces que definen las operaciones adicionales para dirigido y los bordes no dirigidos.

La interfaz Hyperedge también amplía la interfaz ArchetypeEdge, y proporciona un método firma para añadir un vértice a un hyperedge existente. Las implementaciones Escasos de Edge se corresponden con las implementaciones Escasos de Vertex y gráfico.

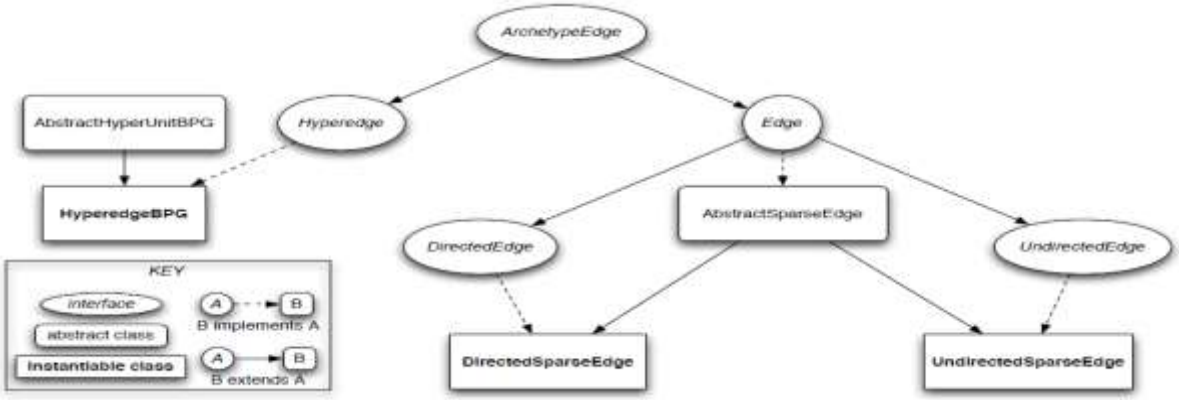



**Ilustración 3. Jerarquía ArchetypeEdge**

### *Crear, Añadir y quitar*

Creación de un gráfico puede realizarse de tres maneras. En primer lugar, se puede llamar al constructor del tipo deseado de gráfica, como en el siguiente ejemplo:

**DirectedGraph g = new DirectedSparseGraph ();**

En segundo lugar, se puede crear un gráfico leyéndolo desde un archivo. Actualmente, JUNG puede leer y escribir sencilla Pajek ( Batagelj y Mrvar ( 2004 )) Y GraphML (Brandes et al. ( 2004 )) Archivos;

En tercer lugar, se puede generar un gráfico algorítmicamente, ya sea con un método definido por el usuario (por ejemplo, uno que genera un gráfico basado en un modelo estocástico especificado), o con una de las clases que JUNG prevé la creación de grafos aleatorios.

Una vez un gráfico ha sido creado usando cualquiera de estos métodos, los vértices pueden ser creados y agregados a este gráfico:

**Vertex v1 = g.addVertex (nueva DirectedSparseVertex ());**

**Vertex v2 = g.addVertex (nueva DirectedSparseVertex ());**

Y una vez que existen vértices, que pueden estar conectados por los bordes:

**E Edge = new DirectedSparseEdge (v1, v2);**
**g.addEdge (e);**

Estas operaciones se pueden utilizar para actualizar la red CiteSeer coautoría como nuevos autores (nuevos vértices), y los nuevos papeles (nuevos bordes).

Tenga en cuenta que la creación de vértices / bordes y agregarlos a un gráfico que en realidad son dos diferentes operaciones, que se pueden combinar en una sola línea de código (como en la creación de vértices / adición ejemplo anterior). La naturaleza de dos etapas de este proceso hace posible la creación de vértices / bordes “huérfanos" que no son parte de un gráfico.

El comportamiento del borde JUNG y el vértice son métodos generalmente no especificado en vértices / bordes huérfanos.Esta distinción (entre la creación y adición) es útil, sin embargo, debido a que nos permite decorar vértices / bordes (es decir, conectar los datos a ellos) antes de añadirlos a un gráfico:

**Vertex v1 = new DirectedSparseVertex ();**
**setProfession (v1, "profesor");**
**g.addVertex (v1);**

En particular, si g contiene un vértice restricción que requiere que cada vértice haya definido una "Profesión", o si **g** es el mantenimiento de una lista de subconjuntos de predicado especificada (una para cada profesión), a continuación, el vértice debe ser tan decorado antes de que se añada a la gráfica.

Extracción de vértices y aristas de un gráfico puede ser útil en varios contextos, tales como la identificación de vértices / bordes (el conjunto de datos CiteSeer prima duplicado o no válido contiene una serie de documentos duplicados y autores); la creación de subgrafos; y la operación de un algoritmo que está poniendo a prueba los efectos de (temporal) cambia la gráfica. Esto puede ser hecho de la siguiente manera:

**g.removeEdge (e);**
**g.removeVertex (v1);**

La extracción de un borde de una gráfica no afectará a cualquier otra parte de la gráfica. La extracción de una vértice de un gráfico puede hacer que los bordes que son incidente a ese vértice que de ser eliminado estos bordes se convierten formando tratos. (Un borde mal formado es uno que es incidente al número equivocado de vértices. En los gráficos, donde los bordes están definidos para conectar los dos vértices, la eliminación de un vértice se traducirá en la eliminación de todos sus bordes incidentes.).

La extracción de un elemento de una gráfica no libera la memoria utilizada por dicho objeto; de hecho, un elemento puede ser removido de un gráfico y luego volver a insertarlo en ese gráfico o en un diferente gráfico). Al igual que con todos los programas de Java, el recolector de basura de Java es responsable de liberar la memoria para un objeto una vez que ya no está siendo utilizado.

## Restricciones

***Las siguientes restricciones se aplican a todos los bordes, vértices, y gráficos JUNG:***

- Un vértice / borde puede ser sólo en un gráfico a la vez.
- Un vértice / borde sólo puede ser añadido a un gráfico dado de una vez.
- Un borde puede no crearse incidente a vértices "huérfanos".
- Un borde no puede ser creado que une los vértices en diferentes gráficos.
- La direccionalidad de un borde debe coincidir con el de los vértices que se conecta, y la de la gráfica a la que se está añadiendo. (Así, por ejemplo, un DirectedSparseEdge No se puede añadir a una UndirectedGraph.)

Las limitaciones de vértices y bordes se definen mediante la Interfaz predicado, y puede ser visitada (y editado) a través de los getVertexConstraints y métodos getEdgeConstraints proporcionadas por ArchetypeGraph:

Colección edge_constraints = g.getEdgeConstraints ();
// No permitir bordes paralelos
**edge_constraints.add (Graph.NOT_PARALLEL_EDGE);**
// No permitir la libre bucles (bordes que conectan un vértice a sí mismo)
edge_constraints.add (nueva NotPredicate (SelfLoopPredicate.getInstance ());

Las restricciones pueden editarse libremente para gráficos vacíos, pero no se pueden añadir a los gráficos no vacíos (esto evita problemas de consistencia, ya que las restricciones sólo se verifican cuando los vértices y aristas se añaden a un gráfico).

No todas predicados están apropiado para usarse como limitaciones; Por ejemplo, **ReciprocatedDirectedEdgePredicate** solamente pases bordes dirigidos <u, v> para los que no también existe una ventaja antiparalela <v, u>; desde los bordes se añaden a un gráfico de uno a la vez, sin borde puede pasar este predicado cuando se comprueba como una restricción. La documentación para el predicados que JUNG proporciona generalmente especificar si se pueden usar como restricciones.

## 2.4 Edisoncorsx

El paquete Swing es parte de la JFC (Java Foundation Classes) en la plataforma Java. La JFC provee facilidades para ayudar a la gente a construir GUIs. Swing abarca componentes como botones, tablas, marcos, etc...

Las componentes Swing se identifican porque pertenecen al paquete javax.swing es por esa razón que las interfaces graficas de usuario se han convertido en una parte muy importante del software, no solo por la funcionalidad y la facilidad de manejo que puedan llegar a representar sino también porque hace de un software que sea atractivo.
La librería edisoncorsx se encarga de manejar la apariencia visual de los componentes y la forma en la que el usuario interactúa con ellos.

## 2.5 Jespxml

Jespxml en una librería que extiende de la clase “File”, la cual se usa para obtener información sobre archivos.

La librería Jespxml posee múltiples métodos para lectura y escritura de archivos escritos en el lenguaje xml.

Para comenzar a utilizar dicha librería se necesita crear una instancia de la clase Jespxml, utilizando el constructor personalizado:

```
//la ruta del archivo xml que se desee procesar

public JespXML(String pathname) {
        super(pathname);
        stream = null;//tipo de datos InputStream
        encoding = null;//tipo de encoding
    }

//función de lectura XML

public Object leer(Class clase) throws JAXBException{
        JAXBContext context = JAXBContext.newInstance(clase);
        Unmarshaller unmarshaller = context.createUnmarshaller();
        return unmarshaller.unmarshal(this);
    }
```

```
//funciones de escritura XML

public void escribir(Object objeto, Class clase) throws JAXBException,
FileNotFoundException, IOException{
        JAXBContext context = JAXBContext.newInstance(clase);
        Marshaller marshaller = context.createMarshaller();
        marshaller.setProperty(Marshaller.JAXB_FORMATTED_OUTPUT, true);
        crearCarpetasNecesarias();
        FileOutputStream fos = new FileOutputStream(this);
        marshaller.marshal(objeto, fos);
        fos.close();
    }
```

## 2.6 iText

**IText** es una biblioteca open source para crear y manipular archivos PDF en Java. Fue escrita por Bruno Lowagie, Paulo Soares, y otros; está distribuida bajo la Affero General Public License.

El mismo documento puede ser exportado en múltiples formatos, o múltiples instancias del mismo formato. Los datos pueden ser escritos a un fichero o, por ejemplo, desde un servlet a un navegador web.

El soporte de PDF de iText es, sin embargo, bastante extensivo. Esto soporta firmas basadas en PKI de PDF, cifrado de 40-bit y 128-bit, corrección de colores, PDF/X, gestión de colores por perfiles ICC, y es anfitriona de otras características.

## 2.7 Diagramas de red

Los diagramas de redes facilitan la representación de las relaciones de prioridad, con sucesiones lógicas y secuenciales. Para obtener un diagrama de redes adecuado, debemos tener actividades que se puedan identificar fácilmente, que tengan inicio y fin, que guarden relación entre ellas y con un tiempo específico para realizarse.

Toda red o diagrama permite realizar un control permanente del avance de obras, objetivos y metas, conforme a los calendarios previstos, señalando además el camino más corto de ejecución sin sacrificar la calidad (parecido al concepto de óptimo de pareto.

Red: Es una combinación de eventos, que describen de manera lógica la ejecución de los proyectos o actividades empresariales.

### 2.7.1 Elementos de una red

**Evento**: Señala el inicio y el fin de la tarea o acción, no consume tiempo ni recursos.

- Se representa a través de un nodo o un círculo.

**Actividad**: Consiste en un conjunto de tareas, que deben ejecutarse, para la realización de una obra; consume tiempo, tiene inicio y fin, requiere mano de obra, materia prima y otros recursos.

- Se representa por una flecha, cuya dirección indica la secuencia.

**Actividad ficticia**: Es aquella que no consume tiempo ni trabajo. Se representa por líneas entrecortadas y sirve para guardar la lógica de la red.

**Camino Crítico**: Es el camino más largo a través de la red y representa el menor tiempo posible para la ejecución del proyecto.


**Ilustración 4. Partes básicas de diagramación**

Las actividades o eventos deben ser debidamente enumerados o identificados dentro de una secuencia adecuada.

La forma de la red dependerá del nivel de actividades, si todos los elementos se hacen al mismo tiempo, tendrá una forma más vertical si es secuencial será horizontal.

***El tiempo que consume una actividad puede ser de tres tipos:***

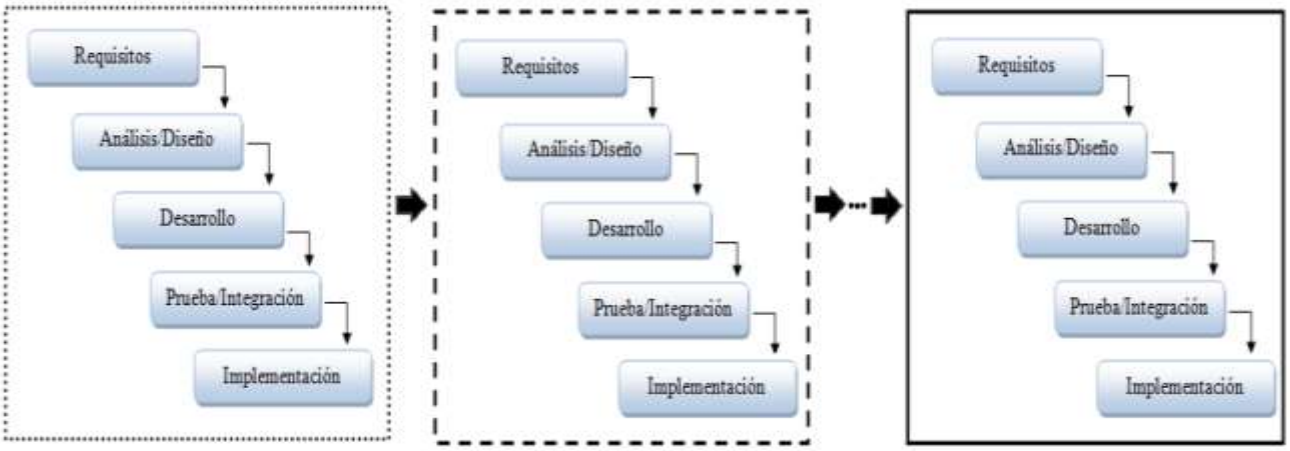
- Tiempo optimista: Constituye el menor tiempo en el que se puede ejecutar una actividad
- Tiempo pesimista: La peor situación posible (sin representar el fin del proyecto).
- Tiempo probable: Tiempo exacto posible, se pueden obtener un tiempo estimado a través de la siguiente relación:

  $te = a + 4m + b / 6$ donde: a = tiempo optimista, b = tiempo pesimista m = tiempo más probables te = tiempo estimado.

# 3. DISEÑO METODOLÓGICO

## Modelo iterativo

Los objetivos de una Iteración se establecen en función de la evaluación de las Iteraciones precedentes. Desde el principio, al final de cada Iteración se le entrega al usuario una versión completa y mejorada del Producto. El usuario es quien luego de cada Iteración evalúa el producto y corrige o propone mejoras. Estas Iteraciones irán refinando el sistema y se repetirán hasta obtener un producto que satisfaga al usuario. La Especificación de requisitos se realiza en forma creciente: a medida que los usuarios logran un mejor entendimiento del problema, éste es reflejado en el software. Es decir, el producto de cada etapa de especificación de requisitos es un agregado o mejora al producto de la etapa de especificación anterior.




| Versión 1 | Versión 2 | Versión 3 |
|---|---|---|
| 1. Se explora el fichero de configuración de la máquina virtual.<br><br>2. Se utiliza la librería JespXML para la interacción con el fichero de la máquina virtual.<br><br>3. Se visualiza por consola las interconexiones de las máquinas virtuales que se encuentren en red interna. | 1. Se utiliza la librería JUNG para visualizar y manipular las máquinas vituales.<br><br>2. Se visualiza de manera gráfica la topología de red creada por las máquinas virtuales conectadas en redes internas.<br><br>3. Se exporta la topología de red en formato de imagen. | 1. Se incorpora los comandos de VirtualBox para la creación, clonación y eliminación de las máquinas virtuales por medio de la clase Process de Java.<br><br>2. Se genera reporte en formato PDF de las máquinas virtuales contenidas en la topología. |

## 3.1 Descripción general de proyecto

Se pretende desarrollar una aplicación de escritorio en el entorno de java con el objetivo que sea multiplataforma y que el usuario cree las máquinas virtuales para diseñar el escenario de red deseado. Podrá configurar las conexiones de desde los adaptadores de cada máquina virtual, remover y agregar los elementos (swicth, pc), minimizar y maximizar el escenario de red, guardar el escenario de red en formato de imagen, generar reporte de las máquinas virtuales en formato PDF.

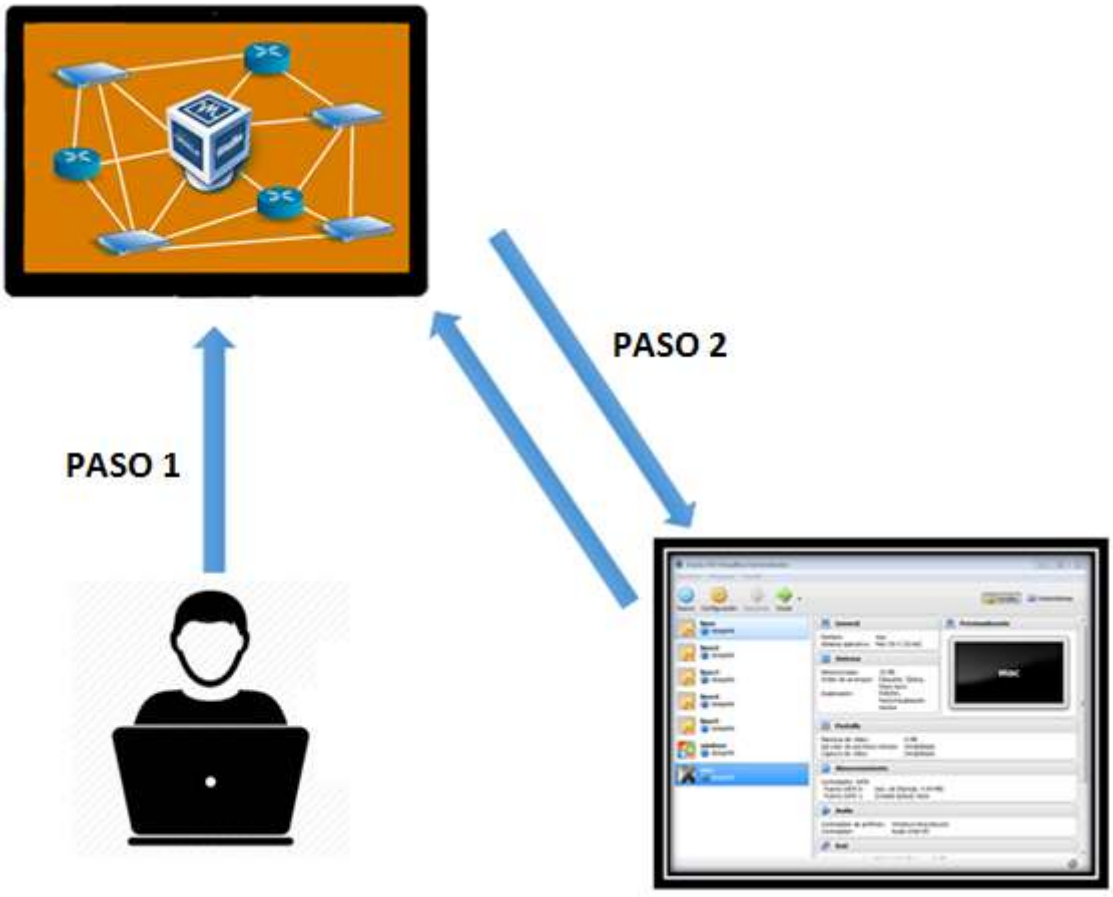



**Ilustración 4.Diagrama de implementación correcta de la aplicación**

## 3.2 Etapas del trabajo

### 3.2.1 Etapa 1: Etapa exploratoria

En esta etapa se exploran los archivos de la herramienta virtualizadora “VirtualBox” con el objetivo de saber cuál es el fichero o los ficheros donde se almacenan los datos de cada máquina virtual y la estructura que contiene cada archivo de máquina virtual.

***Resultado:*** se exploraron los archivos del VirtualBox, obteniendo que las máquinas virtuales se crean en una carpeta por defecto llama “VirtualBox VMs” ubicada en la cuenta del usuario en donde las máquinas vituales están dentro de un carpeta con su mismo nombre, esta carpeta contiene varios archivos pero el que contiene los datos para manipular dicha máquina es el fichero con extensión “.vbox” (VirtualBox Machine Definition), fichero escrito en lenguaje XML.

### 3.2.2 Etapa 2: Localización de interfaces de red

En esta etapa se establece el directorio donde buscara las máquinas virtuales de manera estática para localizar los ficheros de cada máquina virtual y así acceder a ellos para obtener la información y presentarla en pantalla.

***Resultado:*** se esperaba obtener todos los datos de las interfaces de red de cada máquina virtual creada.

### 3.2.3 Etapa 3: Visualización de conexiones de máquinas virtuales en redes internas.

Una vez obtenidos los datos de las interfaces de red se procede a filtrar dichos datos creando una lista en los cuales se almacena la interfaz de red, nombre de la red y el tipo de conexión que esta posee, dicha lista es almacenada en una lista de equipos la cual luego es leída para obtener los nombres de las redes que existen en todas las máquinas virtuales. Los datos son representados gráficamente a través de librería JUNG.

***Resultado:*** representación de máquinas a través de los vértices y las conexiones de dichas maquinas a las redes internas a través de los bordes.

### 3.2.4 Etapa 4: Manipulación de ficheros creados por virtual box.

Manipulación de los ficheros de configuración a través de la librería Jespxml accediendo a dichos ficheros modificando el nombre de la red que fue seleccionada.

***Resultado:*** se puede enlazar de una red interna a otra o desconectarlo de ser deseado, actualizando los atributos del fichero y el gráfico en tiempo real.

### 3.2.5 Etapa 5: Creación, clonación y eliminación de máquinas virtuales

En esta etapa el usuario podrá crear, clonar y eliminar máquinas virtuales, esto se hace atreves de una clase “Process” que es nativa de java en conjunto con los comando que se pueden ejecutar por medio del vboxmanage, otorgado por la herramienta virtualizador VirtualBox.

***Resultado:*** Se visualiza una ventana modal correspondiente a la acción que el usuario desee.

### 3.2.6 Etapa 6: Creación de nuevo proyecto

En esta etapa el usuario elabora un proyecto nuevo con máquinas virtuales nuevas o clonadas.

***Resultado:*** Se visualiza una ventana modal con tres pasos, el 1er paso muestra un JtextField para ingresar el nombre del proyecto, el 2do paso si desea crear o clonar máquinas virtuales y el 3er paso muestra el listado de las máquinas virtuales de dicho proyecto con un botón de finalizar.

### 3.2.7 Etapa 7: Almacenar datos en disco

Exportación del escenario de red en formato de imagen con extensión .png utilizando la clase java.awt.Robot, exportación de las máquinas virtuales que se encuentran en la topología en formato PDF utilizando la librería iText.

**Resultado:** imágenes en formato png, reporte de las máquinas virtuales en formato PDF.

### 3.2.8 Etapa 8: Prueba de la aplicación

En esta etapa se realizó la prueba de la aplicación en las siguientes asignaturas: Administración de sistemas operativos. (SEMESTRE VII), Administración de servicios de red I. (SEMESTRE VIII), Administración de servicios de red II. (SEMESTRE IX) de las carreras de Ingeniería en sistemas de información e Ingeniería en telemática que ofrece la UNAN-LEÓN.

**Resultado:** Los usuarios finales pudieron realizar las prácticas planteadas por el docente de una manera sencilla al interactuar con la aplicación dando resultados exitosos.

### 3.2.9 Etapa 9: Redacción del informe final

En esta etapa se contempla la redacción del informe final en el cual se llevó a cabo una serie de procesos para la creación y finalización de la aplicación multiplataforma, documentando cada una de las etapas y visualizando los resultados obtenidos.

**Resultado:** Informe final de tesis monográfica para optar al título de Ingeniero en sistemas de información.

# 3.3 Materiales

## 3.3.1 Hardware

Las siguientes características son la del ordenador donde se llevara a cabo la creación de la aplicación, cabe señalar que se puede utilizar una computadora q no ocupe mucho recurso.

***Características del ordenador:***

- 320 GB de disco duro.
- 2 GB de memoria RAM.
- Procesador AMD E-250 processor 1.5 GHZ

## 3.3.2 Software

- Sistema Operativo Windows 7 Profesional.
- VirtualBox versión 5.0 como mínima
- NetBeans IDE 8.0.2
- JDK (Java Development Kit) v 1.8.0_31-b13

# 4. Resultados

## 4.1 Localización de interfaces de red.

Para la obtención de los datos referentes a las máquinas virtuales y las interfaces de red que poseen se accedió al fichero de configuración de cada máquina virtual a través de la librería **jespxml.** El acceso al código **XML** del fichero se realizada a través del **DOM** (Document Object Model).

La jerarquía que posee este fichero para poder acceder a la información del nodo indicado es la siguiente:

**VirtualBox**

**…….Machine**

**……Hardware**

**……..Network**

**…….. Adapter**

```
<Network>
  <Adapter slot="0" enabled="true" MACAddress="080027EC6B52" cable="true" speed="0" type="82540EM">
    <DisabledModes>
      <NAT>
        <DNS pass-domain="true" use-proxy="false" use-host-resolver="false"/>
        <Alias logging="false" proxy-only="false" use-same-ports="false"/>
      </NAT>
      <NATNetwork name="NatNetwork"/>
    </DisabledModes>
    <InternalNetwork name="red1"/>
  </Adapter>
  <Adapter slot="1" enabled="false" MACAddress="080027503756" cable="false" speed="0" type="82540EM">
    <DisabledModes>
      <NAT>
        <DNS pass-domain="true" use-proxy="false" use-host-resolver="false"/>
        <Alias logging="false" proxy-only="false" use-same-ports="false"/>
      </NAT>
    </DisabledModes>
  </Adapter>
  <Adapter slot="2" enabled="false" MACAddress="080027A355B5" cable="false" speed="0" type="82540EM">
    <DisabledModes>
      <NAT>
        <DNS pass-domain="true" use-proxy="false" use-host-resolver="false"/>
        <Alias logging="false" proxy-only="false" use-same-ports="false"/>
      </NAT>
    </DisabledModes>
  </Adapter>
```

**Ilustración 5.Atributos del nodo "Adapter" dentro del código xml del fichero**

Cuando se accede hasta este nodo (**Adapter**), se captura los valores de cada atributo de cada adaptador como se muestra en la siguiente imagen:

Y los nodos hijos de cada adaptador para poder obtener información del tipo de redes que posee el adaptador. Estos pueden ser:

- InternalNetwork (Red Interna)
- NAT(traduce la dirección ip)
- BridgedInterface (puente)
- HostOnlyInterface (Solo anfitrión)
- No conectado (No se encuentra en ninguna red)


**Ilustración 6.Tipos de redes de cada adaptador**

## 4.2 Visualización de conexiones de máquinas virtuales en redes internas.

Antes de poder representar las máquinas virtuales se debía de filtrar las máquinas que presenten alguna interfaz de red en modo red interna (Internal Network ) obteniendo las redes internas a las que estas máquinas estuvieran conectadas, esto se logró creando una clase que guardara los datos más relevantes de la máquina virtual a como se representa en las siguientes imágenes:

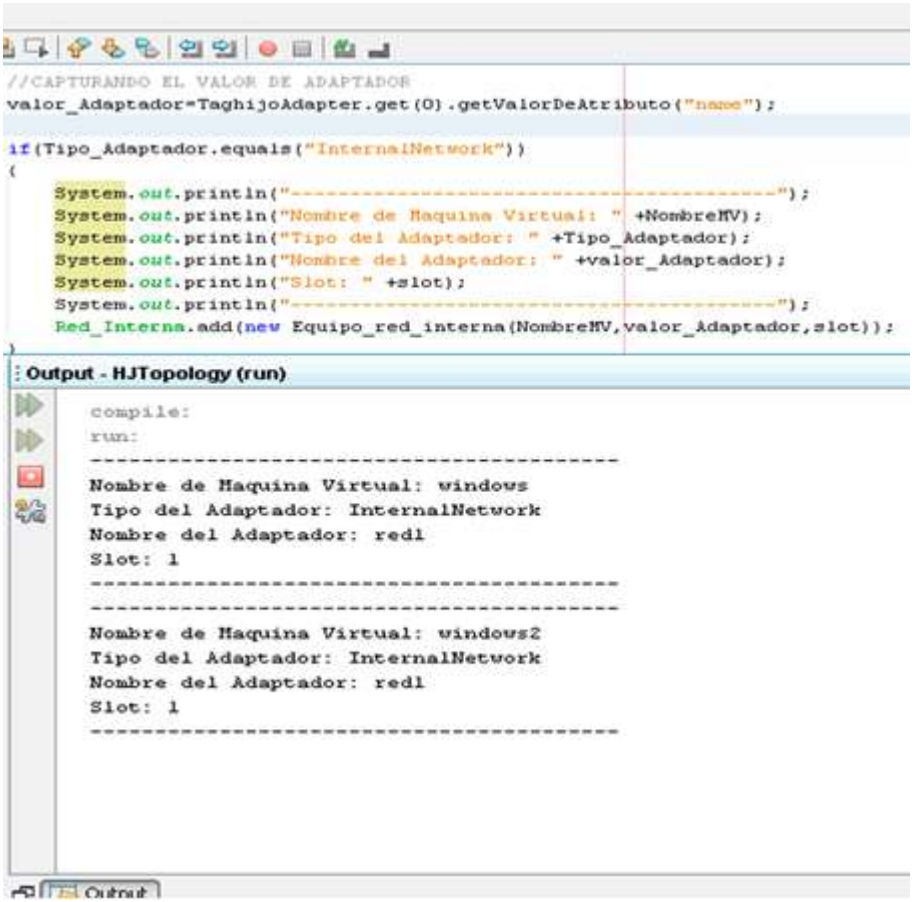



**Ilustración 7.Filtrar máquinas en redes internas**

Luego de filtrar las máquinas virtuales que posean interfaces con el tipo de Red Interna procedemos obtener la ruta de las máquinas virtuales interactivamente mostrando solo las máquinas virtuales en la clase **Equipo_red_interna** como se representa en el siguiente conjunto de imágenes:


**Ilustración 8.Selección de máquinas virtuales de la lista**

Cargando alguna de las máquinas de la lista que se presentan se procede a construir el escenario de red.

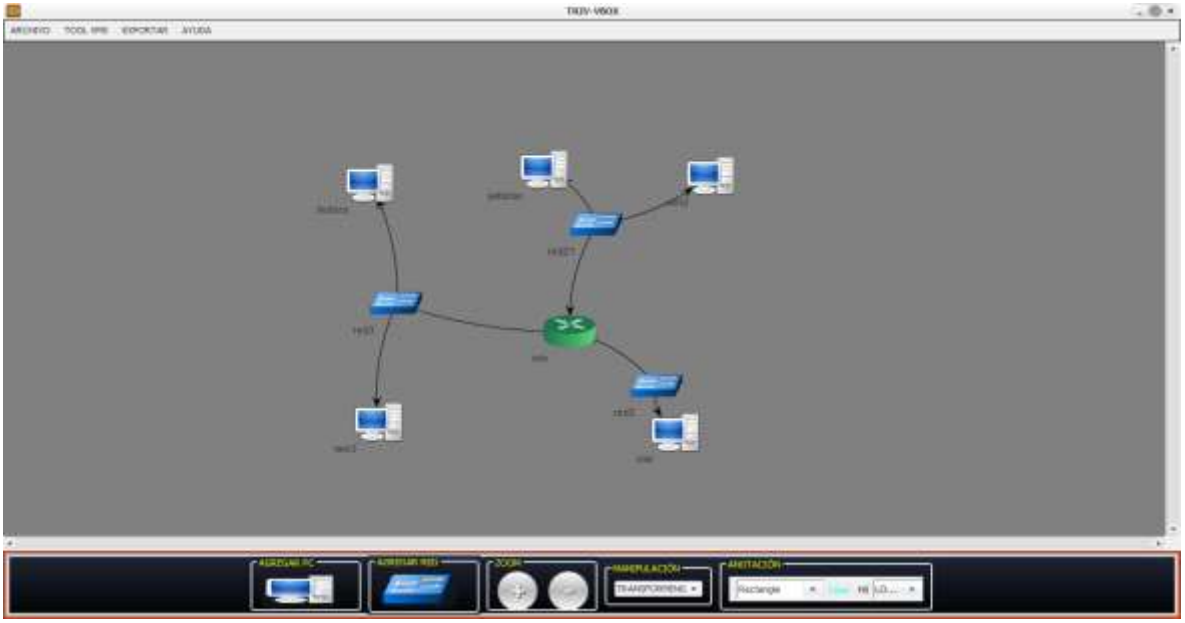


**Ilustración 9.Visualización del escenario de red**

Presentando los equipos terminales, switch y router como Vertex (Vértices) y la conexión entre ellos como Edge (bordes).

## 4.3 Manipulación de ficheros creados por VirtualBox.

Una vez pulsado el botón guardar, se llama a una función denominada “EDITARXML” que se encuentra en el fichero ManipularXml.java se debe de crear una instancia de la clase antes mencionada.

```
    public void EditarXML(File rutaPadre,List<Equipo> Lista_Equipos)
    {
       String Nom="";

       for(int i=0;i<Lista_Equipos.size();i++)
       {
          Nom=Lista_Equipos.get(i).GetNombre();//nombre de la maquina virtual
          String directorio=rutaPadre+"/"+Nom+"/"+Nom+".vbox";
          JespXML xml=null;
          xml=new JespXML(new File(directorio));
          Tag raiz = null;
          try
          {
             raiz=xml.leerXML();

             //CAPTURANDO EL NOMBRE DE LA MAQUINA VIRTUAL
             String NombreMV=raiz.getTagsHijos().get(0).getValorDeAtributo("name");
             System.out.println("\n\nMAQUINA:"+NombreMV);

             if(NombreMV.equals(Nom))//si el nombre de pc q est en la lista de tipo equipo es igual a la del
xml
             {
                for(int j=0;j<Lista_Equipos.get(i).GetList_Adapter().size();j++)
                {
                   int slot=Lista_Equipos.get(i).GetList_Adapter().get(j).GetSlot();
                   String enabled=Lista_Equipos.get(i).GetList_Adapter().get(j).getHabilitado();
                   String cable=Lista_Equipos.get(i).GetList_Adapter().get(j).getCable();
                   String tipoRed=Lista_Equipos.get(i).GetList_Adapter().get(j).GetTipo();
                   String valor=Lista_Equipos.get(i).GetList_Adapter().get(j).GetValor();

                   //Actualizando el enable,cable del adaptador actual

raiz.getTagsHijos().get(0).getTagHijoByName("Hardware").getTagHijoByName("Network").getTagHijoB
yName("Adapter",          Tag.Cantidad.TODOS_LOS_TAGS).get(slot).actualizarValorAtributo("enabled",
enabled);

raiz.getTagsHijos().get(0).getTagHijoByName("Hardware").getTagHijoByName("Network").getTagHijoB
yName("Adapter", Tag.Cantidad.TODOS_LOS_TAGS).get(slot).actualizarValorAtributo("cable", cable);


raiz.getTagsHijos().get(0).getTagHijoByName("Hardware").getTagHijoByName("Network").getTagHijoB
yName("Adapter", Tag.Cantidad.TODOS_LOS_TAGS).get(slot).eliminarTodosLosTagHijos();
                   //Actualizando el tipo y el valor de la red
                   switch(tipoRed)
                   {
                      case "InternalNetwork":

raiz.getTagsHijos().get(0).getTagHijoByName("Hardware").getTagHijoByName("Network").getTagHijoB
yName("Adapter", Tag.Cantidad.TODOS_LOS_TAGS).get(slot).eliminarTodosLosTagHijos();
```

```
                        Tag hijo = new Tag("InternalNetwork");
                        hijo.addAtributo("name",valor);

raiz.getTagsHijos().get(0).getTagHijoByName("Hardware").getTagHijoByName("Network").getTagHijoB
yName("Adapter", Tag.Cantidad.TODOS_LOS_TAGS).get(slot).addTagHijo(hijo);
                        break;

                    case "NAT":

raiz.getTagsHijos().get(0).getTagHijoByName("Hardware").getTagHijoByName("Network").getTagHijoB
yName("Adapter", Tag.Cantidad.TODOS_LOS_TAGS).get(slot).eliminarTodosLosTagHijos();

                        Tag hijoNAT = new Tag("NAT");

                        Tag hijoDNS = new Tag("DNS");
                        hijoDNS.addAtributo("pass-domain", "true");
                        hijoDNS.addAtributo("use-host-resolver", "false");
                        hijoDNS.addAtributo("use-proxy", "false");

                        Tag hijoAlias = new Tag("Alias");
                        hijoAlias.addAtributo("logging","false");
                        hijoAlias.addAtributo("proxy-only","false");
                        hijoAlias.addAtributo("use-same-ports","false");

                        //Añadiendo el DNS y el ALIAS al NAT
                        hijoNAT.addTagHijo(hijoDNS);
                        hijoNAT.addTagHijo(hijoAlias);


raiz.getTagsHijos().get(0).getTagHijoByName("Hardware").getTagHijoByName("Network").getTagHijoB
yName("Adapter", Tag.Cantidad.TODOS_LOS_TAGS).get(slot).addTagHijo(hijoNAT);

                        break;
                }

            }


          }

          xml.escribirXML(raiz);
          xml=null;
          raiz=null;

        }catch(Exception e)
        {
          System.out.println(e.getMessage());
        }
      }
    }
```

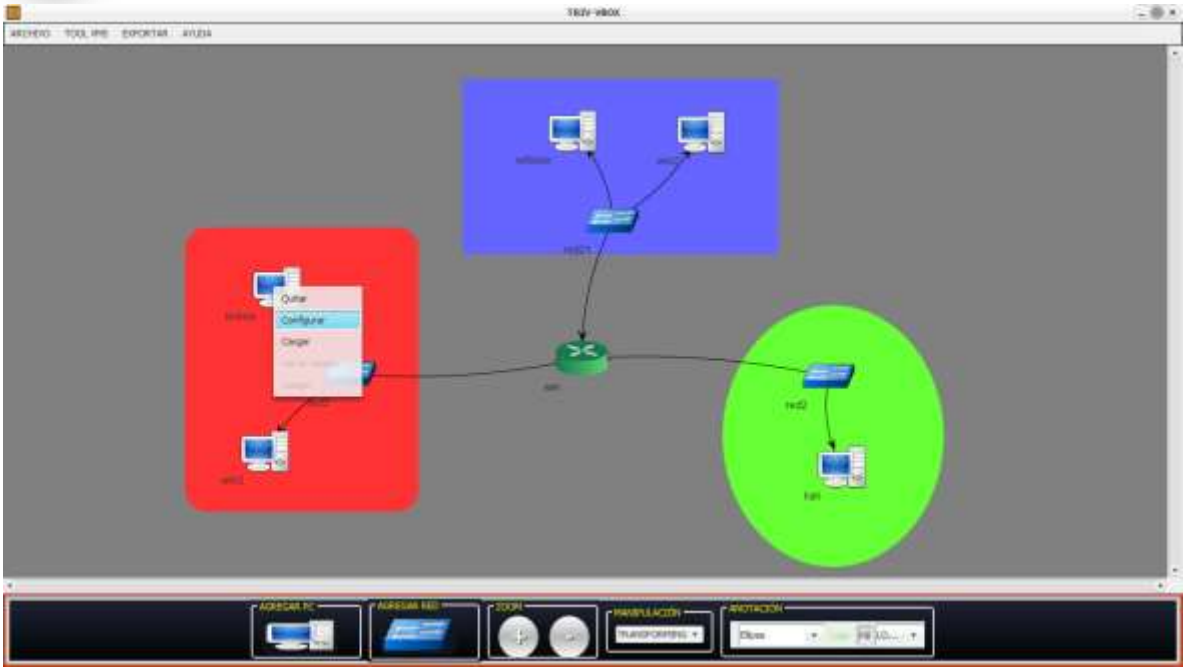


**Ilustración 10.Selección del ítem "configurar" del menú contextual**

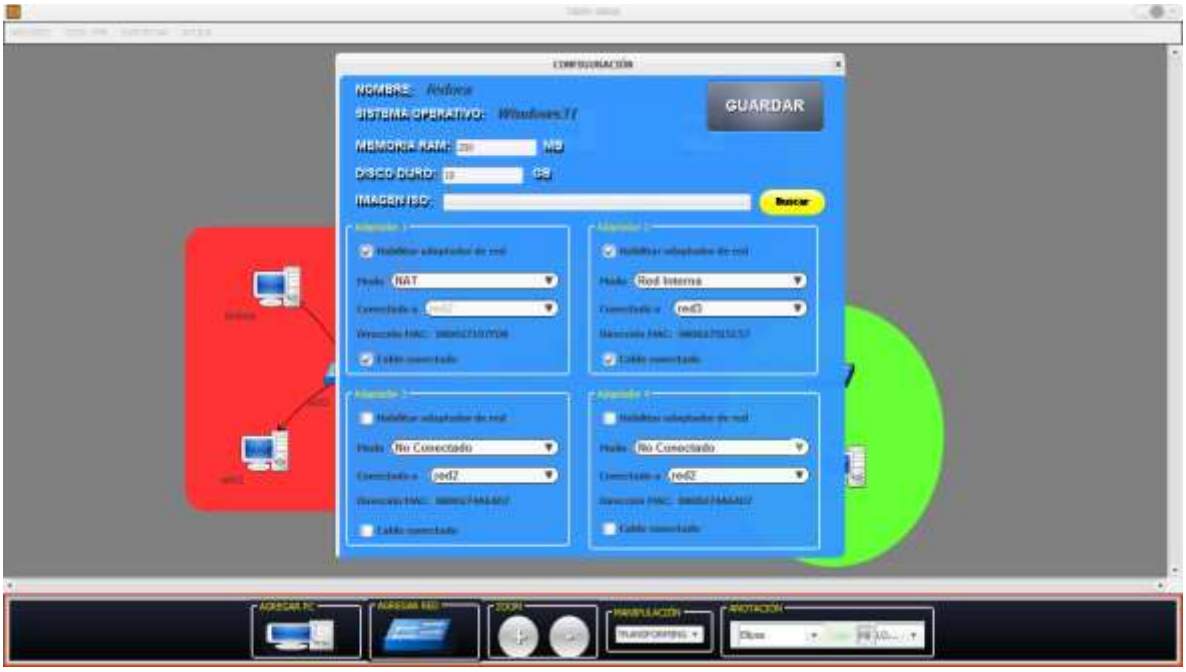



**Ilustración 11.Ventana modal para configurar los adaptadores, memoria RAM, disco duro, imagen ISO de la máquina seleccionada.**

## 4.4 Creación, clonación y eliminación de máquinas virtuales.

Para poder ejecutar los comandos se debe de acceder a los métodos estáticos de la clase llamada Ccomando, creada con el objetivo de lanzar los procesos con el vboxmanage del VirtualBox.

- ✓ **Creación de máquina virtual**

```
//Creamos la máquina virtual y poner sistema operativo

        String comando="vboxmanage createvm --name "+caja_nombre.getText().toString()+" --register
--ostype "+cb_OSversion.getSelectedItem().toString();

        String cadenaBuscada=Ccomando.buscarRutaentrecomando(comando,"Settings");

        String nom_vbox=caja_nombre.getText().toString()+".vbox'";
        int tam_nomvbox=nom_vbox.length();
        int tamCompleto=cadenaBuscada.length();
        int tamanio=tamCompleto-tam_nomvbox;

        String rutaCap=Ccomando.RetornarRuta(cadenaBuscada,16,tamanio);

        //concatenando para que puede ser ejecutado sin error el disco duro a la ruta deseada
        rutaCap="\""+rutaCap;
        String vdi=".vdi\" ";
        String completado=rutaCap+caja_nombre.getText().toString()+vdi;
        //fin de la concatenacion

        //Creacion de disco duro
        int disc=Integer.valueOf(caja_Disco.getText().toString());
        disc=disc*1024;
        comando="";
        comando="VBoxManage createvdi --filename "+completado+" --size "+disc;
        System.out.println(comando);
        Ccomando.Lanzar_Proceso(comando);//lanzando el proceso


        //Creando el controlador SATA
        comando="";
        comando="vboxmanage storagectl "+caja_nombre.getText().toString()+" --name \"Sata
Controller\" --add sata --controller \"IntelAHCI\"  --bootable on";
        Ccomando.Lanzar_Proceso(comando);//lanzando el proceso


        //Asociando el controlador SATA al disco duro
        comando="";
        comando="vboxmanage storageattach "+caja_nombre.getText().toString()+" --storagectl \"Sata
Controller\" --port 0 --device 0 --type hdd --medium "+completado;
        Ccomando.Lanzar_Proceso(comando);//lanzando el proceso

        if(!rutaISO.equals(""))
        {
          //Añadir un DVD y una ISO  para poder empezar con la instalación mas tarde
          comando="";
          comando="vboxmanage storageattach "+caja_nombre.getText().toString()+" --storagectl \"Sata
Controller\" --port 1 --device 0 --type dvddrive --medium "+rutaISO;
          Ccomando.Lanzar_Proceso(comando);//lanzando el proceso
        }
```

```
        //Asignamos la memoria correspondiente a la máquina virtual
        comando="";
        comando="VBoxManage modifyvm "+caja_nombre.getText().toString()+" --memory
"+caja_RAM.getText().toString();
        Ccomando.Lanzar_Proceso(comando);//lanzando el proceso

        JOptionPane.showMessageDialog(null,"MAQUINA VIRTUAL
"+caja_nombre.getText().toString().toUpperCase()+" CREADA CON ÈXITO","INFORMACIÒN",1);

        //VERIFICANDO CREACION DEL GRUPO
        if(!CadRutaGP.isEmpty() && CadRutaGP.length()>0 && !CadRutaGP.equals("") &&
!NomGrupo.isEmpty() && NomGrupo.length()>0 && !NomGrupo.equals(""))
        {
           try {
                comando="";
                comando="VBoxManage modifyvm "+caja_nombre.getText().toString()+" --groups
\"/"+NomGrupo+"\"";
                Ccomando.Lanzar_Proceso(comando);//lanzando el proceso
                AddVMSgrupo++;//esta variable es utilizada para ver cuantas maquinas virtuales se
crearon en el grupo
                        //y para validar la visualizacion de las maquinas virtuales en la topologia

           } catch (Exception ex) {
              Logger.getLogger(JDCrearVMS.class.getName()).log(Level.SEVERE, null, ex);
           }

        }
        //FIN DE LA VERIFICACION DE LA CREACION DE GRUPO Y EL TRASLADO DE LAS MAQUINAS
VIRTUALES A ESE GRUPO


        nuevo();//Se limpian las cajas y se reestablece al modo inicial
```


**Ilustración 12.Ventana modal para crear una nueva máquina virtual.**

### ✓ Clonación de máquina virtual

```
//UTILIZADA PARA LA CLONACIÓN COMPLETA
public void ClonarVMS_Total(String pc_origen,String pc_clon,String grupo)
 {
    String comando="vboxmanage clonevm "+pc_origen+" --mode all --name "+pc_clon+" --groups \"/"+grupo+"\" --register";
    Ccomando.Lanzar_Proceso(comando);//lanzando el proceso

    String cmdGP="VBoxManage modifyvm "+pc_clon+" --groups \"/"+grupo+"\"";
    Ccomando.Lanzar_Proceso(cmdGP);//lanzando el proceso


    ResetearAdaptadores(pc_clon);//reseteando todos los adaptadores
 }


//UTILIZADA PARA LA CLONACIÓN ENLAZADA
 public void ClonarVMS_Enlazada(String pc_origen,String pc_clon,String grupo)
 {
    String cmd1="vboxmanage snapshot "+pc_origen+" take \"Base enlazada para "+pc_origen+" y "+pc_clon+"\"";
    Ccomando.Lanzar_Proceso(cmd1);//lanzando el proceso

    String cmd2="vboxmanage clonevm "+pc_origen+" --snapshot \"Base enlazada para "+pc_origen+" y "+pc_clon+"\" --options link --name "+pc_clon+" --groups \"/"+grupo+"\" --register";
    Ccomando.Lanzar_Proceso(cmd2);//lanzando el proceso

    String cmdGP="VBoxManage modifyvm "+pc_clon+" --groups \"/"+grupo+"\"";
    Ccomando.Lanzar_Proceso(cmdGP);//lanzando el proceso

    ResetearAdaptadores(pc_clon);//reseteando todos los adaptadores
 }
```


**Ilustración 13.Ventana modal para clonar máquina virtual**

- ✓ **Eliminación de máquina virtual**

```
String nombre="",comando="";
nombre=lista_vbox.getSelectedValuesList().get(i).toString();

 comando=" VBoxManage unregistervm "+nombre+" --delete";

  Lanzar_Proceso(comando);//Lanzando proceso
```

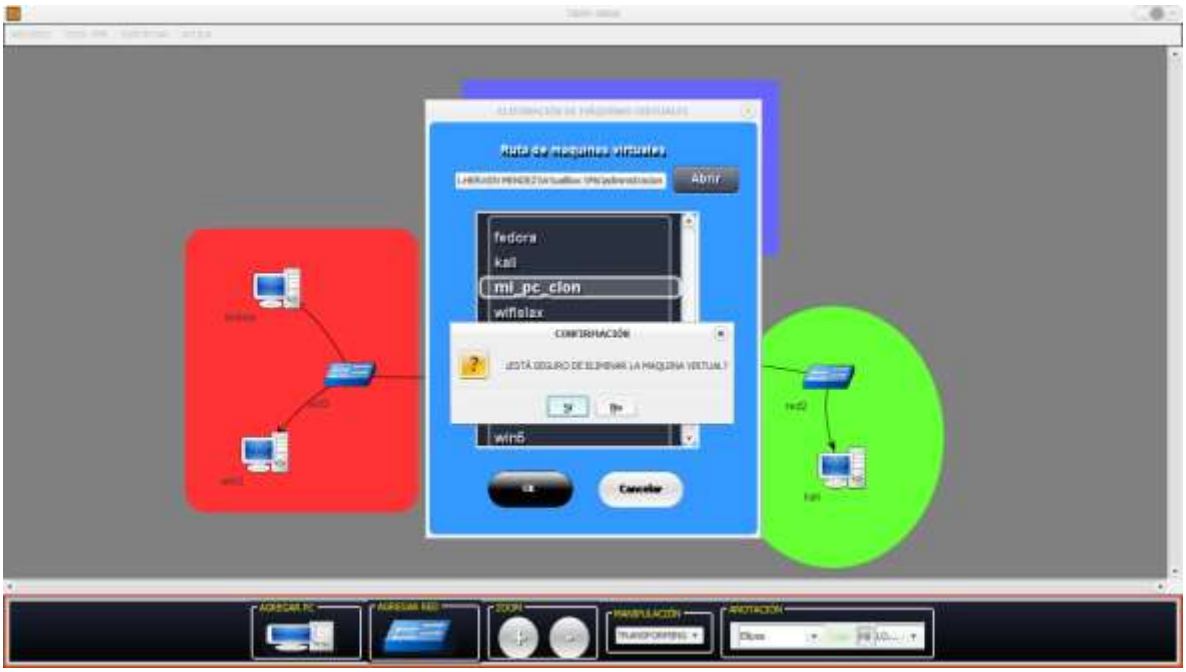



**Ilustración 14.Ventana modal para eliminar las máquina virtuales seleccionadas.**

## 4.5 Creación de un nuevo proyecto

En este el usuario tendrá que seleccionar **ARCHIVO>Nuevo Proyecto**. Una vez pulsado se mostrará una ventana modal con tres pasos.

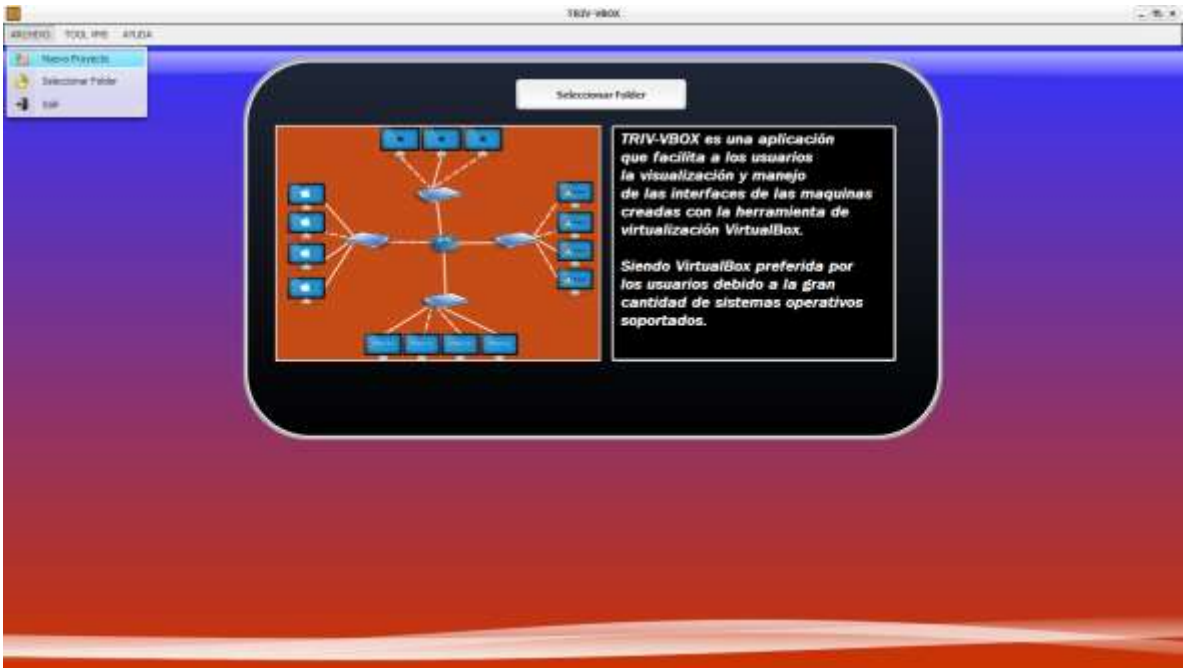



**Ilustración 15.Selección para crear un nuevo proyecto**

- ✓ **Paso #1**


**Ilustración 16.Inserción nombre del proyecto**

- ✓ **Paso #2**


**Ilustración 17.Opciones para crear nuevas máquinas virtuales o clonar si existen.**

- ✓ **Paso #3**


**Ilustración 18.Visualizando listado de las máquinas virtuales del proyecto nuevo.**

## 4.6 Almacenar datos en disco.

- ✓ **Formato imagen**

En este el usuario tendrá que seleccionar **EXPORTAR>Topología en imagen**. Una vez pulsado se presentara una ventana modal la cual pedirá el nombre de la imagen y hacia donde la desea guardar. La imagen se guardara con la extensión **png**.

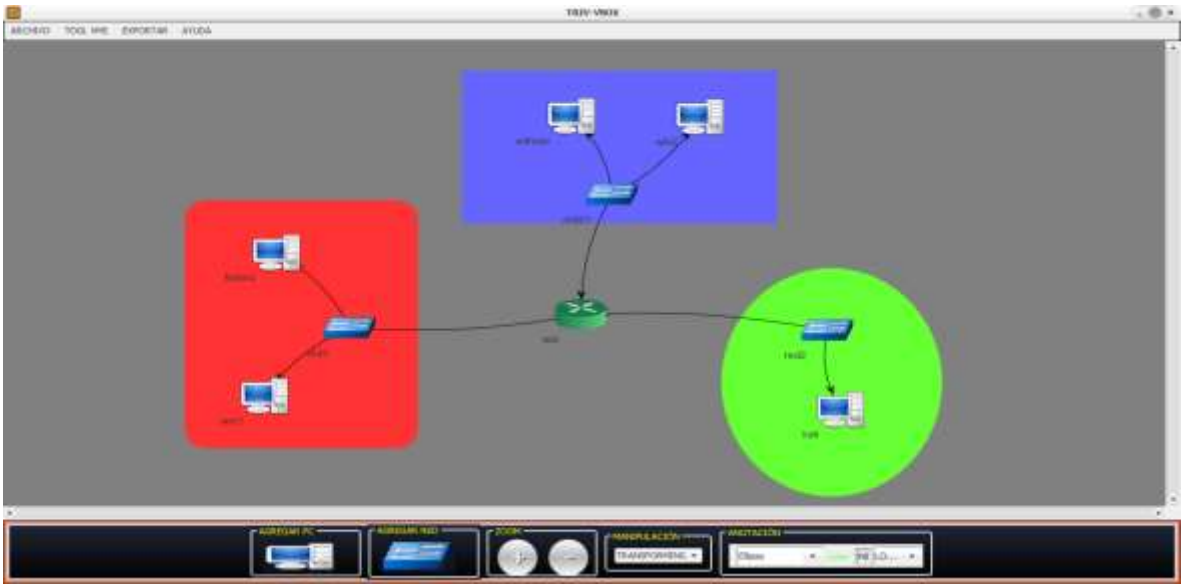


**Ilustración 19.Escenario de red dividido por zonas de colores**

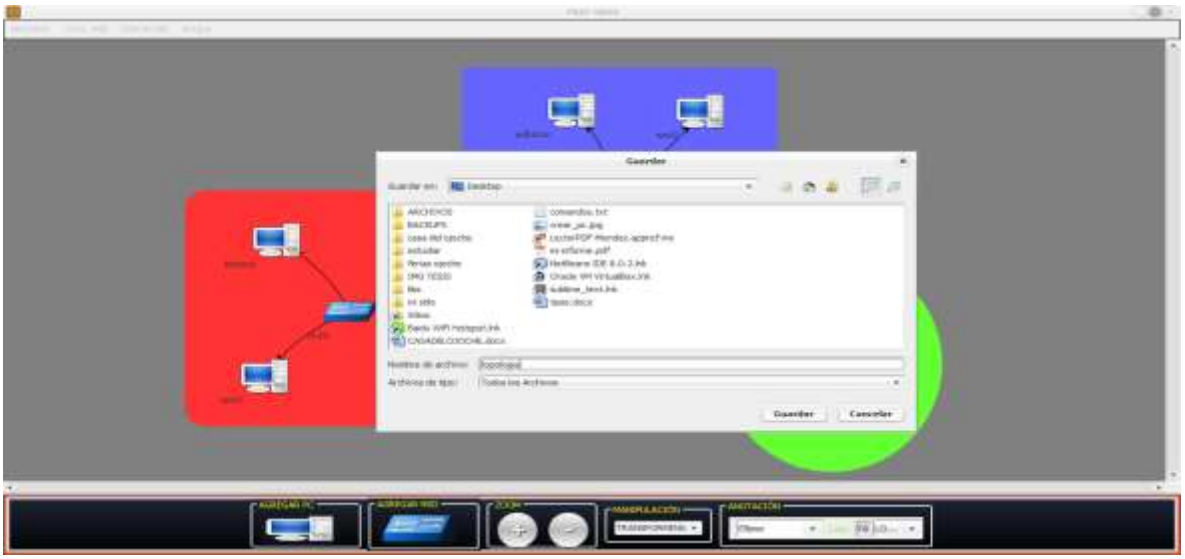


**Ilustración 20.Ventana modal para guardar el escenario de red en formato de imagen**

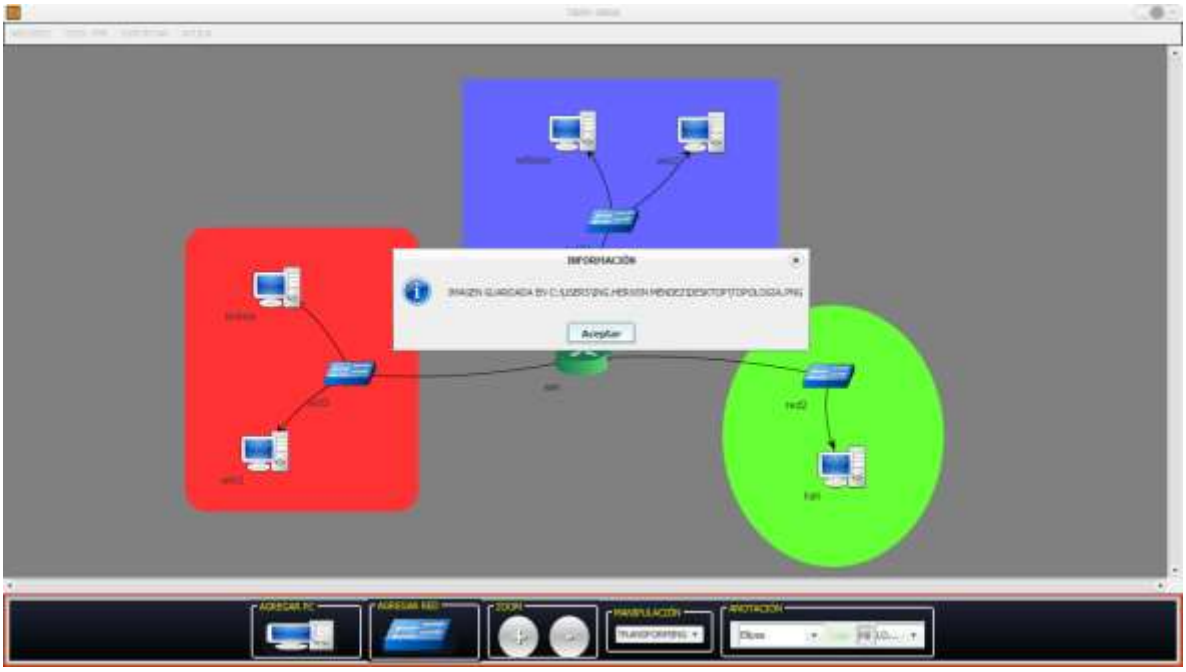



**Ilustración 21.Confirmación de la ruta donde se almacenó la imagen**

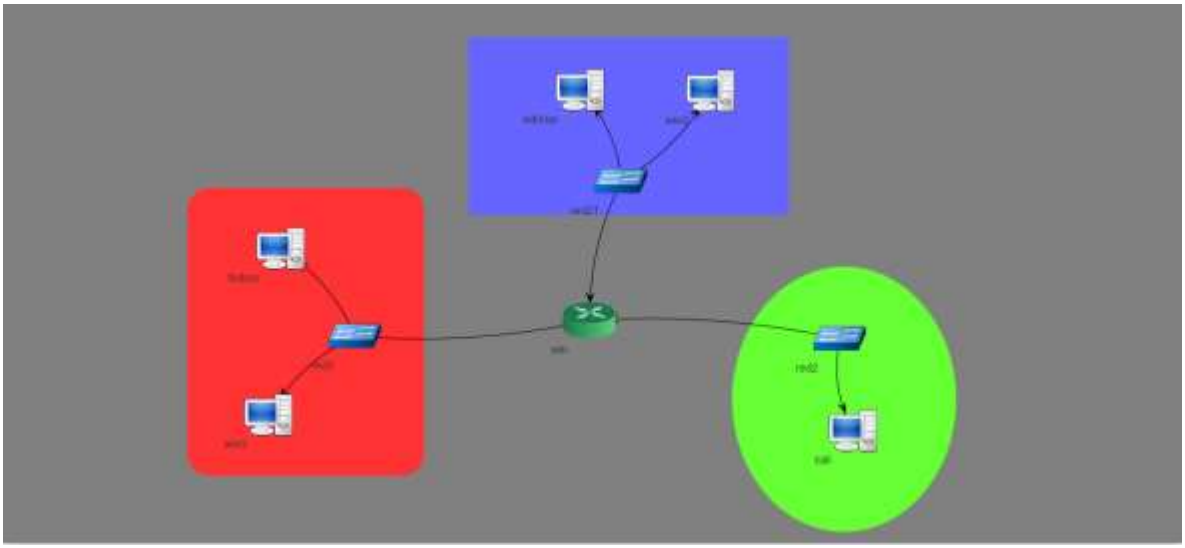


**Ilustración 22.Resultado del escenario de red en formato de imagen**

- ✓ **Formato PDF**

El usuario podrá hacer exportación de las máquinas virtuales que se encuentran en la topología en formato PDF utilizando la librería iText incorporada en la aplicación.

Tendrá que ir a **EXPORTAR ->Reporte PDF** aparecerá una ventana modal en la cual pedirá el nombre como desea guardar el documento y la ruta de almacenamiento.

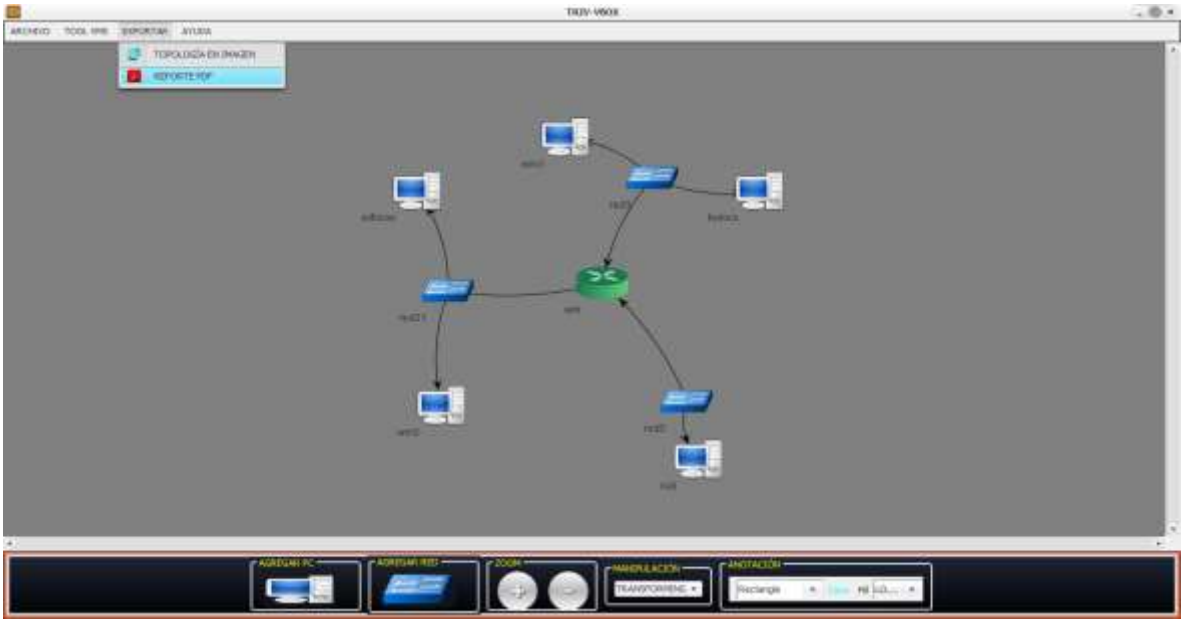


**Ilustración 23.Selección del ítem para generar reporte de la topología.**

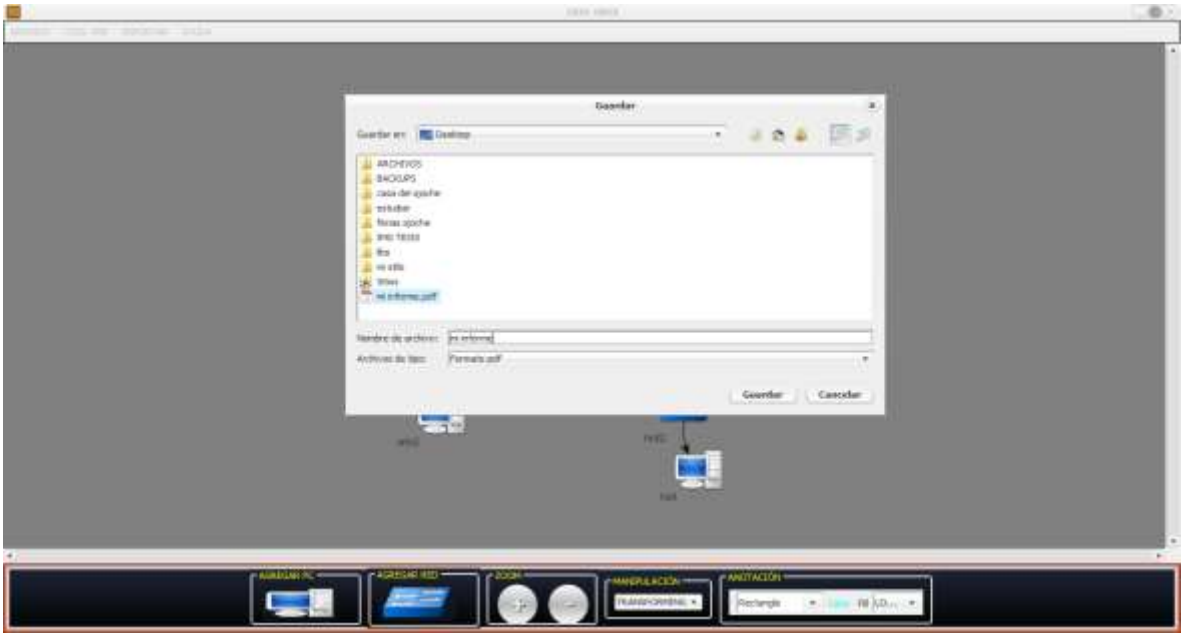


**Ilustración 24.Inserción del nombre y la ruta donde se desea almacenar.**

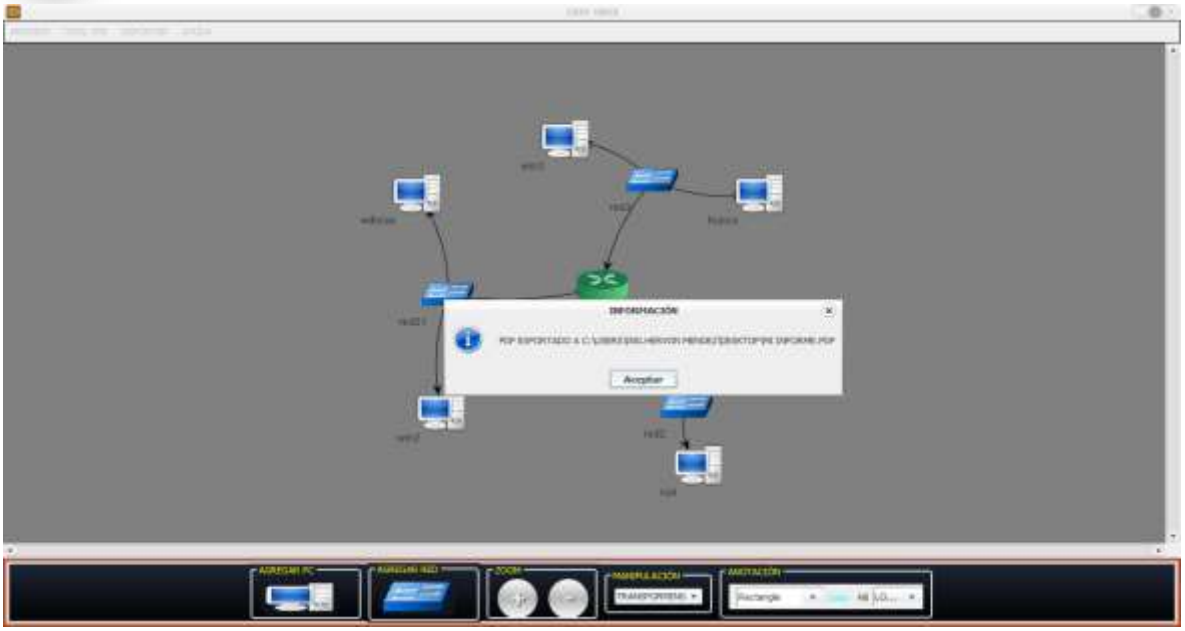


**Ilustración 25.Confirmación de la ruta donde se almacenar el reporte.**

## LEYENDA DE LAS MAQUINAS VIRTUALES

**TRIV-VBOX es una aplicación que facilita a los usuarios la visualización y manejo de las interfaces de las maquinas creadas con la herramienta de virtualización VirtualBox.**

| Máquina Virtual fedora | |
|---|---|
| SISTEMA OPERATIVO | Windows31 |
| MEMORIA RAM | 200 MB |
| ADAPTADOR #1 | |
| TIPO DE RED | NAT |
| DIR MAC: | 080027197FDB |
| HABILITADO | true |
| CABLE | true |
| ADAPTADOR #2 | |
| TIPO DE RED | Red Interna |
| NOMBRE | red3 |
| DIR MAC: | 0800275E5C57 |
| HABILITADO | true |
| CABLE | true |

| Máquina Virtual kali | |
|---|---|
| SISTEMA OPERATIVO | Linux22 |
| MEMORIA RAM | 200 MB |
| ADAPTADOR #1 | |
| TIPO DE RED | Red Interna |
| NOMBRE | red2 |
| DIR MAC: | 080027EC5221 |
| HABILITADO | true |
| CABLE | true |

| Máquina Virtual wifislax | |
|---|---|
| SISTEMA OPERATIVO | Linux22 |
| MEMORIA RAM | 200 MB |
| ADAPTADOR #1 | |
| TIPO DE RED | NAT |
| DIR MAC: | 080027039822 |
| HABILITADO | true |
| CABLE | true |
| ADAPTADOR #2 | |
| TIPO DE RED | Red Interna |

**Ilustración 26.Contenido del reporte "Mi informe.pdf".**

# 5. ASPECTOS FINALES

## 5.1 Conclusiones

Se logró desarrollar una aplicación multiplataforma capaz de generar escenarios de red a partir de la interconexión de las máquinas virtuales en el programa virtualizador VirtualBox, facilitando así el conocimiento y estructura que posee dichas máquinas siendo capaz de interactuar de manera rápida y sencilla, modificando aspectos referentes a las interfaces de red que posean cumpliendo con las metas y objetivos planteados desde el inicio del proyecto.

Esta aplicación fue desarrollada en el lenguaje de programación **Java** que nos brinda que la aplicación pueda ser ejecutada bajo distintas plataformas **(Windows, Linux, Mac),** también se utilizaron librerías externas como **JESPXML** para la interactividad con archivos de configuración que posee cada una de las máquinas virtuales que se crean, **JUNG** para la representación del escenario de red a partir de iconos y grafos dirigidos, **EDISONCORSX** para mejoras en la interfaz de la aplicación, **ITEXT** para generar reportes en pdf. Utilizando este conjunto de herramientas como un solo, nos permitieron alcanzar el fin deseado y obteniendo como resultado una aplicación capaz de gestionar escenarios de red, destinada a estudiantes y maestros del departamento de informática que requieran utilizarla.

## 5.2 Recomendaciones

### Recomendaciones para el uso de la aplicación:

- ✓ En caso de Windows, agregar a la variable de entorno la ruta de la carpeta de instalación de VirtualBox para poder realizar las operaciones (ejecutar, pausar, apagar) desde la aplicación.

### Recomendaciones para mejoras futuras:

- ✓ Permitir que la aplicación pueda ser utilizada con vmware.
- ✓ Incorporar que toda la aplicación interactúe por medio de comandos y no por medio de ficheros.
- ✓ Cambiar icono de las máquinas virtuales según el sistema operativo en el que se hayan creado.

### Recomendaciones para el departamento:

- ✓ Actualizar pensum a fin de proporcionarle a los estudiantes los lenguajes que utilizan en la actualidad las empresas, optimizando así la inserción e interacción en el entorno laboral como profesionales en el área de informática.

- ✓ Proponer intercambios de experiencias con otras universidades o facultades de ciencias y tecnologías para conocer de esta manera los avances en prácticas y desarrollos de otros colegas y estudiantes.

# 6. BIBLIOGRAFÍA

Boey YB, O. J. (s.f.). *Netsight*. Recuperado el 20 de 12 de 2015, de http://jung.sourceforge.net/netsight

Coronel, E. (s.f.). *Recopilación de interfaces y componentes de swing*. Recuperado el 8 de 2 de 2016, de https://sourceforge.net/projects/edisoncorsx/

Garron, G. (s.f.). *Controla VirtualBox desde la consola*. Recuperado el 15 de 3 de 2016, de https://www.garron.me/es/gnu-linux/controla-maquinas-virtuales-virtualbox.html

José Guerrero Peñafiel, A. T. (s.f.). *Virtualización de Servidores con herramientas de software libre*. Recuperado el 21 de 2 de 2016, de http://cdjbv.ucuenca.edu.ec/ebooks/ts167.pdf

Joshua O'Madadhain, S. W.-B. (s.f.). *Analysis and Visualization of Network Data using JUNG*. Recuperado el 5 de 1 de 2016, de http://www.jstatsoft.org/

Pinto, P. P. (s.f.). *JespXML [Leer y escribir un archivo XML desde Java]*. Recuperado el 14 de 1 de 2016, de http://pperezpinto.blogspot.com/2013/10/jespxml-v11-actualizado-leer-y-escribir.html

Sourceforge.Itext java PDF library.Recuperado el 20 de 09 de 2016,de https://sourceforge.net/projects/itext/

# 7. ANEXOS

## 7.1 Fichero de acceso para la obtención de los datos de cada máquina virtual

Para la obtención de los datos referente a cada máquina virtual y las interfaces de red que poseen, se accedió desde NetBeans IDE utilizando una librería específica llamada "**jespxml"** con la cual se puede leer y escribir formato **XML**, lenguaje de programación en el cual está escrito el fichero que aparece seleccionado en la siguiente imagen:

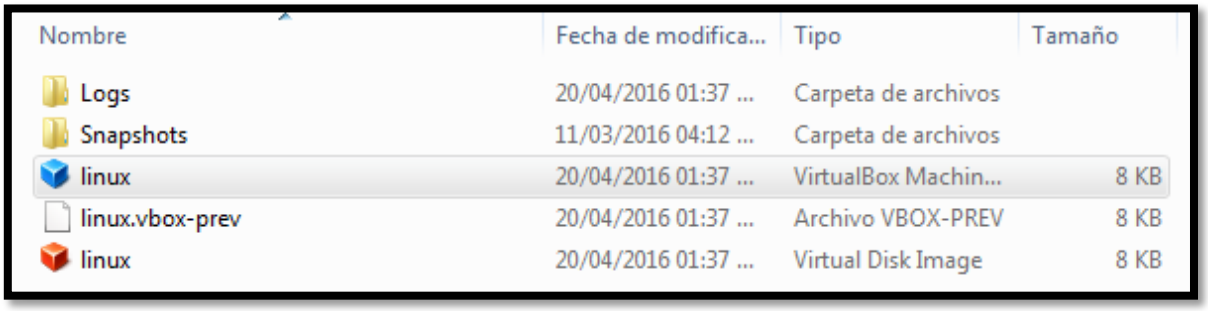



**Ilustración 27.Fichero de configuración de la máquina virtual**

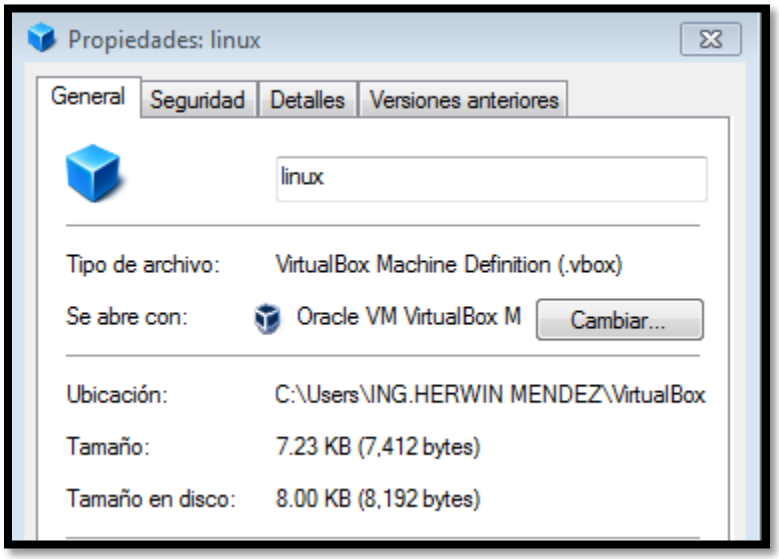



**Ilustración 28.Propiedades del fichero**

## 7.2 Clases java para almacenar los datos capturados del fichero

```
public class Equipo {
    private String nombre="";//nombre del equipo
    private String so="";//nombre del so
    private int memoria=0;//nombre de la memoria
    private List<Adaptador> lista_Adapt=null; //esta es una lista de tipo adaptador(contendra slot,tipo,valor)
    private String estado;

    public String getEstado() {
        return estado;
    }

    public void setEstado(String estado) {
        this.estado = estado;
    }

    public Equipo()
    {
        nombre="";
        so="";
        memoria=0;
        lista_Adapt=null;
        estado="";
    }

    public Equipo(String nombre,String so,int memoria,List<Adaptador> list_Adap)
    {
        this.nombre=nombre;
        this.so=so;
        this.memoria=memoria;
        lista_Adapt=list_Adap;
        estado = "";
    }

    //METODOS SET
    public void SetNombre(String nombre)
    {
        this.nombre=nombre;
    }

    public void SetSistOper(String so)
    {
        this.so=so;
    }
    public void SetMemoria(int memoria)
    {
        this.memoria=memoria;
    }

    public void SetList_Adapter(List<Adaptador> list_Adap)
    {
        lista_Adapt=list_Adap;
    }

    //METODOS GET
    public String GetNombre()
    {
        return this.nombre;
    }

    public String GetSistOper()
    {
        return this.so;
    }

    public int GetMemoria()
    {
        return this.memoria;
    }

    public List<Adaptador> GetList_Adapter()
    {
        return lista_Adapt;
    }

    @Override
    public String toString()
    {
        return "\n>>NOMBRE: "+GetNombre()+"\n>>SISTEMA OPERATIVO: "+GetSistOper()+"\n>>MEMORIA RAM: "+GetMemoria();
    }

}
```

La clase **Equipo** contendrá todos los datos de cada máquina virtual (Nombre, SO, Memoria, estado, Lista de adaptadores).

Cabe recalcar que se creará un ArrayList de tipo Adaptador, es una clase que contiene toda la información de cada adaptador y que se presentará en la siguiente página.

**Ilustración 29.Código fuente de la clase Equipo**

```java
public class Adaptador {
    private int slot=0;//este es el numero del slot
    private String tipo="";//Este es el tipo de Adaptador ya sea(InternalNetwork(red interna),NAT,BridgedInterface(puete),HostOnlyInterface(solo anfitrion))
    private String valor="";//Es el nombre de la red. por ejemplo <InternalNetwork name="red 1"/> quiere decir que la variable valor tendra "red1"
    private String Habilitado="";//si esta habilitado o deshabilitado
    private String MAC="";//Direccion mac de la maquina virtual
    private String cable="";//Estado del conexion del cable (true o false)

    public Adaptador()
    {
        this.slot=0;
        this.tipo="";
        this.valor="";
        this.Habilitado="";
        this.MAC="";
        this.cable="";
    }

    public Adaptador(int slot,String tipo,String valor,String Habilitado,String MAC,String cable)
    {
        this.slot=slot;
        this.tipo=tipo;
        this.valor=valor;
        this.Habilitado=Habilitado;
        this.MAC=MAC;
        this.cable=cable;
    }

    //METODOS SET
    public void SetSlot(int slot)
    {
        this.slot=slot;
    }

    public void SetTipo(String tipo)
    {
        this.tipo=tipo;
    }

    public void SetValor(String valor)
    {
        this.valor=valor;
    }


     public void setHabilitado(String Habilitado) {
         this.Habilitado = Habilitado;
     }

     public void setMAC(String MAC) {
         this.MAC = MAC;
     }

     public void setCable(String cable) {
         this.cable = cable;
     }

    //METODOS GET
    public int GetSlot()
    {
        return this.slot;
    }

    public String GetTipo()
    {
        return this.tipo;
    }

    public String GetValor()
    {
        return this.valor;
    }

     public String getHabilitado() {
         return Habilitado;
     }

     public String getMAC() {
         return MAC;
     }

     public String getCable() {
         return cable;
     }
```

```
    @Override
    public String toString()
    {
        String hab="";
        String cab="";
        String red="";

        if(getHabilitado().equalsIgnoreCase("true")){hab="Habilitado";}else{hab="Deshabilitado";}
        if(getCable().equalsIgnoreCase("true")){cab="Conectado";}else{cab="Desconectado";}

        switch(GetTipo())
        {
            case "InternalNetwork":
                    red=GetTipo()+": "+GetValor();
                     break;
            case "NAT":
                    red=GetTipo();
                     break;
        }

        String parrafo="<pre>Adaptador #"+(GetSlot()+1)+"   "+red+"    "+hab+"   "+cab+"</pre>";

        return parrafo;
    }
}
```

**Ilustración 30.Codigo fuente de la clase Adaptador**