GRID CON PARÁMETROS DTN
CASO DE ESTUDIO: COMPUTADORES PARA EDUCAR – UNIVERSIDAD DEL CAUCA

Anexos

Ing. Juan Carlos Arteaga Córdoba

Director: Msc. Héctor Fabio Jaramillo Ordóñez

Universidad Del Cauca
Instituto de Postgrados en Electrónica y Telecomunicaciones
Facultad de Electrónica Y Telecomunicaciones
Departamento De Telemática
Grupo de Ingeniería Telemática
Línea de Investigación en Servicios Avanzados de Telecomunicaciones
Popayán, Julio de 2009

Tabla de Contenido

Anexo A: Instalación de openMosix

En este anexo si indica la forma de instalar el software empleado en la implementación de la capa cluster. La implementación de ésta capa se realiza utilizando el concepto de múltiples máquinas físicas que contribuyen con su potencia de procesamiento y se comportan como un solo sistema denominado *Cluster* SSI o Imagen Simple de Sistema. En otras palabras un usuario que esté conectado al *Cluster* tendrá la sensación de estar trabajando en un equipo de buenas capacidades. En esta capa, para el desarrollo del *Cluster*, se eligió un sistema conocido cómo openMosix el cual trabaja como una extensión del *kernel* de Linux y permite el perfecto agrupamiento en *Cluster*s y balanceo de carga de potencia de procesamiento entre sistemas interconectados en una red [1].

El proceso de dicha instalación se detalla a continuación.

Construcción de un *Cluster* con OpenMosix.

Requerimientos:

- Dos o más computadores, rápidos o lentos no importa su configuración hardware, lo único que requieren son tarjetas de red y estar conectados a la red.

- Un LiveCD de Cluster Knoppix.

Instalación de OpenMosix.

OpenMosix consiste de dos partes, la primera parte corresponde al modificador de *kernel* el cual modifica el *kernel* de Linux para adaptarlo al funcionamiento dentro de un cluster, es decir habilitarlo para compartir procesamiento y además poder enviar ciertos procesos cuando el actual procesador llega a un límite de procesamiento predefinido, la segunda parte corresponde a las herramientas de usuario que permiten administrar y controlar el cluster [2].

Antes de instalar openMosix se debe tener en cuenta que este hace una modificación al *kernel* del Sistema operativo, por esta razón es muy recomendable que la versión de *kernel* de openMosix coincida con la versión de *kernel* del sistema operativo, por esta

razón generalmente es mucho más sencillo instalar openMosix desde un LiveCD que ya trae el openMosix adaptado al *kernel* del sistema operativo utilizado [3]

Por lo anterior en este documento se mostrarán tres formas de tener openMosix, instalando desde las fuentes, instalado desde archivos RPM y desde un LiveCD de Knoppix.

1. **Instalar desde los fuentes y modificación del *kernel*:**
   En este procedimiento lo que se hace es modificar el *kernel* del sistema operativo sobre el que se va a instalar openMosix. Para fines de este documento se utilizará la versión 2.4.24 del *kernel* [4].

   Para descargar openMosix se lo puede hacer de la página principal del proyecto, que aunque ya está cerrado, aún posee el link de descarga: http://openmosix.sourceforge.net.

   Luego se debe descargar una versión del *kernel*, para este documento se utilizó la versión que se encuentra en la siguiente dirección: http://www.kernel.org/pub/linux/kernel/v2.4/linux-2.4.24.tar.bz2.

   a. Una vez se tenga las dos versiones del *kernel*, es decir la del sistema operativo y la de openMosix se debe descomprimir y almacenar en el directorio fuente del *kernel* en este caso lo haremos en:

      ```
      /usr/src/linux-2.4.24.
      ```

   b. Copiar el modificador de *kernel* de openMosix al directorio fuente del *kernel* y ejecútelo, para esto se siguen los siguientes comandos:

      ```
      mv /root/openMosix-2.4.24.gz /usr/src/linux-2.4.24

      cd /usr/src/linux-2.4.24

      zcat openMosix-2.4.24.gz | patch -Np1
      ```

      Si se tuvo algún problema en este paso por favor refiérase al siguiente enlace: http://howto.x-tend.be/openMosix-HOWTO/x332.html

   c. A continuación se debe configurar las fuentes del *kernel* para esto se deben ejecutar los siguientes comandos:

      ```
      make config
      make menuconfig
      make xconfig
      ```

   d. Una vez que se tenga la pantalla de configuración del *kernel* se puede habilitar las siguientes opciones de openMosix en el *kernel*:

```
CONFIG_MOSIX=y

# CONFIG_MOSIX_TOPOLOGY is not set

CONFIG_MOSIX_UDB=y

# CONFIG_MOSIX_DEBUG is not set

# CONFIG_MOSIX_CHEAT_MIGSELF is not set

CONFIG_MOSIX_WEEEEEEEEE=y

CONFIG_MOSIX_DIAG=y

CONFIG_MOSIX_SECUREPORTS=y

CONFIG_MOSIX_DISCLOSURE=3

CONFIG_QKERNEL_EXT=y

CONFIG_MOSIX_DFSA=y

CONFIG_MOSIX_FS=y

CONFIG_MOSIX_PIPE_EXCEPTIONS=y

CONFIG_QOS_JID=y
```

e. Finalmente se finaliza todo compilando el *kernel* con el siguiente comando:

```
make dep bzImage modules modules_install
```

f. Ahora reinicie el equipo y si todo salió bien se tendrá la opción de seleccionar el *kernel* que se instaló y se puede iniciar desde él.

2. Instalar openMosix en Debian

Para instalar openMosix en Debian se lo puede hacer desde un archivo RPM. Si se tiene una distribución basada en RPM, se pude obtener una imagen del *kernel* precompilada con openMosix habilitado desde la página del proyecto de openMosix.

Este es un método moderadamente sencillo de instalación de openMosix. Este método puede trabajar con Red Hat, SUSE y otras distribuciones de Linux. Para instalar se requieren dos archivos RPM [4]

a) openmosix-kernel

b) openmosix-toools

Sencillamente se debe ejecutar el comando:

```
rpm –Uvh openmosix*.rpm
```

Pero para el caso específico de Debían es mejor utilizar el comando `apt-get` para instalar openmosixview, el cual permitirá obtener una interfaz gráfica de usuario (GUI) que permitirá administrar el cluster.

El procedimiento básico es el siguiente:

1. Encuentre los paquetes:

```
cd /usr/src
apt-get install kernel-source-2.4.24 kernel-package \
openmosix kernel-patch-openmosix
```

2. Descomprímalos y cree los *links*:

```
tar vxjf kernel-source-2.4.24.tar.bz2
ln –s /usr/src/kernel-source-2.4.24 /usr/src/linux
```

3. Ejecute el *patch*:

```
cd /usr/src/Linux
../kernel-patches/i386/apply/openmosix
```

4. Instale el *kernel:*

```
make menuconfig
make-kpkg kenrel-image modules_image
cd ..
dpkg –I kernel-image-*-openmosix-*.deb
```

5. Después se puede utilizar `apt-get` para instalar openmosixview GUI para administrar el cluster:

```
apt-get install openmosixview
```

3. Utilizando el LiveCD de Knoppix (ClusterKanoppix)

En este caso simplemente se coloca el CD en la unidad y el arranca solo y openMosix ya está preinstalado y preconfigurado solo se debe configurar al gusto y listo, a esto se lo conoce como ClusterKnoppix.

La gran ventaja de utilizar este tercer método es que simplemente iniciando el sistema desde el LiveCD de knoppix este se adiciona automáticamente al C*luster*. No se requiere tener instalado el sistema operativo en el disco duro. Esto lo hace muy útil para instalar un *Cluster*, en el caso de que el tiempo sea un factor determinante. Otra característica que vuelve atractivo este método es que no se requiere quemar 10 copiad del CD para hacer un sistema de 20 clusters. Simplemente es necesario iniciar el sistema con el CD y luego ejecutar el comando.

```
knoppix-terminalopenmosixserver
```

Este comando corresponde a una colección de *scripts* que permiten iniciar servicios como dhcp, tftp, nfs en una máquina que esté ejecutando Knoppix, de tal manera que se comparte por red el *kernel*, un initrd generado dinámicamente y la imagen compresa. Es decir, este comando permite configurar un *terminal server* con soporte para *Cluster*.

Otra forma de iniciar el sistema es iniciar por red, en este caso se deben tener tarjetas de red compatibles con PXE, en este caso el sistema descargará la imagen del *kernel* y ejecutará el Cluster Knoppix [4].

Luego de ejecutar este comando para el caso del servidor aparecen tres opciones:

- **Setup:** esta opción permite configurar o reconfigurar por primera vez el servidor y reiniciarlo.
- **Start:** esta opción permite iniciar el servidor.
- **Stop:** esta opción detiene el servidor.

Si se selecciona la opción de configuración, debe aparecer un cuadro de diálogo que muestre las tarjetas de red disponibles, seleccione la tarjeta que vaya a utilizar y haga clic en ok.

Iniciando el cluster

En este se utilizarán dos máquinas con openMosix instalado. A continuación se ilustra cómo adicionar sistemas al *Cluster* y cómo hacerlos trabajar juntos. openMosix tiene dos formas de hacer esto:

1. Auto descubrimiento de nodos del cluster

   OpenMosix incluye un demonio llamado *omdiscd* el cual identifica los otros nodos openMosix en la red utilizando paquetes *multicast*. Gracias a esto no se requiere de la configuración individual de cada nodo. Esta es una simple manera de crear un cluster que inicie una máquina y se asegure que está en la red. Después de realizar estos pasos se debe tener la posibilidad de descubrir los nodos que existen en el cluster de manera automática.

   Asegúrese de tener la red configurada de manera apropiada. Por ejemplo si se asigno una IP 192.168.1.10 la primera interfaz de *Ethernet* y su *Gateway* por defecto es 192.168.1.1 se debería hacer algo cómo:

   ```
   ifconfig eth0 192.168.1.1 netmask 255.255.255.0 broadcast 192.168.1.255 up
   ```

   (Este comando configura la interfaz *Ethernet* de la red)

   ```
   route add default gw 192.168.1.1
   ```

   (Adiciona la *Gateway* por defecto)

Para observar si el demonio de auto-descubrimiento se inició automáticamente en el arranque es necesario el siguiente comando:

```
Ps aux| grep 'omdiscd'
```

Con este comando se observaría si el demonio *omdiscd* se está ejecutando en el sistema.

Si no es así se debe iniciar manualmente escribiendo *omdiscd*. Si se desea ver los nodos que se van adicionando a la red, se debe ejecutar *"omdiscd –n"*. Esto ayudará a solucionar los problemas de auto descubrimiento.

2. El archivo de configuración /etc/openmosix.map

Si nos e desea realizar un auto descubrimiento, se lo realizaría manualmente mediante el archivo openmosix.map. Este archivo contiene una lista de los nodos en el cluster, y tiene que ser el mismo para todos los nodos dentro del *Cluster*. Su sintaxis es muy simple, ésta tiene tres campos: ID del nodo, Dirección IP y Número.

- El ID del Nodo es un número único que identifica al nodo
- Dirección IP. La dirección IP del nodo.
- El número especifica cuantos hay en el rango después de la IP.

Por ejemplo si se tienen los siguientes nodos:

```
192.168.1.10

192.168.1.11

192.168.1.12

192.168.1.50
```

El archivo se debería ver así:

```
1 192.168.1.1 2

2 192.168.1.1 1
```

Aunque en realidad se podría tener especificado manualmente las direcciones OP 192.168.1.11 y 192.168.1.12, pero utilizando el campo numero, openMosix cuenta los últimos octetos de la IP y te evita el problema de hacer entradas individuales.

Una vez que se tenga lista la configuración se puede controlar openMosix utilizando el *script* `init.d` que debería estar instalado. De lo contrario se debe descargar e instalar, haciéndolo ejecutable y copiando el archivo `init.d` a un directorio, así:

```
mv ./openmosix /etc/init.d
chmod 755 /etc/init.d/openmosix
```

Ahora se deber iniciar, detener y reiniciar openMosix con los siguientes comando:

```
/etc/init.d/openmosix start
/etc/init.d/openmosix stop
```

```
/etc/init.d/openmosix restart
```

Finalmente si todo está bien se puede probar el Nuevo cluster.

3. Probando el Cluster.

   Existe una pequeña aplicación que se puede utilizar para monitorear la carga del cluster. Escriba `mosmon` y presione *enter*. Se debería tener una ventana como la siguiente Figura A1:

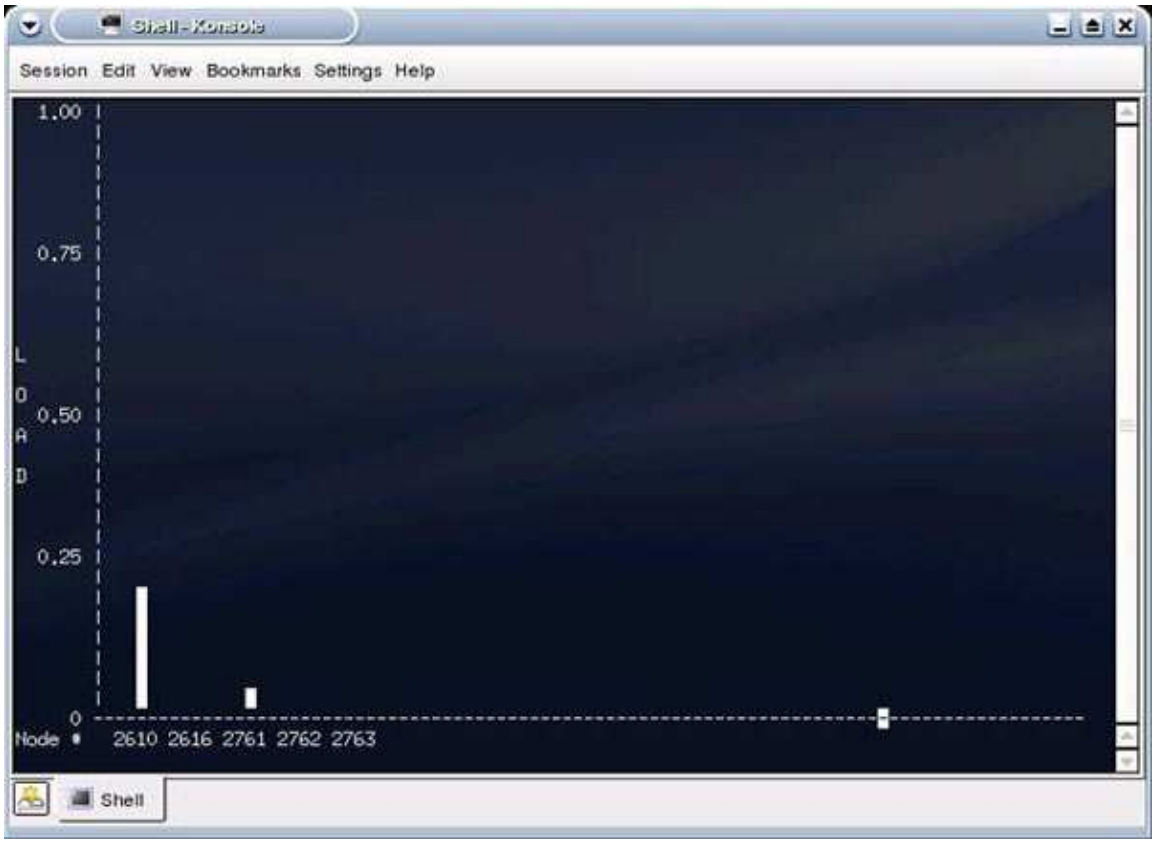

Figura A1: Visualización de los nodos y su respectiva carga en openMosix.

Hagamos el siguiente ejemplo ejecute `mosmon` en un Terminal de comandos y presione control+alt+F1 y luego cambiamos a otro Terminal presionando control+alt+F2, esto permitirá ejecutar un ciclo anidado y utilizar algo de poder de procesamiento para poder probar el cluster creado. Si todo salió bien se debería mirar que la carga en `mosmon` salta de un nodo al otro y luego migra a los otros nodos.

El comando que se requiere ejecutar para el ciclo es el siguiente.

```
awk 'BEGIN' {for(i=0;i<10000;i++)for(j=0;j<10000;j++);}'
```

Si se desea se pueden iniciar múltiples procesos *awk* ejecutándolos en *background*. Solo se debe adicionar el carácter '&' a la línea de comando y ejecutarlo muchas veces.

Regresemos al comando `mosmon` presionando control+alt=F1, se debería mirar un crecimiento en la carga del nodo actual y luego una lenta distribución hacia las otras máquinas del cluster como en la siguiente figura A2:

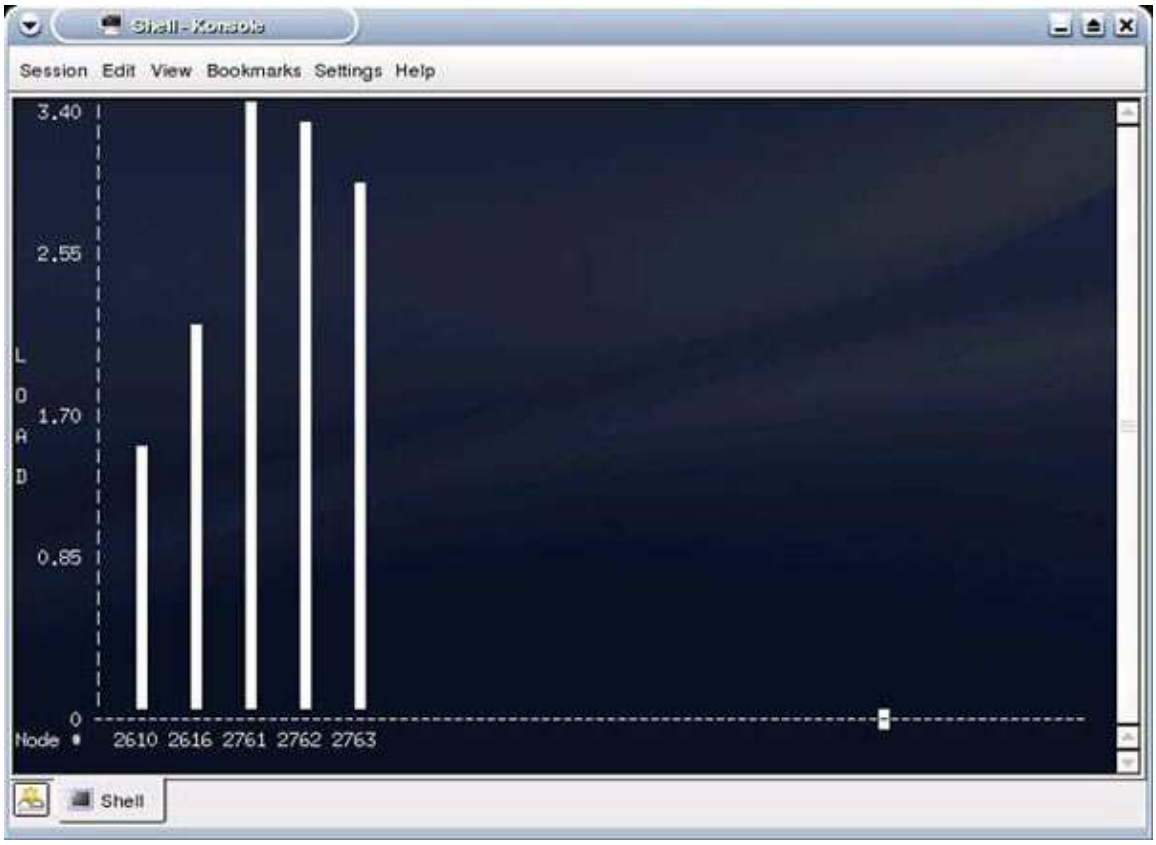

Figura A2: Visualización de los nodos y su respectiva carga en openMosix, esta vez con mucha carga de trabajo en etre ellos.

Si todo está bien tendrías la imagen anterior. Si se desea probar como sería todo sin el cluster pues se puede detener openMosix con el comando:

```
/etc/init.d/openmosix stop
```

Luego ejecutar el siguiente *script*:

```
#!/bin/sh

awk 'BEGIN {for(i=0;i<10000;i++)for(j=0;j<10000;j++);}'

date
```

4. Controlar el Cluster

El grupo openMosix ha desarrollado varias formas de controlar el cluster, esto se puede hacer a través de línea de comandos o a través de herramientas GUI (Interfaz gráfica de usuario) basadas en X.

Así entonces se mostrará las principales herramientas para controlar y monitorear:
En línea de comandos [4]:

- mosmon: muestra la carga en cada uno de los nodos, su velocidad, memoria, uso, entre otros. Presionando 'h' se podrá ver las opciones disponibles de este comando.
- mosctl: es un comando muy poderoso que permite controlar como el sistema se comporta dentro del cluster, algunas de las opciones interesantes son:
  - mostctl block: Detiene los procesos de otras personas que se estén ejecutando en nuestro sistema.
  - mostctl –block: El opuesto de los de arriba.
  - mostctl lstay: Detiene los procesos locales y los migra a otros nodos para el procesamiento.
  - mostctl nolstay: El opuesto del de arriba.
  - mostctl setspeed <number>: Fija la máxima velocidad de procesamiento con la que se contribuye. Por ejemplo el límite para un Pentium 3 de 1Ghz es 10000.
  - mostctl whois <node number>: Responde con la dirección IP de un nodo en particular.
  - mostctl expel: Expulsa cualquier proceso remoto y bloquea los nuevos que provengan de él.
  - mostctl bring: Trae de regreso cualquier proceso propio local que se haya migrado a otros nodos.
  - mostctl status <node number>: El cual muestra si un nodo está activo.
  - mosrun: Permite ejecutar un proceso controlando los procesos que se ejecutan sobre él.
  - mps: Es cómo el comando 'ps', el cual muestra una lista de procesos, pero además muestra cual es el nodo en el cual se está ejecutando.
  - migrate: este comando permite migrar manualmente a un proceso para cualquier nodo, la sintaxis es la siguiente: `migrate <pid> <node #>`. También se puede utilizar `migrate <pid> balance` para hacer, de manera automática, balanceo de carga sobre un proceso.
  - dsh: distributed Shell. Permite ejecutar un comando en todos lso nodos de manera simultánea. Por ejemplo: `dsh –a reboot` el cual reiniciará todos los nodos.

En el caso de la GUI, se puede iniciar con el comando `openmosixview`. Este permite mirar y administrar todos los nodos en el cluster. Además muestra el balanceo de carga eficientemente de un cluster en tiempo cercano al real. Se puede ver también cual es el total de velocidad y RAM que el cluster te provee, ver Figura A3.

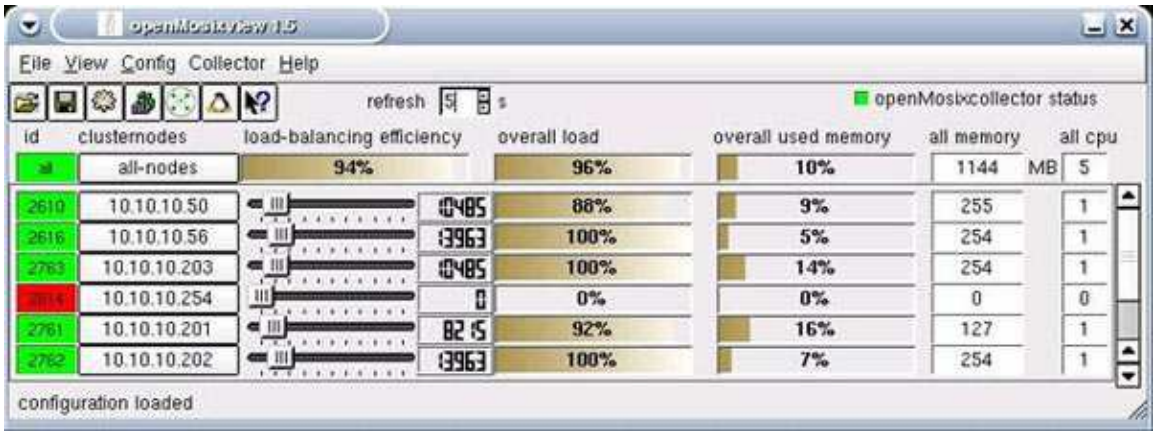

Figura A3: Visualización de los nodos y su respectiva carga en openMosixview 1.5.

Cabe anotar que los nodos que están en línea se representan en color verde y los que están fuera de línea en color rojo. Otra cosa muy importante que tiene openmosixview es la GUI para controlar la migración de procesos. Ver Figura A4.

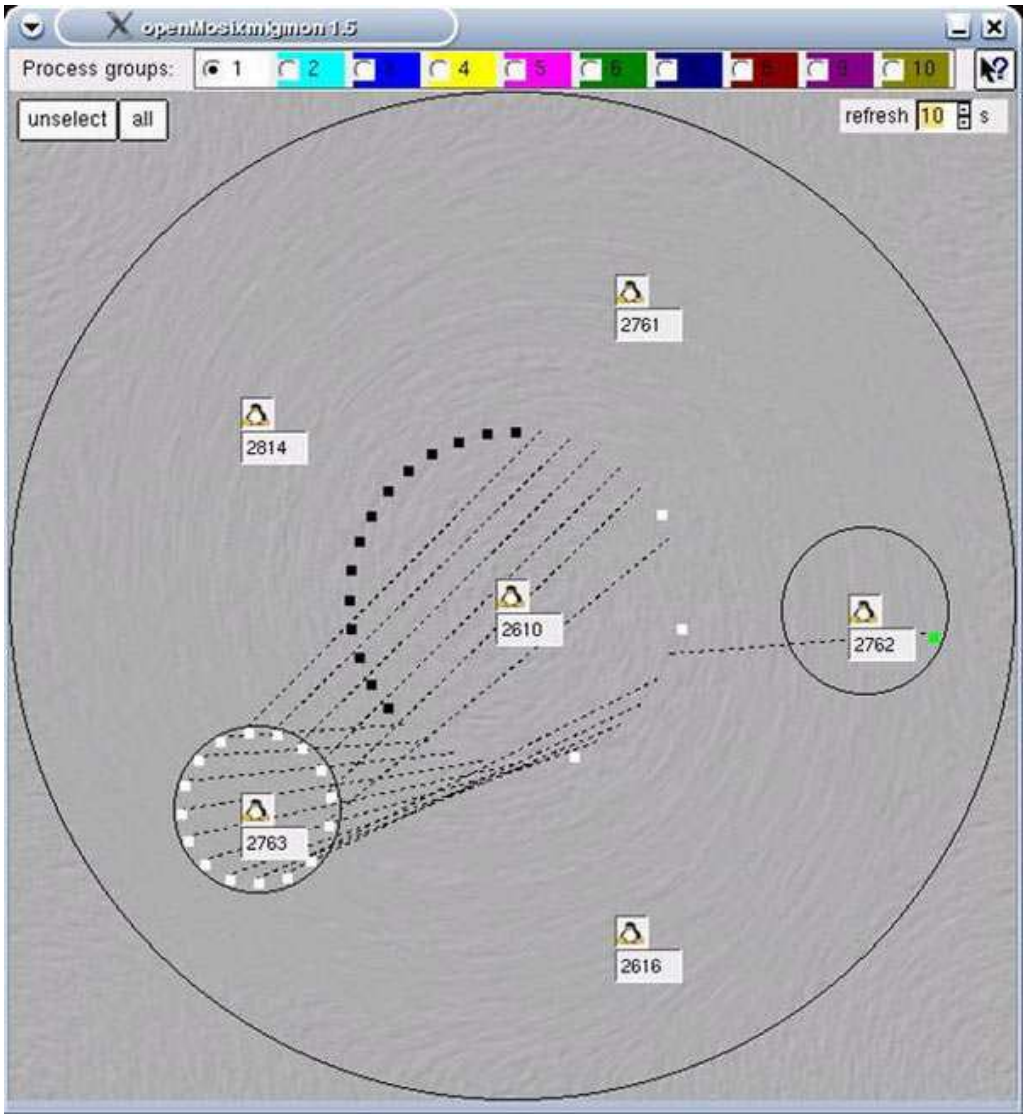

Figura A4: Visualización de los nodos y con la posibilidad de controlar la migración de procesos.

En el lugar central de la figura se puede observar el nodo actual (nodo local) y en los extremos, alrededor de él, los otros nodos del cluster. El anillo exterior representa los procesos ejecutándose en su caja local. Si se selecciona con el cursor cualquiera de ellos se puede ver el PID y el nombre del proceso. Cada vez que uno de sus procesos migra de un nodo a otro, se podrá observar cómo se separa y aparece en un nuevo nodo con una línea que se une al sistema local.

Además se puede controlar de manera manual el proceso de migración. Se puede arrastrar y soltar los procesos en otro nodo, incluso se pueden seleccionar muchos procesos y luego soltarlos juntos a otro nodo. Si se hace doble clic sobre un proceso ejecutándose en un nodo remoto, este regresará a su nodo inicial y se ejecutará de manera local.

Otro aspecto importante es que openmosix posee un monitor de procesos el cual muestra cada proceso y el nodo en el cual se está ejecutando.

Existe además un analizador de historia que muestra como se ha comportado la carga de procesos en un periodo de timpo. Esto permite mirar cómo se utilizó el *Cluster* en un punto de tiempo determinado ver Figura A5:

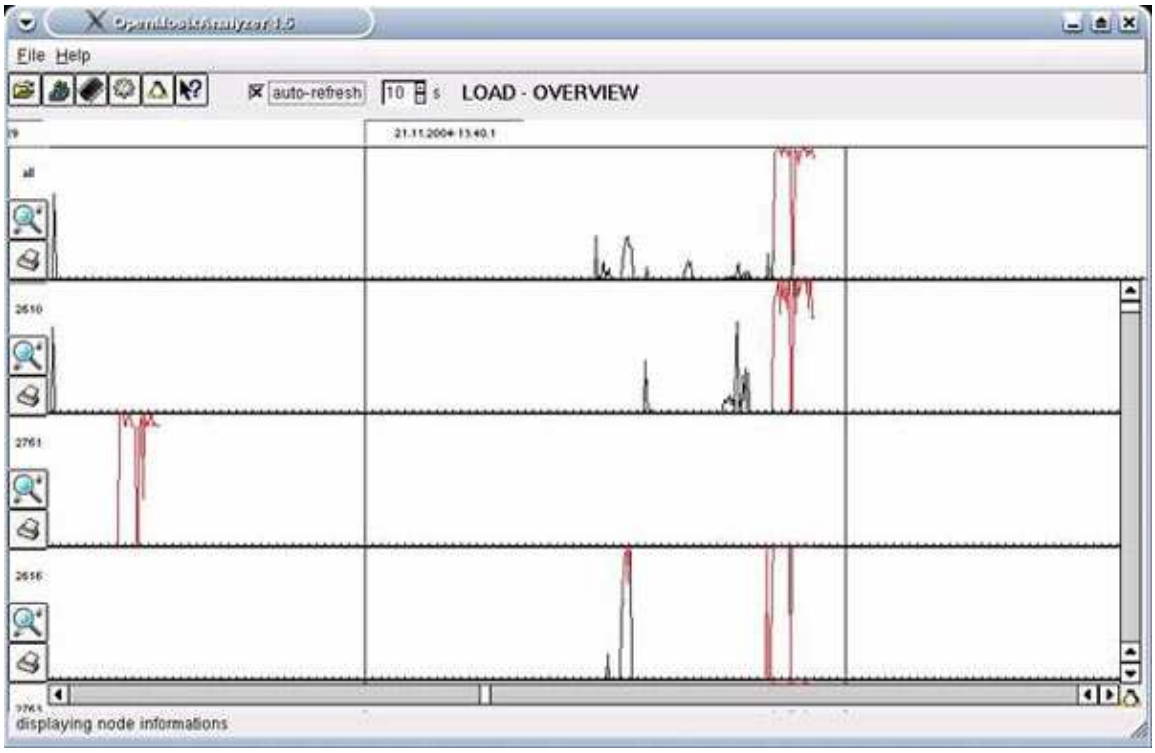

Figura A5: Analizador histórico de carga de opneMosixview.

Como se puede observar, las herramientas visuales presentadas en la GUI son muy poderosas, permiten una gran cantidad de funcionalidades que la línea de comandos puede hacer.

Referencias

[1] M. Catalán, “El manual para el *Clustering* con OpenMosix” Versión 1.0. Mikel a.ka.mc2 & Buytaert, 2004.

[2] Source forge, Inc. openMosix 1999 [En línea]. Disponible: http://sourceforge.net/projects/openmosix/ [Consulta: marzo, 2009].

[3] Joseph D. Sloan, “High Performance Linux *Cluster*s with OSCAR, Rocks, OpenMosix, and MPI,” O´Relley editors, 2004.

[4] Firewall.cx 2009 [En línea]. Disponible: http://www.firewall.cx/linux-openmosix-intro.php [Consulta: marzo, 2009].

Anexo B: Instalación de cosMos

En este anexo si indica la forma de instalar el software empleado en la implementación de la capa cluster desde un Terminal con un sistema operativo Windows, dicho software se denomina cosMos.

El desarrollo de este software comenzó como trabajo de desarrollo de una herramienta para el grupo de seguridad en la universidad de Macquarie en 2003 - un equipo que incluyó Rob Dartnell, este último de Ian y Ty Miller. La necesidad allí era demostrar la debilidad en seguridad de una aplicación particular [1].

La distribución CHAOS fue creada para llenar esta necesidad, y desarrollada por casi dos años bajo una Licencia General Pública para permitir que los miembros de la Comunidad de openMosix se beneficien de los alcances de la seguridad empleados alrededor del software de openMosix (la tecnología de agrupamiento que se agrega al núcleo de linux). Mejoras en la seguridad llevadas a cabo por el equipo incluyeron túneles IPSEC para todas las comunicaciones del cluster, filtración de paquetes de estado de alerta para cada nodo, una imagen pequeña del sistema operativo que permitió inicio PXE para memorias de PC remotos, la creación de cluster sin manipulación, entre otros [1].

A mediados del 2004 CHAOS fue adaptado a la estructura de Colinux, permitiendo a OpenMosix correr como un nodo sobre un PC Windows por primera vez. La versión de CHAOS creada para coLinux fue adaptada denominada CosMos (Chaos-OS on Microsoft-OS) y fue lanzada bajo Licencia General Pública – completa con un software instalador para Windows [1].

Construcción del cluster con CosMos / CHAOS

El cluster a construir debe contener un nodo maestro (de tipo ClusterKanoppix) en este caso se cuenta con un Cluster openMosix al cual se va a integrar diversos nodos con sistemas operativos Windows mediante cosMos.

El proceso de instalación y puesta a punto de esta herramienta es muy simple, como requisitos indispensables se debe tener los siguientes:

- Dos o más computadores, rápidos o lentos no importa su configuración hardware, lo único que requieren son tarjetas de red y estar conectados a la red.

- Un LiveCD de Cluster Knoppix, o ya contar con una maquina donde esté instalado openMosix, de hecho en este anexo se parte del hecho que ya se cuenta con una maquina previamente instalada una distribución de openMosix tal como se detallo en el Anexo 1.

- La distribución del software de cosMos – CHAOS, esta distribución puede encontrarse en el siguiente enlace:
  http://midnightcode.org/projects/chaos/code/setup-0.1.5.0.exe

Instalación de cosMos

Estando en Windows, se instala el archivo ejecutable setup-0.1.5.0.exe simplemente haciendo doble clic sobre dicho archivo.

Los pasos siguientes son la selección del tipo de adaptador que se instalará en el proceso, en este caso se seleccionó el adaptador TAP-Win32 Adapter. Los demás pasos son muy intuitivos por tanto son omitidos en este anexo.

Una vez instalada la herramienta se inicia el programa desde:

Inicio – Programas – cosMos – Power on, Iniciada la herramienta se debe observar una ventana como la de la figura B1

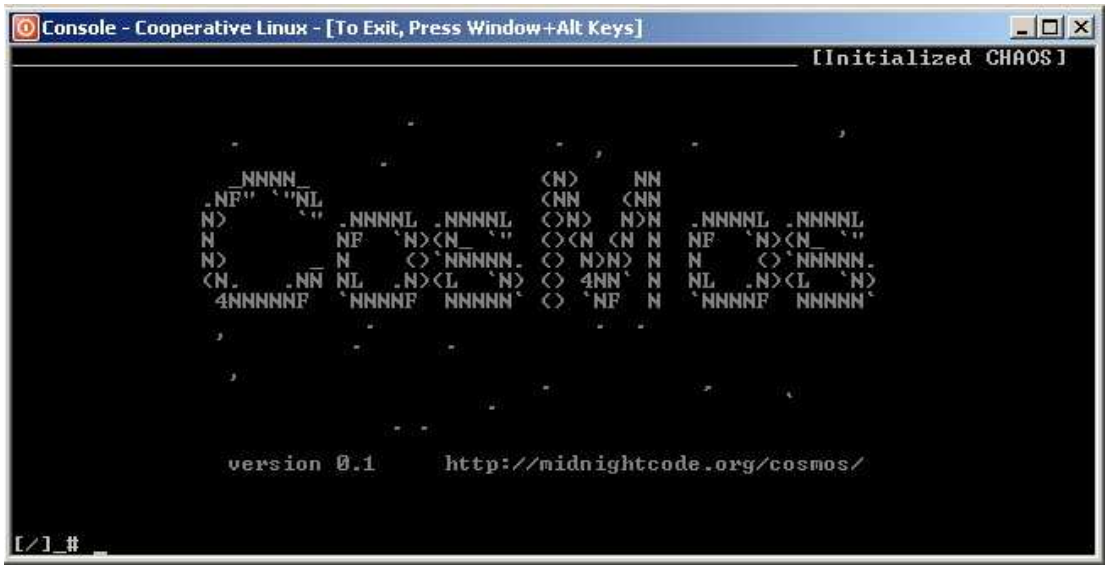

Figura B1: Consola de comandos de cosMos.

Seguidamente es indispensable realizar una simple configuración en las conexiones de red del equipo donde se este corriendo cosMos, este tipo de configuración nos permite realizar la conexión de la máquina virtual donde esta cosMos con el equipo y esta a su vez hacia el exterior donde se encuentra openMosix.

Para ello se debe ingresar a las conexiones de red desde:

Inicio – Configuración – Conexiones de red.

Una vez ubicados en dicha ventana observamos que se encuentran disponibles dos conexiones denominadas:

- Conexión de área local (n) (si existe más de una conexión de área local se la identificará con números desde 1 hasta n según el número que se tenga de conexiones instaladas)

- TAP-Win32 Adapter.

Para crear una correcta configuración de las conexiones de red hay que crear un puente de red entre este par de conexiones. Este proceso se realiza seleccionando con el puntero del Mouse estas conexiones y posteriormente dar clic derecho y seleccionar "conexiones de puente".

Una vez realizado el proceso se cebe tener una ventana de conexiones como la que se muestra en la figura B2.

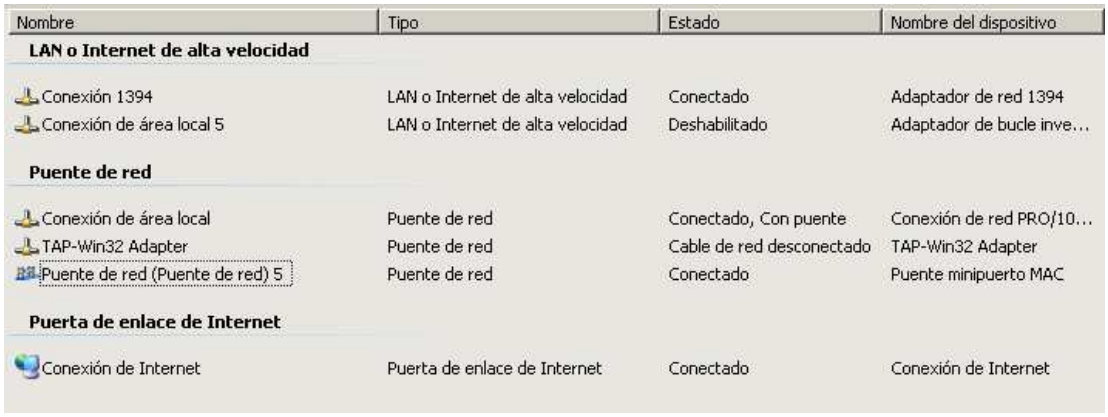

Figura B2: Conexiones de red, Puente de red entre la conexión de área local y el TAP-Win32 Adapter.

De ahora en adelante cosMos ya tiene la conexión habilitada para comunicarse con openMosix, por tanto es posible poder establecer dicha conexión con los siguientes comandos en consola [2]:

Mosmon provee la información de los nodos fundamentales unidos al Cluster ver figura B3.

```
mosmon
```

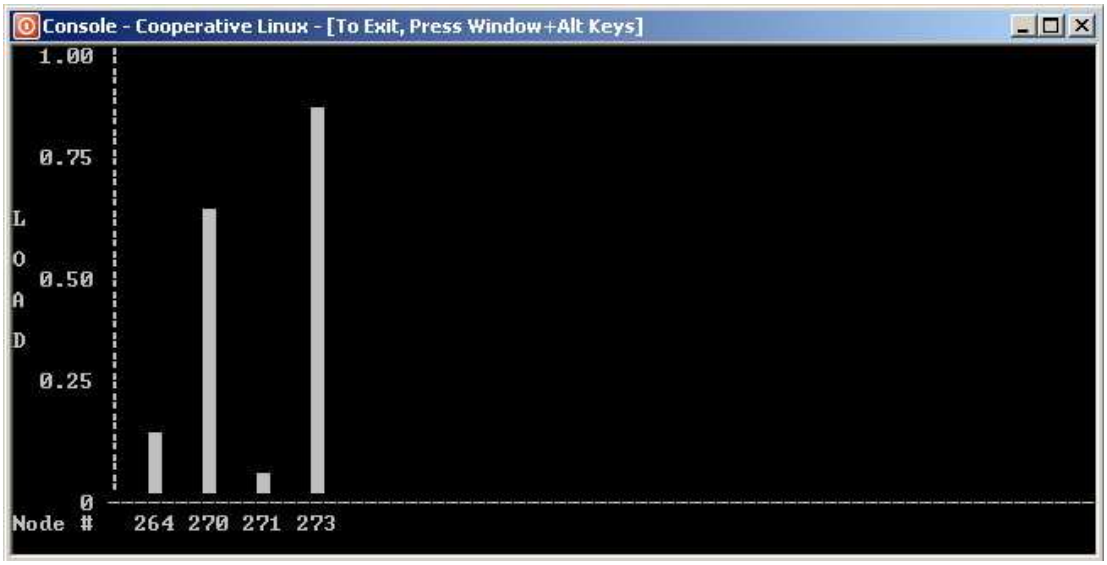

Figura B3: Visualización de los nodos y su respectiva carga en openMosix desde cosMos.

Mtopo es version de “top” de openMosix, el cual muestra los procesos en el cluster y que cantidad de recursos de memoria y CPU estan consumiendo, ver figura B4.

```
mtop
```

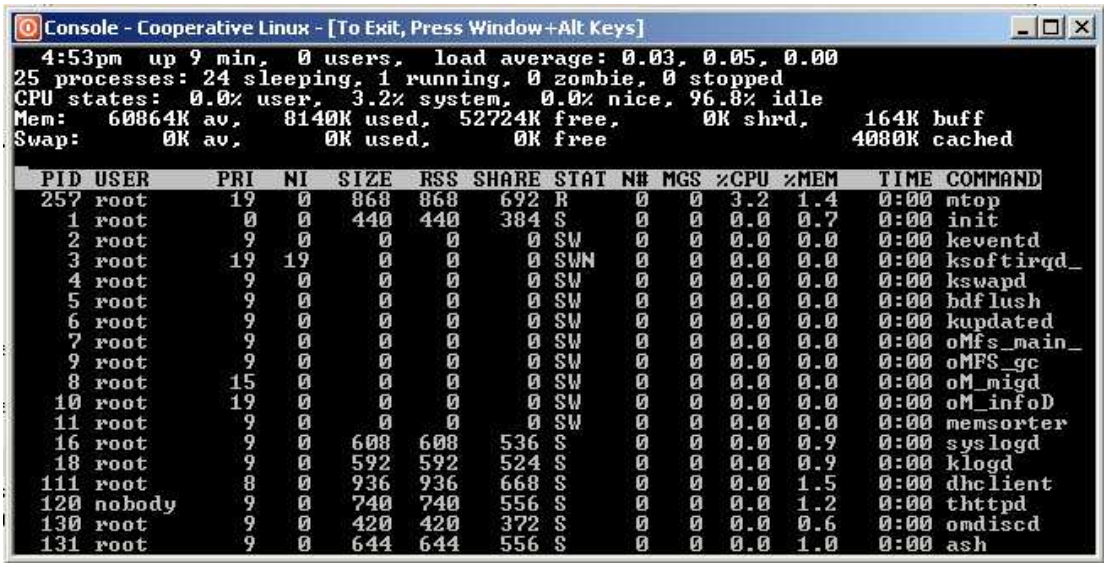

Figura B4: Visualización de los procesos en cluster y su repsectivo consumo de recursos.

Showmap, deposita en una de texto la base de datos de los nodos del cluster. Ver figura B5.

```
showmap
```

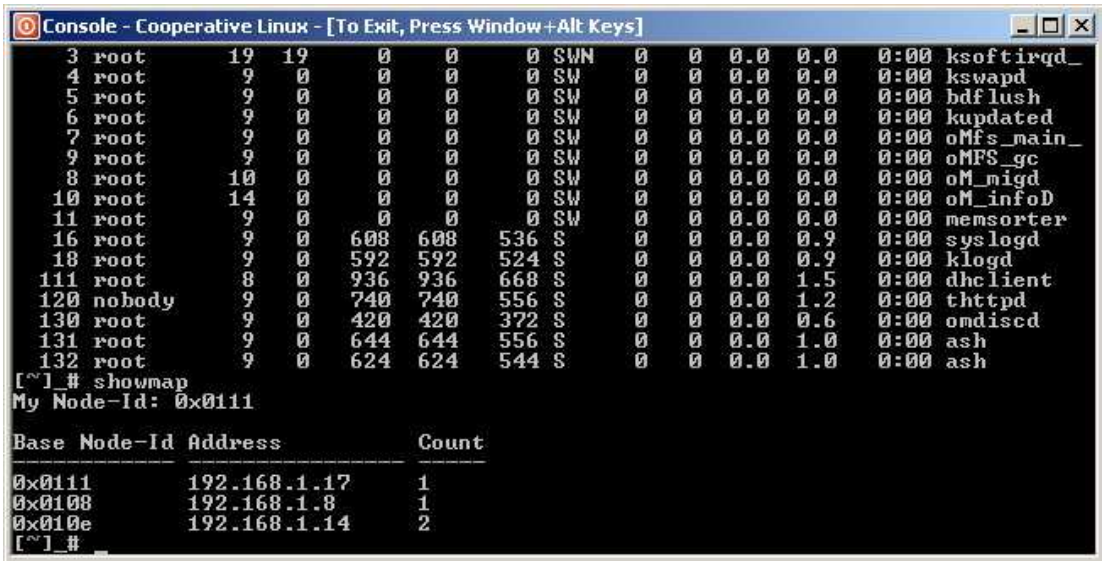

Figura B5: Información de los nodos del cluster mediante el comando showmap.

Referencias

[1] I. Latter, “Planet Series Project , Midnightcode,” Project Chaos / CosMos (*The Great Systems*) 2004 [En línea]. Disponible: http://midnightcode.org/projects/chaos/ [Consulta: marzo, 2009].

[2] I. Latter, “Heterogeneous Clusters; Running ClusterKnoppix as a master node to a CHAOS drone army,” howto - heterogenous clusters.doc-Release-1.1 (467) Macquarie University, 2003.

Anexo C: Instalación de coLinux

En este anexo se detallará como instalar coLinux en Windows XP, la imagen del sistema operativo Linux que se utilizará o dicho de otra manera el sistema operativo Linux que se tendrá montado sobre Windows será Debian [1], [2].

Para ello es necesario tener descargados los siguientes paquetes de la página de coLinux:

- El pauete “colinux-0.7.3.exe,” enlace:
  - http://sourceforge.net/project/showfiles.php?group_id=98788
- La versión anterior “colinux-0.6.1.exe,” enlace:
  - http://sourceforge.net/project/downloading.php?groupname=colinux&filename=colinux-0.6.1.exe&use_mirror=ufpr
- El paquete “Winpcap_3_0.exe,” enlace:
  - http://www.winpcap.org/install/default.htm
- Distribución de Debian - Debian-3.0r0.ext3.1gb.bz2, enlace:
  - http://sourceforge.net/project/downloading.php?groupname=colinux&filename=Debian-3.0r0.ext3.1gb.bz2&use_mirror=ufpr
- Partición Swap - swap_512Mb.bz2, enlace:
  - http://gniarf.nerim.net/colinux/swap/

Esta instalación asume que se cuenta con una conexión a Internet y Windows XP como sistema operativo *host*.

Las viejas versiones como coLinux 0.6.4 utilizan un archivo de configuración XML el cual se usa en esta guía a modo de información. Las versiones actuales tienen un nuevo formato de texto plano, específicamente para versiones 0.7.1 y superiores.

Antes de proceder con la instalación de coLinux es necesario modificar el archivo de configuración boot.ini de Windows XP SP2, en él, hay que cambiar en la siguiente línea, on por off así:

```
/noexecute=Always Off.
```

A continuación se procede a realizar la instalación de coLinux; simplemente se ejecuta el archivo colinux-0.7.3.exe o el colinux-0.6.1.exe según sea el caso. Una vez hecho

esto se especifica el directorio de instalación, preferiblemente c:\colinux; cuando se pida descargar la imagen de alguna distribución Linux, se recomiendo no aceptar la descargar durante la instalación, ya que es mejor descargarla previamente del enlace que se detalló anteriormente. Para los paquetes de instalación anteriores como el colinux-0.6.1.exe el Winpcap esta incluido, pero, en los paquetes de instalación actuales no está incluido, por tanto hay que instalarlo independientemente.

Después de la instalación, el directorio c:\colinux debe tener los siguientes archivos dependiendo de la versión instalada ver tabla C1:

Versión 0.7.3	***Versión 0.6.2***
colinux-bridged-net-daemon.exe	colinux-bridged-net-daemon.exe
colinux-console-fltk.exe	colinux-console-fltk.exe
colinux-console-nt.exe	colinux-console-nt.exe
colinux-daemon.exe	colinux-daemon.exe
colinux-net-daemon.exe	colinux-net-daemon.exe
colinux-slirp-net-daemon.exe	colinux-slirp-net-daemon.exe
example.conf	default.colinux.xml
initrd.gz	initrd.gz
linux.sys	linux.sys
cofs.txt	cofs.txt
colinux-daemon.txt	colinux-daemon.txt
debugging.txt	news.txt
NEWS.txt	README.txt
README.txt	Uninstall.exe
Uninstall.exe	vmlinux
vmlinux	vmlinux-modules.tar.gz
vmlinux-modules.tar.gz	[netdriver]
[netdriver]	

Tabla C1: Archivos en el directorio de coLinux.

Para este caso la imagen de Debian se descargó en un archivo compreso con extensión .bz2, por esta razón es necesario descomprimirla y ubicarla en la carpeta creada c:\colinux, el tamaño aproximado del archivo descomprimido es de 1GB.

Al igual que la distribución de Debian, la partición *swap* también se debe descargar como un archivo .bz2, la selección de este archivo depende del tamaño que se quiera

dar a la partición, se recomienda seleccionar este tamaño de acuerdo al tamaño físico de la memoria RAM del computador *host*, por ejemplo: para una memoria RAM física de 512 MB, el archivo a descargar para la *swap* debe ser swap_512Mb.bz2. Este archivo se debe descomprimir y ubicar en la carpeta creada c:\colinux, el tamaño aproximado del archivo descomprimido es de 524MB.

Después que ya todo este en la carpeta c:\colinux, se realiza la configuración del archivo example.conf para la versión 0.7.3 o el default.colinux.xml para la versión 0.6.2 de la siguiente manera:

Example.conf:

```
#
# This is an example for a configuration file that can
# be passed to colinux-daemon in this manner:
#
#    colinux-daemon @example.conf
#
# Note that you can still prepend or append configuration and
# boot parameters before and after '@', or you can use more
# that one '@ to load several settings one after another.
#
#    colinux-daemon @example.conf @overrider.conf mem=32
#
# Full list of config params is listed in colinux-daemon.txt.

# The default kernel
kernel=vmlinux

# File contains the root file system.
# Download and extract preconfigured file from SF "Images for 2.6".
cobd0="c:\colinux\Debian-3.0r0.ext3.1gb"

# Swap device, should be an empty file with 128..512MB.
cobd1="c:\colinux\swap_512Mb"

# Tell kernel the name of root device (mostly /dev/cobd0,
# /dev/cobd/0 on Gentoo)
# This parameter will be forward to Linux kernel.
root=/dev/cobd0

# Additional kernel parameters (ro = rootfs mount read only)
ro

# Initrd installs modules into the root file system.
# Need only on first boot.
initrd=initrd.gz

# Maximal memory for linux guest
#mem=64

# Slirp for internet connection (outgoing)
# Inside running colinux configure eth0 with this static settings:
# ipaddress 10.0.2.15   broadcast  10.0.2.255   netmask 255.255.255.0
# gateway   10.0.2.2    nameserver 10.0.2.3
eth0=slirp

# Tuntap as private network between guest and host on second linux device
eth1=tuntap
```

```
# Setup for serial device
#ttys0=COM1,"BAUD=115200 PARITY=n DATA=8 STOP=1 dtr=on rts=on"

# Run an application on colinux start (Sample Xming, a Xserver)
#exec0=C:\Programs\Xming\Xming.exe,":0 -clipboard -multiwindow -ac"
```

default.colinux.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<colinux>
<block_device       index="0"       path="\DosDevices\C:\colinux\Debian-3.0r0.ext3.1gb"
enabled="true"/>
<block_device index="1" path="\DosDevices\c:\colinux\swap_512Mb" enabled="true"/>
<bootparams>root=/dev/cobd0</bootparams>
<image path="vmlinux"/>
<memory size="128"/>
<network index="0" type="tap" name="TAP-Win32 Adapter"/>
<eth0=tuntap/>
</colinux>
```

Ahora para establecer la conexión a Internet es necesario renombrarla con el nombre del adaptador. Para esto se debe acceder en Windows XP a inicio, luego seleccionar configuración, conexiones de red y se hace doble clic en conexiones de red, donde se despliegan las conexiones de red disponibles. Los nombres de estas conexiones son “conexión de área local 1” el número varia según la cantidad de conexiones, generalmente es la 1 o la 2. A esta nueva conexión creada tras instalar coLinux le cambiamos el nombre y la denominamos “TAP-Win32 Adapter” este nombre debe estar también dentro de los archivos de configuración anteriormente descritos, de tal manera que el adaptador arranque cuando se inicie coLinux. En el mismo adaptador es necesario establecer la dirección IP y la mascara de subred, simplemente se da clic derecho en propiedades, doble clic en Protocolo Internet (TCP/IP) y se fijan estos campos con los valores 192.168.0.1 y 255.255.255.0 respectivamente.

Una vez renombrado el adaptador y configurados los valores de IP y mascara de subred, se ingresa en la conexión a Internet del adaptador externo, el cual corresponde a la conexión que habitualmente se usa para conectarse a Internet. Se la selecciona y se ingresa a propiedades, luego a opciones avanzadas, en conexión compartida a Internet se selecciona la casilla Permitir a usuarios de otras redes conectarse a través de la conexión a Internet de este equipo, y por ultimo se aceptan los cambios; es probable que sobresalga un cuadro de diálogo de advertencia donde especifica que el adaptador podrá ser configurado con la dirección IP 192.168.0.1, en este cuadro de diálogo se debe hacer clic en aceptar.

Por último antes de intentar arrancar coLinux se debe crear un archivo colinux.bat con el siguiente contenido:

```
colinux-daemon.exe -t nt -c default.colinux.xml
```

Si se especifica la opción *–t nt*, cuando inicie coLinux, este creará una sola ventana en la que se observará información sobre el proceso de carga de coLinux, sin embargo si desea conocer toda la información del arranque de kernel Linux se debe especificar *t*

fltk, este modo no es necesario especificar puesto que colinux inicia en este modo por defecto.

Para iniciar coLinux simplemente se ejecuta el archivo colinux.bat, este archivo inicia el proceso de carga de coLinux en una ventana y al mismo tiempo activa la conexión de Internet, representada por el adaptador de red denominado “TAP-Win32 Adapter”, si todo esta correcto al final del proceso de carga de colLinux y Debian, debe salir en la ventana algo como lo siguiente:

```
Debian GNU/Linux 3.0 colinux tty1
colinux login:
```

El *login* es *root* y no necesita *password*, una vez dentro de Debian se puede proceder a manipularlo normalmente con comandos de Linux Debian.

Para colinux no es conveniente cerrar la ventana simplemente, se recomienda utilizar el comando:

```
shutdown -h now
```

Este comando inicia el procedimiento de cerrado.

Para que la versión de Debian que se ha instalado tenga acceso a Internet se debe recordar lo realizado anteriormente, con las direcciones IP tanto en Debian como en el adaptador TAP, se sugiere que si existen inconvenientes en este proceso, se deshabilite el *firewall* de Windows o cualquier antivirus que tenga activo un *firewall*. Para entenderlo fácilmente observe la siguiente figura C1:

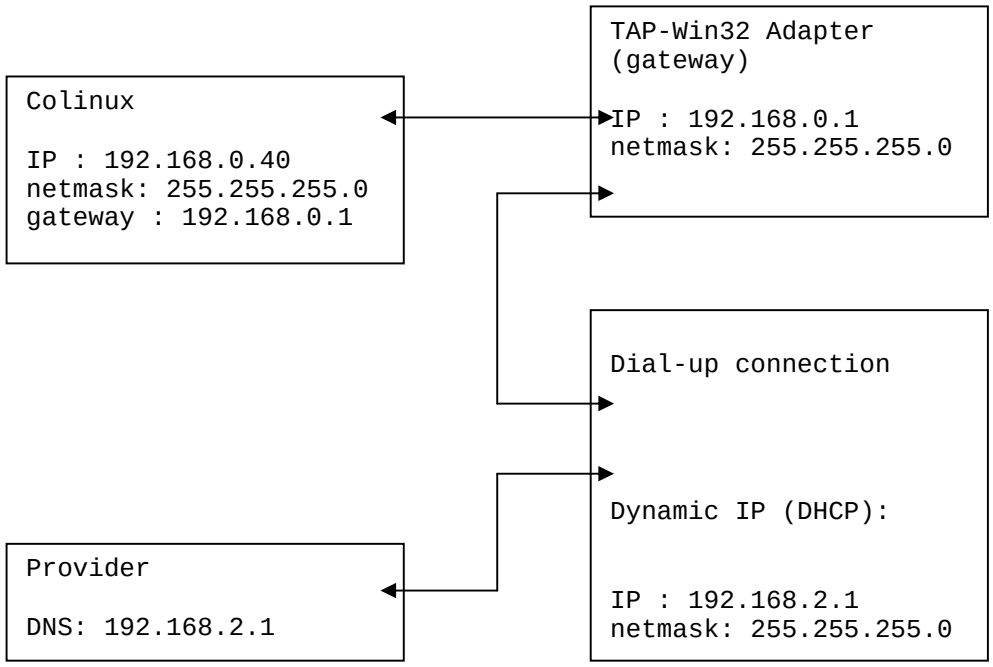

Figura C1: configuración de red para los diferentes conexiones.

En la figura C1 anterior se puede observar que coLinux Debian tiene configurada una dirección IP determinada, esta se puede ingresar editando el archivo de configuración `/etc/network/interfaces` de la siguiente manera:

Para editar este archivo se puede utilizar el programa editor "*nano*". Para utilizar este programa es necesario conocer los siguientes comandos. Para Salir (ctrl.-X), para Guardar (ctrl.-O). Utilizando estos comandos se puede editar el archivo de configuración de red llamado interfaces de la siguiente manera:

```
nano interfaces
```

Al editar el archivo interfaces este debe quedar como sigue:

```
# Used by ifup(8) and ifdown(8). See the interfaces(5) manpage or
# /usr/share/doc/ifupdown/examples for more information.


auto lo eth0

iface eth0 inet static
address 192.168.0.40
gateway 192.168.0.1
netmask 255.255.255.0
iface lo inet loopback
```

Esta configuración debe ser compatible con la configuración de la conexión compartida del TAP-Win32 Adapter que debe ser IP: 192.168.0.1 y la mascara de red: 255.255.255.0.

Una vez hecho esto se necesita abrir una ventana de línea de comandos, lo cual se puede hacer fácilmente desde Inicio, ejecutar y escribir en el cuadro de texto la palabra "cmd" y se hace clic en aceptar, se debe desplegar una ventana de línea de comandos, en ella se podrán observar las conexiones de red en Windows con la ayuda del comando `ipconfig /all`, ejecútese este comando y se obtendrá la siguiente información:

```
C:\Documents and Settings\Administrador>ipconfig /all

Configuración IP de Windows

        Nombre del host . . . . . . . . . . : LOBITO
        Sufijo DNS principal  . . . . . . . :
        Tipo de nodo. . . . . . . . . . . . : híbrido
        Enrutamiento IP habilitado. . . . . : Sí
        Proxy WINS habilitado. . . . . .    : No
        Lista de búsqueda de sufijo DNS:    orbitel.net.co

Adaptador Ethernet TAP-Win32 Adapter          :

        Sufijo de conexión específica DNS :
        Descripción. . . . . . . . . . . .  : TAP-Win32 Adapter
```

```
        Dirección física. . . . . . . . . : 00-FF-66-4A-B4-94
        DHCP habilitado. . . . . . . . . .  : No
        Dirección IP. . . . . . . . . . . . : 192.168.0.1
        Máscara de subred . . . . . . . . . : 255.255.255.0
        Puerta de enlace predeterminada   :

Adaptador Ethernet Conexión de área local          :

        Sufijo de conexión específica DNS : orbitel.net.co
        Descripción. . . . . . . . . . . .  : Conexión de red PRO/100 VE de Intel(
R)
        Dirección física. . . . . . . . . : 00-00-39-FC-90-DD
        DHCP habilitado. . . . . . . . . .  : No
        Autoconfiguración habilitada. . . : Sí
        Dirección IP. . . . . . . . . . . . : 192.168.2.11
        Máscara de subred . . . . . . . . . : 255.255.255.0
        Puerta de enlace predeterminada   : 192.168.2.1
        Servidor DHCP . . . . . . . . . . . : 192.168.2.1
        Servidores DNS . . . . . . . . . . .: 192.168.2.1
        Concesión obtenida . . . . . . . .  : lunes, 12 de enero de 2009 10:59:55
        Concesión expira . . . . . . . . . .: jueves, 22 de enero de 2009 10:59:55
```

En coLinux con el comando `ifconfig` se debe obtener lo siguiente:

```
colinux:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:43:4F:4E:45:30
          inet addr:192.168.0.40  Bcast:192.168.0.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:206 errors:0 dropped:0 overruns:0 frame:0
          TX packets:6 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:70137 (68.4 KiB)  TX bytes:327 (327.0 b)
          Interrupt:2

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)
```

En Windows se debe hacer ping a la IP 192.168.0.40 y debe dar respuesta a este ping como se muestra:

```
C:\Documents and Settings\Administrador>ping 192.168.0.40

Haciendo ping a 192.168.0.40 con 32 bytes de datos:

Respuesta desde 192.168.0.40: bytes=32 tiempo<1m TTL=64
Respuesta desde 192.168.0.40: bytes=32 tiempo<1m TTL=64
Respuesta desde 192.168.0.40: bytes=32 tiempo<1m TTL=64
Respuesta desde 192.168.0.40: bytes=32 tiempo<1m TTL=64

Estadísticas de ping para 192.168.0.40:
    Paquetes: enviados = 4, recibidos = 4, perdidos = 0
    (0% perdidos),
Tiempos aproximados de ida y vuelta en milisegundos:
    Mínimo = 0ms, Máximo = 0ms, Media = 0ms
```

Lo mismo desde colinux, se debe hacer ping a 192.168.0.1; se utiliza ping – c 4 para que se muestren 4 items del ping, esto es porque en Linux el comando ping solo sigue enviando paquetes indefinidamente caso contrario a lo que ocurre en Windows, puesto que en este último solo se envían 4 paquetes. Para poder finalizar el ping en Linux sin necesidad de escribir ping –c 4 simplemente se oprime ctrl. C.

El proceso de ping desde Linux debe dar este tipo de respuesta:

```
colinux:~# ping -c 4 192.168.0.1
PING 192.168.0.1 (192.168.0.1): 56 data bytes
64 bytes from 192.168.0.1: icmp_seq=0 ttl=128 time=1.8 ms
64 bytes from 192.168.0.1: icmp_seq=1 ttl=128 time=1.8 ms
64 bytes from 192.168.0.1: icmp_seq=2 ttl=128 time=1.6 ms
64 bytes from 192.168.0.1: icmp_seq=3 ttl=128 time=1.2 ms

--- 192.168.0.1 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 1.2/1.6/1.8 ms
```

Por otro lado si aun no se tiene salida a Internet, se debe editar el archivo `/etc/resol.conf`, en el hay que escribir la dirección del DNS de nuestro proveedor de servicio de Internet (ISP), por tanto el archivo debe verse algo así como:

```
Nameserver 192.168.2.1
#Nameserver 192.168.0.1
```

Ahora es posible realizar un ping desde coLinux a cualquier dirección en Internet, por ejemplo, a www.google.com, esto debe mostrar algo como:

```
colinux:~# ping -c 4 www.google.com
PING www.l.google.com (74.125.47.99): 56 data bytes
64 bytes from 74.125.47.99: icmp_seq=0 ttl=244 time=82.2 ms
64 bytes from 74.125.47.99: icmp_seq=1 ttl=244 time=89.2 ms
64 bytes from 74.125.47.99: icmp_seq=2 ttl=244 time=152.9 ms
64 bytes from 74.125.47.99: icmp_seq=3 ttl=244 time=67.6 ms

--- www.l.google.com ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 67.6/97.9/152.9 ms
```

Si se tiene este tipo de respuesta, todo esta perfecto, coLinux ya esta corriendo perfectamente sobre Windows.

Referencias

[1] D. Alón “Cooperative Linux,” *Proceedings of the Linux Symposium Volume One*, Ottawa, Ontario, Canada. 2004.

[2] SoureceForge.net CoLinux 2009. [En línea]. Disponible: http://www.colinux.org/ [Consulta: marzo, 2009].

Anexo D: Instalación de Globus Toolkit

La instalación del paquete de Globus Toolkit 4.2.0.1 fue realizada en 3 máquinas con las siguientes características:

Maquina 1: 2 x PIII, 512 MB RAM, discos SCSI
SO: Linux Debian Etch
Nombre: portal.grid.lsc.dc.uba.ar
Usuario: lobo

Maquina 2: PII, 400MHz, cache 512k, 128 MB RAM, disco HDD IDE
SO: ClusterKanoppix V3.6 Linux Debian Woody Kernel 2.4.27-om-20040808
Nombre: 192.168.1.8
Usuario: cpeunicauca1

Maquina 3: PIII, 1,7MHz, 512 MB RAM, discos HDD ATA
SO: Linux Debian Etch
SO: ClusterKanoppix V3.6 Linux Debian Woody Kernel 2.4.27-om-20040808
Nombre: 192.168.1.8
Usuario: cpeunicauca2

Sitios Web con información adicional básica para la instalación:

- Página Debain [1]

- Página Globus: [2], [3]

Conexión remota:

SSH: PuTTY

CONFIGURACIÓN DE LA PRIMERA MÁQUINA

Prerrequisitos:

A continuación se realiza una lista de chequeo que hay que tener en cuenta:

Los siguientes programas son prerrequisitos para que la instalación de globus sea correcta, por eso en caso de no tenerlos se deben instalar, bien sea por línea de comandos utilizando el comando apt-get install o por el Gestor de Paquetes *Synaptic*.

Para buscar si existen, se puede utilizar diversos comandos dependiendo de la ocasión por ejemplo para java:

```
lobo@neo:~$ aptitude search java (muestra que esta y no esta instalado de ese paquete)
lobo@neo:~$ apt-cache search java (muestra que esta instalado)
```

Cualquier inconveniente con el repositorio, editar el archivo para encontrar más paquetes de instalación, exactamente agregarle non-free, así:

```
lobo@portal:/etc/apt$ vim sources.list

deb http://debian.dc.uba.ar/debian/ etch main contrib non-free
# Line commented out by installer because it failed to verify:
#deb-src http://debian.dc.uba.ar/debian/ etch main

deb http://debian.dc.uba.ar/debian-security/ etch/updates main contrib non-free

# Line commented out by installer because it failed to verify:
#deb http://security.debian.org/ etch/updates main contrib
# Line commented out by installer because it failed to verify:
#deb-src http://security.debian.org/ etch/updates main contrib
```

Para maquinas con ClusterKnoppix, es conveniente hacer la actualización de este sistema para ello copiamos en el `source.list` lo siguiente:

```
# Stable
deb http://archive.debian.org/debian sarge main contrib non-free

# Stable Sources
deb-src http://archive.debian.org/debian sarge main contrib non-free
```

Una vez hecho esto, si se cuenta con más de una máquina que se desee actualizar a Sarge, para agilizar la actualización se hace una copia de los archivos de la maquina actualizada a la maquina a actualizar, estos archivos son los que el “apt” utiliza para el *update* y el *upgrade*, estos archivos se encuentran en la siguiente ruta:

```
/var/cache/apt/archives
```

Con ellos ya guardados nos se ejecuta:

```
Apt-get update
```

Y después de un tiempo:

```
Apt-get dist-upgrade
```

La Lista de requisitos empieza en este momento, a continuación se indica que tipo de paquete debe estar instalado y que hacer para verificarlo y si es el caso para instalarlo:

zlib

Primero observemos si están instaladas las librerías zlib de desarrollo para GSI-OpenSSH:

```
lobo@neo:~$ dpkg --list | grep zlib
ii  libcompress-zlib-perl             1.42-2                                Perl module
for creation and manipulation of
ii  libio-zlib-perl                   1.04-1                                IO:: style
interface to Compress::Zlib
ii  zlib-bin                          1.2.3-13                              compression
library - sample programs
ii  zlib1g                            1.2.3-13                              compression
library - runtime
ii  zlib1g-dev                        1.2.3-13                              compression
library - development
lobo@neo:~$
```

En el caso que no este instalado, se lo instala con el comando `apt-get` o se lo descarga y se realiza lo siguiente:

Preparar Zlib para su compilación:

```
CFLAGS="$CFLAGS -fPIC" \
    ./configure --prefix=/usr --shared
```

La opción *-fPIC* ayuda para asegurar la calidad de la librería dinámica zlib.
Algunos paquetes esperan que la librería zlib estática esté presente en el sistema. Para satisfacer a estos programas, compilar tanto la librería compartida como la estática:

```
make LIBS="libz.so.1.1.4 libz.a"
```

Instalar las librerías:

```
make LIBS="libz.so.1.1.4 libz.a" install
```

La librería compartida de zlib debe instalarse en el directorio `/lib`. De este modo, en el caso de que debas arrancar sin el directorio `/usr`, los programas vitales del sistema todavía tendrán acceso a la librería:

```
mv /usr/lib/libz.so.* /lib
```

El enlace simbólico `/usr/lib/libz.so` apunta a un fichero que no existe, debido a que lo hemos movido. Crea un enlace simbólico a la nueva localización de la librería:

```
ln -sf ../../lib/libz.so.1 /usr/lib/libz.so
```

J2SDK

A continuación se revisa que se tenga instalada la versión de Java 1.4.2 o superior:

```
lobo@neo:~$ java -version
java version "1.5.0_14"
```

```
Java(TM) 2 Runtime Environment, Standard Edition (build 1.5.0_14-b03)
Java HotSpot(TM) Client VM (build 1.5.0_14-b03, mixed mode, sharing)
lobo@neo:~$
```

Si no está instalado Java se procede a descargar el binario jdk-1_5_0_14-linux-i586.bin de la siguiente página [4]:

Ahora se guarda en el directorio `/usr/java` y se ejecuta:

```
root@cpedell:/usr/java# sudo ./jdk-1_5_0_14-linux-i586.bin
```

En caso de que retorne que el comando no existe entonces se debe ejecutar directamente es decir sin sudo, si aún así sale otro error que dice que no tiene permisos suficientes entonces se le cambia los permisos al archivo:

`/jdk-1_5_0_14-linux-i586.bin` con el comando `cmod 777 jdk-1_5_0_14-linux-i586.bin.`

Después de actualizar la versión de java con la que se va a trabajar se recomienda actualizar el *path* así:

```
export JAVA_HOME=/usr/java/jdk1.5.0_14

root@cpedell:/usr/java# update-alternatives --install /usr/bin/java java
/usr/java/jdk1.5.0_14/bin/java 4
root@cpedell:/usr/java# update-alternatives --config java

There is only 1 program which provides java
(/usr/java/jdk1.5.0_14/bin/java). Nothing to configure.
root@cpedell:/usr/java# java -version
java version "1.5.0_14"
Java(TM) 2 Runtime Environment, Standard Edition (build 1.5.0_14-b03)
Jav
```

Ant

A continuación se revisa si “ant” está instalado de igual manera que el anterior:

```
lobo@neo:~$ ant -version
Apache Ant version 1.6.5 compiled on July 1 2006
lobo@neo:~$
```

En este caso se recomienda instalar la versión apache-ant-1.6.2 manualmente de la siguiente dirección [5].

Se descarga el paquete de instalación apache-ant-1.6.2-bin.tar.tar y se instala en la carpeta `/usr/local/` como es un archivo tipo “tar.tar” simplemente se utiliza el comando `tar xvf` o `un tar xzf:`

```
lobo@neo:/usr/local$  de tal forma que al final todo quede aqui:
lobo@neo:/usr/local/apache-ant-1.6.2$
lobo@neo:/usr/local/tar -xzf apache-ant-1.6.2-bin.tar.tar
```

No es necesario instalarlo solo se trata de descomprimirlo.

GCC

Se requiere también de los compiladores gcc, por lo menos la versión 1.3.5:

```
lobo@neo:/$ gcc --version
gcc (GCC) 4.1.2 20061115 (prerelease) (Debian 4.1.1-21)
Copyright (C) 2006 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

g++

g++ debe estar por lo menos en la versión 1.3.5:

```
lobo@neo:/$ g++ --version
g++ (GCC) 4.1.2 20061115 (prerelease) (Debian 4.1.1-21)
Copyright (C) 2006 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

lobo@neo:/$
```

GNU versiones de tar/make/sed:

tar: debe estar por lo menos en la versión 1.14:

```
lobo@neo:/$ tar --version
tar (GNU tar) 1.16
Copyright (C) 2006 Free Software Foundation, Inc.
This is free software.  You may redistribute copies of it under the terms of
the GNU General Public License <http://www.gnu.org/licenses/gpl.html>.
There is NO WARRANTY, to the extent permitted by law.

Written by John Gilmore and Jay Fenlason.
lobo@neo:/$
```

sed: debe estar en por lo menos la versión 4.1.1:

```
lobo@neo:/$ sed --version
GNU sed version 4.1.5
Copyright (C) 2003 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE,
to the extent permitted by law.
lobo@neo:/$
```

make: debe estar por lo menos en la version 3.80:

```
lobo@neo:/$ make --version
GNU Make 3.81
Copyright (C) 2006  Free Software Foundation, Inc.
This is free software; see the source for copying conditions.
There is NO warranty; not even for MERCHANTABILITY or FITNESS FOR A
PARTICULAR PURPOSE.

This program built for i486-pc-linux-gnu
```

```
lobo@neo:/$
```

Perl: Versión mínima v5.8.4:

```
lobo@neo:/$ perl --version

This is perl, v5.8.8 built for i486-linux-gnu-thread-multi

Copyright 1987-2006, Larry Wall

Perl may be copied only under the terms of either the Artistic License or the
GNU General Public License, which may be found in the Perl 5 source kit.

Complete documentation for Perl, including FAQ lists, should be found on
this system using "man perl" or "perldoc perl".  If you have access to the
Internet, point your browser at http://www.perl.org/, the Perl Home Page.

lobo@neo:/$
```

Sudo for GRAM: versión mínima 1.6.8:

```
lobo@neo:/$ sudo -V
Sudo version 1.6.8p12
lobo@neo:/$
```

Postgresql: versión postgre 8.2.5:

```
lobo@neo:/$ dpkg --list | grep postgres
ii  postgresql               7.5.22                          object-relational SQL
database management sy
ii  postgresql-7.4           7.4.19-0etch1                   object-relational SQL
database, version 7.4
ii  postgresql-client        7.5.22                          front-end programs
for PostgreSQL (transitio
ii  postgresql-client-7.4    7.4.19-0etch1                   front-end programs
for PostgreSQL 7.4
ii  postgresql-client-common  71                           manager for multiple
PostgreSQL client versi
ii  postgresql-common         71                            manager for
PostgreSQL database clusters
lobo@neo:/$
```

Para la instalación de postgresql fue necesario poner a funcionar el comando “apt-get”, para lo cual se actualizó el `sourece.list`, en el insertamos la siguiente lista de repositorios:

```
root@box:/# vi /etc/apt/sources.list

deb http://archive.debian.org/debian-archive/debian/ woody main contrib non-free
deb http://mirrors.kernel.org/debian woody main contrib
deb-src http://http.us.debian.org/debian woody main contrib non-free
```

Después, ejecutamos lo siguiente:

```
root@box:/# apt-get update

root@box:/# apt-get upgrade,
```

Por último:

```
root@box:/# apt-get install postgresql
```

COMPILAR EL TOOLKIT:

Una vez listos los prerrequisitos para instalar globus toolkit, se descarga de la página de globus la última versión del toolkit de globus. Para propósitos de este anexo se utilizó la versión gt4.2.0-x86_deb_4.0-installer.tar.tar, la cual se puede descargar de la página de globus.

En el Terminal ingrese como usuario administrador root:

```
lobo@neo:/$ sudo su -
Password:
neo:~#
```

Usuario Globus

Una vez como root se procede a crear un usuario nuevo llamado globus:

```
neo:~# adduser globus
Añadiendo usuario 'globus' ...
Agregando nuevo grupo `globus' (1001) ...
Agregando nuevo usuario `globus' (1001) con grupo `globus' ...
Creando el directorio personal '/home/globus' ...
Copiando archivos desde '/etc/skel' ...
Introduzca la nueva contraseña de UNIX:
Vuelva a escribir la nueva contraseña de UNIX:
passwd: contraseña actualizada correctamente
Cambiando la información de usuario para globus
Introduzca el nuevo valor, o presione ENTER para el predeterminado
  Nombre completo []: globus
  Número de habitación []
  Teléfono del trabajo []:
  Teléfono de casa []:
  Otro []:
¿Es correcta la información? [y/N] y
 neo:~#
```

Ahora se crea una carpeta para globus y se le da permisos:

```
neo:~# mkdir /usr/local/globus-4.2.0.1
neo:~# chown globus:globus /usr/local/ globus-4.2.0.1
```

Ahora se copia el instalador a la carpeta recién creada, después se ingresa a la carpeta y como usuario globus se descomprime el instalador:

```
neo:/usr/local/globus-4.2.0.1# su globus
globus@neo:/usr/local/globus-4.2.0.1$
globus@neo:/usr/local/globus-4.2.0.1$ tar xvf gt4.2.0-x86_deb_4.0-installer.tar.tar
```

Seguidamente hay que configurar las variables de entorno:

Si no se esta seguro donde se encuentran estas variables, es posible buscarlas como sigue:

```
globus@neo:/$ which java
/usr/bin/java
globus@neo:/$
```

Después se utiliza el comando `ls –lisa` para ubicar exactamente donde esta el ejecutable;

```
globus@neo:/$ ls -lisa /usr/bin/java
529512 0 lrwxrwxrwx 1 root root 22 2008-09-24 13:47 /usr/bin/java ->
/etc/alternatives/java
globus@neo:/$ ls -lisa /etc/alternatives/java
33375 0 lrwxrwxrwx 1 root root 40 2008-09-24 13:47 /etc/alternatives/java ->
/usr/lib/jvm/java-1.5.0-sun/jre/bin/java
globus@neo:/$
```

Lo mismo se haría para “ant” y “globus” pero en este caso ya se conoce donde están, pues los fueron instalados manualmente, por tanto se continúa con:

```
globus@neo:/$ export JAVA_HOME=/usr/lib/jvm/java-1.5.0-sun
globus@neo:/$ export ANT_HOME=/usr/local/apache-ant-1.6.2
globus@neo:/$ export GLOBUS_LOCATION=/usr/local/globus-4.2.0.1
```

En los demás equipos puede encontrarse Java en otro sitio:

```
export  JAVA_HOME='/usr/java/jdk1.5.0_14/
```

Configuramos el PATH:

```
globus@neo:/ export PATH=$ANT_HOME/bin:$JAVA_HOME/bin:$PATH
```

NOTA: para no estar configurando las variables de entorno constantemente se edita el archivo `profile` así:

```
root@cpedell:/etc# vi /etc/profile
```

y al final de ese archivo justo después de los `Alias` pegar:

```
export JAVA_HOME=/usr/java/jdk1.5.0_14
export ANT_HOME=/usr/local/apache-ant-1.6.2
export GLOBUS_LOCATION=/usr/local/globus-4.2.0.1
export PATH=$ANT_HOME/bin:$JAVA_HOME/bin:$PATH
source /usr/local/globus-4.2.0.1/etc/globus-user-env.sh
```

Guardar, cerrar y volver a iniciar sesión

Ahora después de lo anterior, dentro de la carpeta: gt4.2.0-x86_deb_4.0-installer$enseguida ejecutamos lo siguiente:

```
globus@neo:/usr/local/globus-4.2.0.1/gt4.2.0-x86_deb_4.0-installer$    ./configure    --
prefix=$GLOBUS_LOCATION
checking for javac... /usr/lib/jvm/java-1.5.0-sun/bin/javac
checking for ant... /usr/share/ant/bin/ant
configure: creating ./config.status
config.status: creating Makefile

checking build system type... i686-pc-linux-gnu
checking for javac... /usr/java/j2sdk1.4.2_10//bin/javac
```

```
checking for ant... /usr/local/apache-ant-1.6.5/bin/ant
configure: creating ./config.status
config.status: creating Makefile
```

Ahora ya está todo listo para iniciar el procedimiento de compilación del toolkit. Se utiliza el comando `tee` para construir un `log`.

```
globus@neo:/usr/local/globus-4.2.0.1/gt4.2.0-x86_deb_4.0-installer$    make    |    tee
installer.log
```

Después de un tiempo se ejecuta:

```
globus@neo:/usr/local/globus-4.2.0.1/gt4.2.0-x86_deb_4.0-installer$ make install
…
BUILD SUCCESSFUL
Total time: 20 seconds
..Done
```

Si sale un error como:

```
running /usr/local/globus-4.2.0.1/setup/globus/setup-globus-gram-job-manager..[
Changing to /usr/local/globus-4.2.0.1/setup/globus ]
ERROR: Required perl module XML::Parser not found
ERROR: Command failed
make: *** [postinstall] Error 9
```

Para solucionar esto se debe instalar `libxml-parser-perl` con `apt-get.`

```
root@cpedell:/usr/local/globus-4.2.0.1/gt4.2.0-x86_deb_4.0-installer# apt-get install
libxml-perl
```

Ahora simplemente se vuelve a ejecutar el comando `make install.`

CONFIGURACION DE LA SEGURIDAD EN LA PRIMERA MAQUINA

Se llama las variables de entorno y el *script* fuente `globus-user-env.sh`:

Si se usa el *shell sh* for *Bourne shell*:

```
globus@neo:/$ source $GLOBUS_LOCATION/etc/globus-user-env.sh
```

Eso es dependiendo del *shell* que se utilice, en este caso es el *bash* clásico.

Antes de continuar se debe crear un usuario en este caso se llamará cpeunicauca2.

Seguimos con la Autoridad Certificante Simple:

```
globus@neo:/$ $GLOBUS_LOCATION/setup/globus/setup-simple-ca

WARNING: GPT_LOCATION not set, assuming:
         GPT_LOCATION=/usr/local/globus-4.0.1
```

```
   C e r t i f i c a t e   A u t h o r i t y   S e t u p

..... ......... ........ .....
..............................

*************************************************************************

setup-ssl-utils: Complete
```

Si sale algún error como este:

```
/usr/local/globus-4.2.0.1/setup/globus/setup-simple-ca: line 699: build.log: Permission
denied
ERROR:  could not run build command: /usr/local/globus-4.2.0.1/sbin/gpt-build -force
/home/globus/.globus/simpleCA//globus_simple_ca_32a9f489_setup-0.20.tar.gz
```

La solución será la siguiente:

Cambiar los permisos de la carpeta `/home/globus/tmp`:

```
su root
chmod 777 -R /home/globus/tmp
chmod 777 -R /home/globus
chmod 777 -R /usr/local/globus-4.2.0.1
```

Continuando antes de todo hay que asegurarse estar como root para realizar el siguiente paso

```
globus@neo:/$ su lobo
Password:
lobo@neo:/$ sudo su -
Password:
neo:~#
```

Una vez estando como *root* se verifica el estado de la variable `GLOBAL_LOCATION`:

```
neo:~# echo $GLOBAL_LOCATION
/usr/local/globus-4.2.0.1/
neo:~#
```

Si tiene los valores que debe tener no hay problema, de lo contrario hay que nuevamente dárselos así:

```
neo:~# export GLOBAL_LOCATION=/usr/local/globus-4.2.0.1/
neo:~#
```

Ahora se procede a ejecutar lo siguiente:

```
neo:~# $GLOBUS_LOCATION/setup/globus_simple_ca_CA_Hash_setup/setup-gsi –default
```

o que es lo mismo:

```
neo:/usr/local/globus-4.2.0.1/setup/globus_simple_ca_c4377068_setup#    ./setup-gsi    –
default

setup-gsi: Configuring GSI security
Making /etc/grid-security...
mkdir /etc/grid-security
```

```
Making trusted certs directory: /etc/grid-security/certificates/
mkdir /etc/grid-security/certificates/
Installing /etc/grid-security/certificates//grid-security.conf.c4377068...
Running grid-security-config...
Installing Globus CA certificate into trusted CA certificate directory...
Installing Globus CA signing policy into trusted CA certificate directory...
setup-gsi: Complete
```

Ahora se pasa a generar los archivos o certificados que emite la entidad certificante.

Se cargan nuevamente las fuentes

```
neo:~# source /usr/local/globus-4.2.0.1/etc/globus-user-env.sh
```

Ahora se ejecuta el siguiente comando teniendo en cuenta que en la parte de `hostname` se escribe el `hostname` real o se ejecuta el comando con las comillas volteaditas y el comando `hostname –f`:

```
neo:/# grid-cert-request -host `hostname -f`

Generating a 1024 bit RSA private key
....++++++
.........................++++++
writing new private key to '/etc/grid-security/hostkey.pem'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Level 0 Organization [Grid]:Level 0 Organizational Unit [GlobusTest]:Level 1
Organizational Unit [simpleCA-neo.exp.dc.uba.ar]:Name (e.g., John M. Smith) []:

A private host key and a certificate request has been generated
with the subject:

/O=Grid/OU=GlobusTest/OU=simpleCA-neo.exp.dc.uba.ar/CN=host/neo.exp.dc.uba.ar

------------------------------------------------------------

The private key is stored in /etc/grid-security/hostkey.pem
The request is stored in /etc/grid-security/hostcert_request.pem

Please e-mail the request to the Globus Simple CA jcartco@gmail.com
You may use a command similar to the following:

 cat /etc/grid-security/hostcert_request.pem | mail jcartco@gmail.com

Only use the above if this machine can send AND receive e-mail. if not, please
mail using some other method.

Your certificate will be mailed to you within two working days.
If you receive no response, contact Globus Simple CA at jcartco@gmail.com

neo:/#
```

Una vez hecho esto se verifica que los archivos o certificados estén donde deben estar:

- /etc/grid-security/hostkey.pem
- /etc/grid-security/hostcert_request.pem
- (an empty) /etc/grid-security/hostcert.pem

```
neo:/# cd /etc/grid-security/
neo:/etc/grid-security# ls
certificates          globus-user-ssl.conf  hostcert.pem          hostkey.pem
globus-host-ssl.conf  grid-security.conf    hostcert_request.pem
neo:/etc/grid-security# less hostkey.pem
neo:/etc/grid-security#
```

Ahora se cambia de usuario a globus y no olvidar para que los comandos estén presentes en el usuario globus se debe volver a ejecutar el comando:

```
globus@neo:/$ source /usr/local/globus-4.2.0.1/etc/globus-user-env.sh
```

Ahora ya se ejecuta el siguiente comando:

```
globus@neo:/$    grid-ca-sign    -in    /etc/grid-security/hostcert_request.pem    -out
hostsigned.pem
```

Si todo sale bien entonces se tendrá la siguiente salida:

```
To sign the request
please enter the password for the CA key:******

        The new signed certificate is at:

         /home/globus/.globus/simpleCA//newcerts/01.pem
```

En este paso se creó el archivo `hostsigned.pem` y `01.pem`. Si todo salió bien se debe copiar el archivo `hostsigned.pem` así:

```
cp ~globus/hostsigned.pem /etc/grid-security/hostcert.pem
```

Si por el contrario se obtuvo un error no se preocupe, por que lo único importante hasta aquí es la generación del `01.pem`, el error es el siguiente [6]:

```
ERROR: Cannot create the requested output certificate file
hostsigned.pem
Instead manually copy and rename the new signed certificate at
/home/globus/.globus/simpleCA//newcerts/01.pem
```

Si el anterior error sucede entonces copie el archivo `01.pem` a `/etc/grid-security/hostcert.pem` y reemplazarlo así:

```
root@cpedell:/home/globus# cp /home/globus/.globus/simpleCA//newcerts/01.pem /etc/grid-
security/hostcert.pem
cp: overwrite `/etc/grid-security/hostcert.pem'? y
root@cpedell:/home/globus# ls -l /etc/grid-security/
total 32
drwxrwxrwx  2 root root 4096 Feb  2 21:13 certificates
lrwxrwxrwx  1 root root   62 Feb  2 22:42 globus-host-ssl.conf -> /etc/grid-
security/certificates//globus-host-ssl.conf.32a9f489
lrwxrwxrwx  1 root root   62 Feb  2 22:42 globus-user-ssl.conf -> /etc/grid-
security/certificates//globus-user-ssl.conf.32a9f489
```

```
lrwxrwxrwx  1 root root   60 Feb  2 22:42 grid-security.conf -> /etc/grid-
security/certificates//grid-security.conf.32a9f489
-rw-r--r--  1 root root 2632 Feb  3 01:35 hostcert.pem
-rw-r--r--  1 root root 1363 Feb  2 22:46 hostcert_request.pem
-r--------  1 root root  891 Feb  2 22:46 hostkey.pem
drwxr-xr-x  2 root root 4096 Feb  3 01:22 tmp
root@cpedell:/home/globus#
```

Nota: con el comando `diff` es posible comparar las diferencias entre archivos simplemente se ejecuta `diff archivo1 archivo2.`

Generar los certificados para un usuario (distinto de globus y root)

Hasta ahora se tiene un usuario globus que actúa como autoridad certificadora, un usuario *root* que tiene los certificados del *host*, a continuación se van a crear los certificado para un usuario, en este caso el usuario se llama “cpeunicauca2” que ya debe estar adicionado al sistema.

Inicie sesión como ese usuario y solicite un certificado así:

```
root@cpedell:/etc/grid-security# su cpeunicauca2
cpeunicauca2@cpedell:/etc/grid-security$ source $GLOBUS_LOCATION/etc/globus-user-env.sh
cpeunicauca2@cpedell:/etc/grid-security$ grid-cert-request
```

Este comando generará tres archivos en `/home/cpeunicauca2/.globus/` es similar a la generación de certificados para el *host*, solo que en este caso se hace para el usuario cpeunicauca2. La ejecución de este comando muestra la siguiente salida:

```
…..
A private key and a certificate request has been generated with the subject:

/O=Grid/OU=GlobusTest/OU=simpleCA-cpedell/CN=cpeunicauca

If the CN=cpeunicauca is not appropriate, rerun this
script with the -force -cn "Common Name" options.

Your private key is stored in /home/cpeunicauca2/.globus/userkey.pem
Your request is stored in /home/cpeunicauca2/.globus/usercert_request.pem

Please e-mail the request to the Globus Simple CA cpeunicauca2@cpedell
You may use a command similar to the following:

  cat /home/cpeunicauca2/.globus/usercert_request.pem | mail cpeunicauca2@cpedell

Only use the above if this machine can send AND receive e-mail. if not, please
mail using some other method.

Your certificate will be mailed to you within two working days.
If you receive no response, contact Globus Simple CA at cpeunicauca2@cpedell

Los tres archivos generados son una llave, petición de certificado y el certificado de
usuario. Para ver estos archivos ejecutemos:

cpeunicauca2@cpedell:/etc/grid-security$ ls /home/cpeunicauca2/.globus/
usercert.pem  usercert_request.pem  userkey.pem
```

Ahora se necesita obtener la petición del certificado para el usuario globus para que éste la pueda firmar y regresar el certificado firmado al usuario cpeunicauca2. Para esto iniciamos sesión como globus.

```
root@cpedell:/etc/grid-security# su globus
globus@cpedell:/etc/grid-security$ cd /home/cpeunicauca2/.globus/
globus@cpedell:/home/cpeunicauca2/.globus$   source   $GLOBUS_LOCATION/etc/globus-user-
env.sh
globus@cpedell:/home/cpeunicauca2/.globus$ grid-ca-sign -in usercert_request.pem -out
signed.pem
```

Se obtiene la siguiente salida (si sale un error no se preocupe):

```
To sign the request
please enter the password for the CA key:
The new signed certificate is at: /home/globus/.globus/simpleCA//newcerts/02.pem

ERROR: Cannot create the requested output certificate file
   signed.pem
Instead manually copy and rename the new signed certificate at
   /home/globus/.globus/simpleCA//newcerts/02.pem
```

Ahora el certificado está firmado por el usuario globus y se encuentra en el archivo `02.pem`. Este es el archivo firmado que se debe enviar al usuario cpeunicauca2, en nuestro caso simplemente se copiará al archivo `usercert.pem` que se encuentra en la carpeta `home/cpeunicauca2/.globu/usercert.pem`:

```
globus@cpedell:/home/cpeunicauca2/.globus$ su root
Password:
root@cpedell:/home/cpeunicauca2/.globus#                                        cp
/home/globus/.globus/simpleCA/newcerts/02.pem /home/cpeunicauca2/.globus/usercert.pem
cp: overwrite `/home/cpeunicauca2/.globus/usercert.pem'? y
root@cpedell:/home/cpeunicauca2/.globus#
```

En este momento el usuario cpeunicauca2 ya tiene su certificado firmado por la autoridad certificadora representada por el usuario globus: por último es indispensable hacer que las credenciales *host* sean accesibles por el container.

```
As root, run:
root# cd /etc/grid-security
root# cp hostkey.pem containerkey.pem
root# cp hostcert.pem containercert.pem
root# chown globus.globus containerkey.pem containercert.pem
root# chown globus.globus container*.pem
At this point the certificates in /etc/grid-security should look something like:
root# ls -l *.pem
-rw-r--r-- 1 globus globus 1785 Oct 14 14:47 containercert.pem
-r-------- 1 globus globus 887 Oct 14 14:47 containerkey.pem
-rw-r--r-- 1 root root 1785 Oct 14 14:42 hostcert.pem
-r-------- 1 root root 887 Sep 29 09:59 hostkey.pem
```

Nota: Dado el caso que sea un equipo externo no se puede copiar tan fácilmente los certificados en tal caso se debe enviar los certificados vía e-mail y renombrarlos de la manera apropiada, para esto por favor refiérase a la sección “Setting up security on your first machine” de la guía de globus que se puede encontrar en la dirección [3].

Probar el certificado

Estando en la carpeta `/home/cpeunicauca2/.globus` ejecute el siguiente comando:

```
root@cpedell:/# su cpeunicauca2
cpeunicauca2@cpedell:/$ cd /home/cpeunicauca2/.globus/
cpeunicauca2@cpedell:~/.globus$ source $GLOBUS_LOCATION/etc/globus-user-env.sh
cpeunicauca2@cpedell:~/.globus$ grid-proxy-init -debug –verify
```

Si todo sale bien se obtendrá la siguiente salida:

```
cpeunicauca2@cpedell:~/.globus$ grid-proxy-init -debug -verify

User Cert File: /home/cpeunicauca2/.globus/usercert.pem
User Key File: /home/cpeunicauca2/.globus/userkey.pem

Trusted CA Cert Dir: /etc/grid-security/certificates

Output File: /tmp/x509up_u1000
Your identity: /O=Grid/OU=GlobusTest/OU=simpleCA-cpedell/CN=cpeunicauca
Enter GRID pass phrase for this identity:
Creating proxy ............++++++++++++
.....................++++++++++++
 Done
Proxy Verify OK
Your proxy is valid until: Fri Feb  6 17:30:31 2009
```

Si sale el siguiente error:

```
grid-proxy-init: error while loading shared libraries: libssl.so.0.9.8: cannot open
shared object file: No such file or directory
cpeunicauca2@cpedell:~/.globus$
```

Se debe actualizar las versiones de `libssl.so.0.9.8` y `libcrypto.so` a las nuevas versiones.

Primero se renombra los paquetes del `libssl.so` así:

```
cpeunicauca2@cpedell:~/.globus$ su root
Password:
root@cpedell:/home/cpeunicauca2/.globus# updatedb
root@cpedell:/home/cpeunicauca2/.globus# locate libssl.so
/usr/lib/i486/libssl.so.0.9.7
/usr/lib/i586/libssl.so.0.9.7
/usr/lib/i686/cmov/libssl.so.0.9.7
/usr/lib/libssl.so
/usr/lib/libssl.so.0.9.7
root@cpedell:/home/cpeunicauca2/.globus# cp /usr/lib/libssl.so.0.9.7
/usr/lib/libssl.so.0.9.8
root@cpedell:/home/cpeunicauca2/.globus# su cpeunicauca2
cpeunicauca2@cpedell:~/.globus$ source $GLOBUS_LOCATION/etc/globus-user-env.sh
cpeunicauca2@cpedell:~/.globus$ grid-proxy-init -debug -verify
grid-proxy-init: error while loading shared libraries: libcrypto.so.0.9.8: cannot open
shared object file: No such file or directory
cpeunicauca2@cpedell:~/.globus$ su root
Password:
root@cpedell:/home/cpeunicauca2/.globus# locate libcrypto.so
/usr/lib/i486/libcrypto.so.0.9.7
/usr/lib/i586/libcrypto.so.0.9.7
/usr/lib/i686/cmov/libcrypto.so.0.9.7
/usr/lib/libcrypto.so
```

```
/usr/lib/libcrypto.so.0.9.7
/usr/share/chaos/lib/libcrypto.so.0.9.7
/usr/share/chaos/lib/libcrypto.so.4
root@cpedell:/home/cpeunicauca2/.globus# cp /usr/lib/libcrypto.so.0.9.7
/usr/lib/libcrypto.so.0.9.8
root@cpedell:/home/cpeunicauca2/.globus# su cpeunicauca2
cpeunicauca2@cpedell:~/.globus$ source $GLOBUS_LOCATION/etc/globus-user-env.sh
cpeunicauca2@cpedell:~/.globus$ grid-proxy-init -debug -verify
grid-proxy-init: /usr/lib/libcrypto.so.0.9.8: no version information available
(required by grid-proxy-init)
grid-proxy-init: /usr/lib/libcrypto.so.0.9.8: no version information available
(required by /usr/local/globus-4.2.0.1/lib/libglobus_gsi_proxy_core_gcc32.so.0)
grid-proxy-init: /usr/lib/libcrypto.so.0.9.8: no version information available
(required by /usr/local/globus-4.2.0.1/lib/libglobus_gsi_credential_gcc32.so.0)
grid-proxy-init: /usr/lib/libssl.so.0.9.8: no version information available (required
by /usr/local/globus-4.2.0.1/lib/libglobus_gsi_callback_gcc32.so.0)
grid-proxy-init: /usr/lib/libcrypto.so.0.9.8: no version information available
(required by /usr/local/globus-4.2.0.1/lib/libglobus_gsi_callback_gcc32.so.0)
grid-proxy-init: /usr/lib/libcrypto.so.0.9.8: no version information available
(required by /usr/local/globus-4.2.0.1/lib/libglobus_gsi_sysconfig_gcc32.so.0)
grid-proxy-init: /usr/lib/libcrypto.so.0.9.8: no version information available
(required by /usr/local/globus-4.2.0.1/lib/libglobus_gsi_cert_utils_gcc32.so.0)
grid-proxy-init: /usr/lib/libcrypto.so.0.9.8: no version information available
(required by /usr/local/globus-4.2.0.1/lib/libglobus_openssl_gcc32.so.0)
grid-proxy-init: /usr/lib/libssl.so.0.9.8: no version information available (required
by /usr/local/globus-4.2.0.1/lib/libglobus_openssl_error_gcc32.so.0)
grid-proxy-init: /usr/lib/libcrypto.so.0.9.8: no version information available
(required by /usr/local/globus-4.2.0.1/lib/libglobus_openssl_error_gcc32.so.0)
grid-proxy-init: /usr/lib/libcrypto.so.0.9.8: no version information available
(required by /usr/local/globus-4.2.0.1/lib/libglobus_proxy_ssl_gcc32.so.0)

User Cert File: /home/cpeunicauca2/.globus/usercert.pem
User Key File: /home/cpeunicauca2/.globus/userkey.pem

Trusted CA Cert Dir: /etc/grid-security/certificates

Output File: /tmp/x509up_u1000
Your identity: /O=Grid/OU=GlobusTest/OU=simpleCA-cpedell/CN=cpeunicauca
Enter GRID pass phrase for this identity:
Creating proxy .................++++++++++++
............++++++++++++
grid-proxy-init: relocation error: /usr/local/globus-
4.2.0.1/lib/libglobus_proxy_ssl_gcc32.so.0: symbol asn1_const_Finish, version
OPENSSL_0.9.8 not defined in file libcrypto.so.0.9.8 with link time reference
```

Este último error se produce por las versiones del `libcrypto.so.0.9.8` para esto se requiere utilizar las versiones actualizadas de este archivo y reemplazar las siguientes carpetas:

```
/usr/lib/i486
/usr/lib/i586
/usr/lib/i686
```

y además copiar el archivo (Nota: deben quedar los dos archivos tanto la versión 0.9.8 como la versión 0.9.7 pero solo en el caso del archivo `/usr/lib/libcrypto.so.0.9.7`)
`/usr/lib/libcryto.so.0.9.8.`

Para esto se debe tomar los archivos que tengan la versión 0.9.7 y cambiarles el nombre (por ejemplo antecediéndolos de la palabra old).

Luego para comprobar que todo va bien se debe ejecutar el siguiente comando:

```
cpeunicauca2@cpedell:~/.globus$ locate libcrypto.so
/usr/lib/i486/libcrypto.so.0.9.8
/usr/lib/i586/libcrypto.so.0.9.8
/usr/lib/i686/cmov/libcrypto.so.0.9.8
/usr/lib/libcrypto.so
/usr/lib/libcrypto.so.0.9.8
/usr/lib/old_libcrypto.so.0.9.7
/usr/share/chaos/lib/libcrypto.so.0.9.7
/usr/share/chaos/lib/libcrypto.so.4
```

Ahora solo resta volver a ejecutar el comando:

```
cpeunicauca2@cpedell:~/.globus$ grid-proxy-init -debug -verify
```

Si todo sale bien entonces obtendremos la siguiente salida:

```
User Cert File: /home/cpeunicauca2/.globus/usercert.pem
User Key File: /home/cpeunicauca2/.globus/userkey.pem

Trusted CA Cert Dir: /etc/grid-security/certificates

Output File: /tmp/x509up_u1000
Your identity: /O=Grid/OU=GlobusTest/OU=simpleCA-cpedell/CN=cpeunicauca
Enter GRID pass phrase for this identity:
Creating proxy ............++++++++++++
.......................++++++++++++
 Done
Proxy Verify OK
Your proxy is valid until: Fri Feb  6 17:30:31 2009
```

El útlimo paso de los certificados es crear el grid-mapfile, el cual se crea como usuario root para la autorización:

Por lo tanto se debe tener:

- el subject de nombre de usuario
- el nombre de la cuenta

Para el subject cmo usuario cpeunicauca2 se hace lo siguiente:

```
cpeunicauca2@cpetoshiba:/root$ source /usr/local/globus-4.2.0.1/etc/globus-user-env.sh
cpeunicauca2@cpetoshiba:/root$ grid-cert-info -subject
/O=Grid/OU=GlobusTest/OU=simpleCA-cpetoshiba/CN=cpeunicauca
```

Para el nombre de cuenta:

```
cpeunicauca2@cpetoshiba:/etc/grid-security$ whoami
cpeunicauca2
```

Ahora si como root se escribe dentro del archivo grid-mapfile como sigue:

```
root@cpetoshiba:~# cd /etc/grid-security/
```

```
root@cpetoshiba:/etc/grid-security# /usr/local/globus-4.2.0.1/sbin/grid-mapfile-add-
entry -dn "/O=Grid/OU=GlobusTest/OU=simpleCA-cpetoshiba/CN=cpeunicauca" -ln
cpeunicauca2
Modifying /etc/grid-security/grid-mapfile ...
/etc/grid-security/grid-mapfile does not exist... Attempting to create /etc/grid-
security/grid-mapfileNew entry:
"/O=Grid/OU=GlobusTest/OU=simpleCA-cpetoshiba/CN=cpeunicauca" cpeunicauca2
(1)   entry added
```

Por ultimo se verifica que todo quede en orden dentro de nuestro grid-mapfile:

```
cpeunicauca2@cpetoshiba:/etc/grid-security$ less grid-mapfile
"/O=Grid/OU=GlobusTest/OU=simpleCA-cpetoshiba/CN=cpeunicauca" cpeunicauca2
cpeunicauca2@cpetoshiba:/etc/grid-security$
```

Nota: si por algún caso no es posible obtener información con el comando `grid-cert-info –subject` se puede obtener el subject con el comando `grid-proxy-init -verify –debug` y lo se obtiene del campo `Your identity: /O=Grid/OU=GlobusTest/OU=simpleCA-cpetoshiba/CN=cpeunicauca`.

Para hacerlo con correo electrónico

Se siguen los siguientes pasos:

```
root@cpedell:/home/globus/.globus# su globus
globus@cpedell:~/.globus$ cd /home/globus/.globus/
globus@cpedell:~/.globus$ source /usr/local/globus-4.2.0.1/etc/globus-user-env.sh
globus@cpedell:~/.globus$ grid-ca-sign -in usercert_request.pem -out
signed.pem

To sign the request
please enter the password for the CA key:

The new signed certificate is at: /home/globus/.globus/simpleCA//newcerts/02.pem

/bin/cp: cannot create regular file `signed.pem': Permission denied
globus@cpedell:~/.globus$
```

Despues se hace lo siguiente:

```
globus@cpedell:~/.globus$ su -
Password:
root@cpedell:~# cp /home/globus/.globus/simpleCA/newcerts/02.pem
/home/globus/.globus/usercert.pem
cp: overwrite `/home/globus/.globus/usercert.pem'? y
root@cpedell:~#
```

Bueno ahora se necesita el certificado enviado por mail, se lo envía de la siguiente manera:

Desde root:

```
neo:~# cat /home/lobo /.globus/usercert_request.pem | mail globus
```

Darse cuenta que no se utiliza el globus@neo, ahora que ya esta en el mail, hay que logearse como globus y con el comado mail se busca el correo:

```
globus@neo:~$ mail
Mail version 8.1.2 01/15/2001.  Type ? for help.
"/var/mail/globus": 1 message 1 new
>N  1 lobo@neo.exp.dc.u  Mon Sep 29 12:02   56/1969  Certificado
& exit
globus@neo:~$
```

Una vez se esta allí se ingresa al correo con al número del mensaje en este caso 1

```
>N  1 lobo@neo.exp.dc.u  Mon Sep 29 12:02   56/1969  Certificado
& 1
Message 1:
From lobo@neo.exp.dc.uba.ar Mon Sep 29 12:02:30 2008
Envelope-to: globus@neo.exp.dc.uba.ar
Delivery-date: Mon, 29 Sep 2008 12:02:30 -0300
To: globus@neo.exp.dc.uba.ar
Subject: Certificado
From: lobo@neo.exp.dc.uba.ar
Date: Mon, 29 Sep 2008 12:02:30 -0300

This is a Certificate Request file:

It should be mailed to jcartco@gmail.com


===========================================================================
Certificate Subject:

    /O=Grid/OU=GlobusTest/OU=simpleCA-neo.exp.dc.uba.ar/OU=exp.dc.uba.ar/CN=Juan
 Carlos

The above string is known as your user certificate subject, and it
uniquely identifies this user.
```

Ahora se exporta como un archivo de texto con el comando `s` con el nombre de `request.pem` si quieren ver las ayudas de mail simplemente le se escribe help y salen listados los comandos:

Una vez tengamos el nuevo archivo con el contenido del mail, se edita y se borra el encabezado.

Este archivo debe estar ubicado en:

```
/home/globus
```

El archivo tiene el nombre `request.pem` se ingresa con cualquier editor y se borra toda la cabecera y solo se deja el contenido:

```
This is a Certificate Request file:

It should be mailed to jcartco@gmail.com


===========================================================================
```

```
Certificate Subject:

    /O=Grid/OU=GlobusTest/OU=simpleCA-neo.exp.dc.uba.ar/OU=exp.dc.uba.ar/CN=Juan
 Carlos

The above string is known as your user certificate subject, and it
uniquely identifies this user.

To install this user certificate, please save this e-mail message
into the following file.


/home/lobo/.globus/usercert.pem


      You need not edit this message in any way. Simply
      save this e-mail message to the file.



If you have any questions about the certificate contact
the Globus Simple CA at jcartco@gmail.com

-----BEGIN CERTIFICATE REQUEST-----
MIIB6TCCAVICAQAwdzENMAsGA1UEChMER3JpZDETMBEGA1UECxMKR2xvYnVzVGVz
dDEjMCEGA1UECxMac2ltcGxlQ0EtbmVvLmV4cC5kYy51YmEuYXIxFjAUBgNVBAsT
DWV4cC5kYy51YmEuYXIxFDASBgNVBAMTC0p1YW4gQ2FybG9zMIGfMA0GCSqGSIb3
DQEBAQUAA4GNADCBiQKBgQCtZESPFiPa8OYRb1wIY0NMXvVL6C/SESM+5KrIB4NI
j3jLk+hcYZWc/rgDxgSxwBFdSMxzspQj753d5x3tBTvCrR9+4rBFCU0ThisskBbF
6eh2dVc3M9XDpsbRFNz33bzmQdbFzGHPzq/7j9MnLD1a8dVptV+Y/IcPugscDvXF
5wIDAQABoDIwMAYJKoZIhvcNAQkOMSMwITARBglghkgBhvhCAQEEBAMCBPAwDAYD
VR0TAQH/BAIwADANBgkqhkiG9w0BAQUFAAOBgQAvH3FYA0e/ZDPC1g1RxJbXK8AJ
nnr7TcxbdKqJCDoRW37XXfBoBJxhDcqbmuFODDnZowh994+f7/lACtuMbsMf3UQB
5+1ZxVgJsmc7WA1myxDWlvLe8d4kWH8PE2grBtUFRZaLzznppOcIzq6C6O38cvnj
nGZO+dTie/Uo1CiooQ==
-----END CERTIFICATE REQUEST-----
```

Una vez editado el archivo `request.pem` se procede a ejecurtar el commando:

```
globus@neo:~$ grid-ca-sign -in request.pem -out signed.pem
```

Despuesse realizán los anteriores pasos para este archivo:

Se envía al mail de lobo, se entra al mail de `/home/lobo` se lee el mail y lse exporta como `signed.pem`, se edita con el vim y se borra el encabezado y por último se copia a la carpeta:

```
lobo@neo:~$ cp signed.pem /home/lobo/.globus/usercert.pem
```

Ahora se ejecuta:

```
globus@cpedell:/etc/grid-security$  source  /usr/local/globus-4.2.0.1/etc/globus-user-
env.sh
globus@cpedell:/etc/grid-security$ grid-cert-info -subject
/O=Grid/OU=GlobusTest/OU=simpleCA-cpedell/CN=globus
globus@cpedell:/etc/grid-security$ whoami
globus
globus@cpedell:/etc/grid-security$ su root
Password:
```

```
root@cpedell:/etc/grid-security#  /usr/local/globus-4.2.0.1/sbin/grid-mapfile-add-entry
-dn "/O=Grid/OU=GlobusTest/OU=simpleCA-cpedell/CN=glob
us" -ln globus
Modifying /etc/grid-security/grid-mapfile ...
/etc/grid-security/grid-mapfile  does  not  exist...  Attempting  to  create  /etc/grid-
security/grid-mapfile
New entry:
"/O=Grid/OU=GlobusTest/OU=simpleCA-cpedell/CN=globus" globus
(1) entry added
root@cpedell:/etc/grid-security# less grid-mapfile
"/O=Grid/OU=GlobusTest/OU=simpleCA-cpedell/CN=globus" globus
root@cpedell:/etc/grid-security# su globus


lobo@neo:/$ grid-cert-info -subject
/O=Grid/OU=GlobusTest/OU=simpleCA-neo.exp.dc.uba.ar/OU=exp.dc.uba.ar/CN=Juan Carlos
lobo@neo:/$ whoami
lobo
lobo@neo:/$

neo:/etc/grid-security# /usr/local/globus-4.2.0.1/sbin/grid-mapfile-add-entry -dn
"/O=Grid/OU=GlobusTest/OU=simpleCA-neo.exp.dc.uba.ar/OU=exp.dc.uba.ar/CN=Juan Carlos" -
ln lobo
```

La linea en el grid-mapfile debe quedar como esta [7]:

```
lobo@neo:/etc/grid-security$ less grid-mapfile
"/O=Grid/OU=GlobusTest/OU=simpleCA-neo.exp.dc.uba.ar/OU=exp.dc.uba.ar/CN=Juan Carlos"
lobo
```

Grid FTP

Ahora que ya están los certificados de seguridad en su lugar es posible montar un servicio en este caso el servicio de GridFTP:

Como root y se agrega la siguiente linea al archivo `/etc/services`:

```
neo:~# vim /etc/services
```

`gsiftp 2811/tcp` (agregarlo preferiblemente en orden de puertos)

Entramos como root y modificamos el archivo inetd:

Debemos primero que todo modificar el archivo Inetd.conf o xinetd.conf dependiendo del host, como se muestra en [8]:

Se abbre `/etc/inetd.conf` y se agrega las siguientes lineas, teniendo en cuenta que no se deja espacios depues de root:

```
gsiftp          stream  tcp     nowait  root /usr/bin/env env
GLOBUS_LOCATION=/usr/local/globus-4.2.0.1 LD_LIBRARY_PATH=/usr/
local/globus-4.2.0.1/lib /usr/local/globus-4.2.0.1/sbin/globus-gridftp-server -i
```

Después se verifica que todo haya corrido perfectamente así:

Se crea un archivo en el root de lobo en este caso creamos un archivo denominado prueba:

```
vim prueba
```

Dentro de este archivo se escribe cualquier cosa una vez se tenga el archivo creado en el root de lobo:

```
lobo@neo:/$ ls
bin    dev   initrd          lib         mnt   prueba  selinux  tmp  vmlinuz
boot   etc   initrd.img      lost+found  opt   root    srv      usr  vmlinuz.old
cdrom  home  initrd.img.old  media       proc  sbin    sys      var
lobo@neo:/$
```

Se procede a realizar el envio del archivo así:

```
lobo@neo:/$ globus-url-copy -vb gsiftp://localhost/prueba file:///tmp/file2
Source: gsiftp://localhost/
Dest:   file:///tmp/
  prueba  ->  file2

lobo@neo:/$
```

Una vez hecho esto verificamos que los dos archivos sean idénticos con muestro comando diff

```
lobo@neo:/$ diff /prueba /tmp/file2
lobo@neo:/$
```

Listo, quedo el GridFTP.

Si sale este error

```
cpeunicauca2@cpedell:/$ globus-url-copy -vb gsiftp://localhost/Pruebacpedell
file:///tmp/file2
Source: gsiftp://localhost/
Dest:   file:///tmp/
  Pruebacpedell  ->  file2

error: globus_xio: Unable to open file /tmp/file2
globus_xio: System error in open: Permission denied
globus_xio: A system call failed: Permission denied
cpeunicauca2@cpedell:/$
```

Esto se corrige cambiando los permisos del archivo file2, ubiquese ne la carpeta en la que se encuentra la cel archivo file2 en nuestro caso se ubica en la carpeta tmp. así:

```
root@cpetoshiba:/# chown cpeunicauca2:cpeunicauca2 /file2
root@cpetoshiba:/# su cpeunicauca2

cpeunicauca2@cpedell:/$ globus-url-copy -vb gsiftp://localhost/pruebacpedell
file:///tmp/file2
Source: gsiftp://localhost/
Dest:   file:///tmp/
  pruebacpedell  ->  file2

error: globus_xio: Unable to connect to localhost:2811
globus_xio: System error in connect: Connection refused
globus_xio: A system call failed: Connection refused
```

Si el error persiste, reiniciar el equipo y asegurarse que el hostcert.pem sea igual al 01.pem

```
cpeunicauca1@cpehva:/$ globus-url-copy -vb gsiftp://localhost/pruebacpehva
file:///tmp/file2
Source: gsiftp://localhost/
Dest:   file:///tmp/
  pruebacpehva  ->  file2

error: globus_xio: Unable to connect to localhost:2811
globus_xio: System error in connect: Connection refused
globus_xio: A system call failed: Connection refused
```

Si persiste el error, probablemente no este corriendo el gridftp-server, es conveniente correrlo en una ventana aparte como root ya que el se queda ejecutandose indefinidamente.

```
root@cpehva:/# /usr/local/globus-4.0.8/sbin/globus-gridftp-server -p 2811
```

O intentar correrlo como demonio:

```
root@cpehva:/# /usr/local/globus-4.0.8/sbin/globus-gridftp-server -s -p 2811
```

Iniciando el Contenedor de Servicios Web

Como 157.92.26.45 es la dirección IP. Es necesario fijar esta dirección de lo contrario no funcionará. Para ello edite los siguentes archivos:

```
$GLOBUS_LOCATION/etc/globus_wsrf_core/server-config.wsdd
En estos archivos agregue la linea que esta en negrita dependiendo tu dirección IP

<parameter   name="logicalHost"   value="157.92.26.45"/>   debajo   de   la   sección
<globalConfiguration>. poe ejemplo:

<globalConfiguration>
        <parameter name="logicalHost" value="157.92.26.45"/>
        <parameter name="usageStatisticsTargets"
                   value="usage-stats.globus.org:4810"/>
```

y también el archivo `client-server-config.wsdd`,

```
<globalConfiguration>
                 <parameter          name="logicalHost"          value="192.168.1.8"/>
                   <parameter            name="containerThreads"            value="1"/>
        <parameter name="containerThreadsMax" value="3"/>
```

Ahora se configura una entrada en el archivo `/etc/init.d` para el contenedor de servicios web,

```
neo:~# vim /usr/local/globus-4.2.0.1/start-stop
```

Dentro de aquel archivo se introduce lo siguiente:

```
#! /bin/sh
set -e
```

```
export GLOBUS_LOCATION=/usr/local/globus-4.2.0.1
export JAVA_HOME=/usr/lib/jvm/java-1.5.0-sun
export ANT_HOME=/usr/local/apache-ant-1.6.2
export GLOBUS_OPTIONS="-Xms256M -Xmx512M"

. $GLOBUS_LOCATION/etc/globus-user-env.sh
. $GLOBUS_LOCATION/etc/globus-devel-env.sh

cd $GLOBUS_LOCATION
case "$1" in
    start)
         $GLOBUS_LOCATION/sbin/globus-start-container-detached -p 8443 -nosec
         ;;
    stop)
         $GLOBUS_LOCATION/sbin/globus-stop-container-detached
         ;;
    *)
         echo "Usage: globus {start|stop}" >&2
         exit 1
         ;;
esac
exit 0
```

Hay que tener muy claro las rutas para las variables de entorno y que en enste caso cuanto se inicie en `globus-start-container-detached`, lo hacemos con `–nosec` lo cual hace que no se activen los certificados de seguridad por tanto no se autentican dichos certificados.

Despues se dan `export` misos a dicho archivo

```
neo:~# chmod +x $GLOBUS_LOCATION/start-stop
```

Si por casualidad la ejecución de este script falla y muestra un mensaje como:

```
cpeunicauca2@cpetoshiba:/usr/local/globus-4.2.0.1$          start-stop          start
: bad interpreter: No such file or directory
```

Entonces lo que se debe hacer es lo siguiente, ingrese al archivo utilizando un editor (como vi) de la siguiente forma:

```
vi /usr/local/globus-4.2.0.1/start-stop
```

Presione: y escriba `set fileformat=unix`

Y a continuación presione “*enter*” finalmente se debe escribir `wq!` y nuevamente “enter”, aunque no se note nada el archivo ya está arreglado ahora finalmente vueva a ejecutar el scritp y todo funciona bien.

Ahora como root creamos un script /etc/init.d para llamar el script star-stop del usuario globus:

```
neo:~# vim /etc/init.d/globus-4.2.0.1
```

Dentro de este archivo insertamos el siguiente código:

```
#!/bin/sh -e
```

```
case "$1" in
  start)
    su - globus /usr/local/globus-4.2.0.1/start-stop start
    ;;
  stop)
    su - globus /usr/local/globus-4.2.0.1/start-stop stop
    ;;
  restart)
    $0 stop
    sleep 1
    $0 start
    ;;
  *)
    printf "Usage: $0 {start|stop|restart}\n" >&2
    exit 1
    ;;
esac
exit 0
```

Se Hace lo mismo para que funcione este script:

```
vi /etc/init.d/globus-4.2.0.1
```

Presione: y escriba `set fileformat=unix`

```
neo:~# chmod +x /etc/init.d/globus-4.2.0.1
neo:~# /etc/init.d/globus-4.2.0.1 start
```

En el caso que funcione debe salir:

```
globus@cpetoshiba:/usr/local/globus-4.2.0.1$ /etc/init.d/globus-4.2.0.1 start
Password:
Starting Globus container. PID: 7201
globus@cpetoshiba:/usr/local/globus-4.2.0.1$ /etc/init.d/globus-4.2.0.1 stop
Password:
Stopping Globus container. PID: 7201
Container stopped
globus@cpetoshiba:/usr/local/globus-4.2.0.1$
```

O en el caso que no este funcionando muy bien el scrip de `globus-4.2.0.1` hacemos lo siguiente manualmete:

```
neo:/usr/local/globus-4.2.0.1# ./start-stop start
Starting Globus container. PID: 2737
neo:/usr/local/globus-4.2.0.1# tail -f /usr/local/globus-4.2.0.1/var/container.log
Starting SOAP server at http://157.92.26.45:8443/wsrf/services/

With the following services:

[1]: http://157.92.26.45:8443/wsrf/services/AdminService
[2]: http://157.92.26.45:8443/wsrf/services/AttachmentTestService
.
…
…..
[66]: http://157.92.26.45:8443/wsrf/services/mds/test/subsource/IndexServiceEntry
[67]: http://157.92.26.45:8443/wsrf/services/mds/test/usefulrp/IndexService
[68]: http://157.92.26.45:8443/wsrf/services/mds/test/usefulrp/IndexServiceEntry
```

```
2008-10-08T15:32:26.022-03:00 INFO  impl.DefaultIndexService [ServiceThread-
58,performDefaultRegistrations:261]  guid=747ecca0-9567-11dd-9f3a-e6a6ae5de4df
event=org.globus.mds.index.performDefaultRegistrations.end status=0
```

En el caso se hace lo siguiente:

```
globus@cpetoshiba:/usr/local/globus-4.2.0.1$ /usr/local/globus-4.2.0.1/start-stop start
Starting Globus container. PID: 7033
globus@cpetoshiba:/usr/local/globus-4.2.0.1$ tail -f /usr/local/globus-
4.2.0.1/var/container.log
        at
org.apache.commons.logging.impl.LogFactoryImpl.newInstance(LogFactoryImpl.java:601)
        at
org.apache.commons.logging.impl.LogFactoryImpl.getInstance(LogFactoryImpl.java:333)
        at org.apache.commons.logging.LogFactory.getLog(LogFactory.java:664)
        at
org.globus.wsrf.container.ServiceContainer.<clinit>(ServiceContainer.java:111)
        at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
        at
sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:39)
        at
sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:25)
        at java.lang.reflect.Method.invoke(Method.java:585)
        at org.globus.bootstrap.BootstrapBase.launch(BootstrapBase.java:114)
        at org.globus.bootstrap.ContainerBootstrap.main(ContainerBootstrap.java:40)
Starting SOAP server at http://192.168.1.9:8443/wsrf/services/
With the following services:
[1]: http://192.168.1.9:8443/wsrf/services/AdminService
[2]: http://192.168.1.9:8443/wsrf/services/AttachmentTestService
[3]: http://192.168.1.9:8443/wsrf/services/AuthzCalloutTestService
.
…
…..
[65]: http://192.168.1.9:8443/wsrf/services/mds/test/subsource/IndexService
[66]: http://192.168.1.9:8443/wsrf/services/mds/test/subsource/IndexServiceEntry
[67]: http://192.168.1.9:8443/wsrf/services/mds/test/usefulrp/IndexService
[68]: http://192.168.1.9:8443/wsrf/services/mds/test/usefulrp/IndexServiceEntry
2009-02-07T21:38:46.438+01:00 INFO  impl.DefaultIndexService [ServiceThread-
58,performDefaultRegistrations:261]  guid=51312cc0-f557-11dd-9493-8266d3776979
event=org.globus.mds.index.performDefaultRegistrations.end status=0
globus@cpetoshiba:/usr/local/globus-4.2.0.1$ /usr/local/globus-4.2.0.1/start-stop start
```

Observe aquí que si se hace con seguridad las direcciones salen con https y sin seguridad con http sin la s, en este caso se hizo sin seguridad, hay que tener en cuenta colocar solo `localhost` o sino observar en el `container.log` y ver como se estan cargando los servicios.

Después se prueba con el servicio web siguiente:

```
lobo@neo:/usr/local/globus-4.2.0.1$ counter-client -s
http://157.92.26.45:8443/wsrf/services/CounterService
Got notification with value: 3
Counter has value: 3
Got notification with value: 13
```

Y listo todo queda funcionado.

Configuración del Servicio Reliable File Transfer (RFT)

La guia se consigue en [9]

Lo primero es mirar si existe postgresql instalado, se recomienda una versión superior a la 7.1

```
lobo@neo:~$ apt-show-versions -a -p postgresql
postgresql      7.5.22  install ok installed
postgresql      7.5.22  etch
postgresql/etch uptodate 7.5.22
lobo@neo:~$
```

Luego se configura el demonio postmaster para que acepte las conecciones TCP:

```
neo:~# vim /etc/postgresql/7.4/main/postgresql.conf

#---------------------------------------------------------------------------
# CONNECTIONS AND AUTHENTICATION
#---------------------------------------------------------------------------

# - Connection Settings -

tcpip_socket = true
max_connections = 100
        # note: increasing max_connections costs about 500 bytes of shared
        # memory per connection slot, in addition to costs from shared_buffers
        # and max_locks_per_transaction.
#superuser_reserved_connections = 2
port = 5432
```

En este archivo antes estaba desactivado el `tcpip_socket=false`, se lo cambio por `tcpip_socket=true`.

Ahora se necesita crear un usuario PostgreSQL que conectará la base de datos. Es en esta cuenta bajo la cual el contenedor esta corriendo. Si crea el usuario PostgreSQL corriendo el comando: `su postgres; createuser globus` y obtiene el error: `psql: could not connect to server: No such file or directory Is the server running locally and accepting connections on Unix domain socket "/tmp/.s.PGSQL.5432"?` esto generalemnte significa que tu postmaster no ha iniciado con la opción `-I` o no se ha reiniciado el postmaster despues del mencionado paso anterior.

```
neo:~# su postgres; createuser globus
postgres@neo:/root$ createuser globus
Shall the new user be allowed to create databases? (y/n) y
Shall the new user be allowed to create more new users? (y/n) y
CREATE USER
postgres@neo:/root$
```

Ahora es necesario configurar la seguridad en la base de datos:

```
neo:~# vi /var/lib/postgresql/7.4/main/pg_hba.conf

# IPv6-style local connections:
host rftDatabase "globus" "157.92.26.45" 255.255.255.255 md5

neo:~# /etc/init.d/postgresql-7.4 restart
```

```
Restarting PostgreSQL 7.4 database server: main.
neo:~#
```

Crear la base de datos que es usada para correr RTF (como usuaro globus)

```
globus@neo:/root$ createdb rftDatabase
CREATE DATABASE
globus@neo:/root$

globus@neo:/$       psql       -d       rftDatabase       -f       /usr/local/globus-
4.2.0.1/share/globus_wsrf_rft/rft_schema.sql
psql:/usr/local/globus-4.2.0.1/share/globus_wsrf_rft/rft_schema.sql:6:  NOTICE:   CREATE
TABLE / PRIMARY KEY will create implicit index "requestid_pkey" for table "requestid"
CREATE TABLE
psql:/usr/local/globus-4.2.0.1/share/globus_wsrf_rft/rft_schema.sql:11: NOTICE:  CREATE
TABLE / PRIMARY KEY will create implicit index "transferid_pkey" for table "transferid"
CREATE TABLE
psql:/usr/local/globus-4.2.0.1/share/globus_wsrf_rft/rft_schema.sql:31: NOTICE:  CREATE
TABLE / PRIMARY KEY will create implicit index "request_pkey" for table "request"
CREATE TABLE
psql:/usr/local/globus-4.2.0.1/share/globus_wsrf_rft/rft_schema.sql:66: NOTICE:  CREATE
TABLE / PRIMARY KEY will create implicit index "transfer_pkey" for table "transfer"
CREATE TABLE
psql:/usr/local/globus-4.2.0.1/share/globus_wsrf_rft/rft_schema.sql:72: NOTICE:  CREATE
TABLE / PRIMARY KEY will create implicit index "restart_pkey" for table "restart"
CREATE TABLE
CREATE TABLE
CREATE INDEX
globus@neo:/$
```

Despues configuramos el siguiente archivo:

```
globus@neo:/root$ vim /usr/local/globus-4.2.0.1/etc/globus_wsrf_rft/jndi-
config.xml

</parameter>
            <parameter>
                <name>
                password
                </name>
                <value>
                *****
```

(En este caso globusdb)

Creada la base de datos, cargar el esquema RFT, y cambiar el password en el archivo `jndi-config.xml`. Si la base de datos no es poseida por el mismo usuario como el container, podria necesitar cambiar el parametro username en el `jndi-config.xml`. En este ejemplo, se instalaron como globus y se hizo la base de datos como globus, por tanto solo se cambia el password.

La base de datos esta creada, ahora se reinicia el contenedor para cargar la nueva configuración RFT:

```
neo:~# /usr/local/globus-4.2.0.1/start-stop stop
Stopping Globus container. PID: 2737
Container stopped
neo:~#
neo:~# /usr/local/globus-4.2.0.1/start-stop start
Starting Globus container. PID: 21135
neo:~# head /usr/local/globus-4.2.0.1/var/container.log
```

```
Starting SOAP server at http://157.92.26.45:8443/wsrf/services/

With the following services:

[1]: http://157.92.26.45:8443/wsrf/services/AdminService
[2]: http://157.92.26.45:8443/wsrf/services/AttachmentTestService
[3]: http://157.92.26.45:8443/wsrf/services/AuthzCalloutTestService
[4]: http://157.92.26.45:8443/wsrf/services/CASService
[5]: http://157.92.26.45:8443/wsrf/services/ContainerRegistryEntryService
[6]: http://157.92.26.45:8443/wsrf/services/ContainerRegistryService
neo:~#
```

Ahora se intentará una transferencia RFT para estar seguros que el servicio realmente funciona:

```
lobo@portal:/$ cp /usr/local/globus-4.2.0.1/share/globus_wsrf_rft_test/transfer.xfr
/tmp/rft.xfr
lobo@portal:/$ vim /tmp/rft.xfr

true
16000
16000
false
1
true
1
null
null
false
10
gsiftp://localhost:2811/tmp/prueba
gsiftp://localhost:2811/tmp/rftTest_Done.tmp
```

Despues se ejecuta la siquiente linea para probar el RFT:

```
lobo@portal:/tmp$ rft -h portal.grid.lsc.dc.uba.ar -f /tmp/rft.xfr
Number of transfers in this request: 1
Subscribed for overall status
Termination time to set: 60 minutes

 Overall status of transfer:
Finished/Active/Failed/Retrying/Pending
0/1/0/0/0

 Overall status of transfer:
Finished/Active/Failed/Retrying/Pending
1/0/0/0/0
All Transfers are completed


lobo@portal:/tmp$ diff prueba rftTest_Done.tmp
lobo@portal:/tmp$
```

Si salta algún error se puede hacer lo siguiente:

Se cambia el nombre del host en el archivo `/etc/hosts` así:

```
lobo@portal:/$ vim /etc/hosts

#127.0.1.1      neo.exp.dc.uba.ar       neo
157.92.26.45    portal.grid.lsc.dc.uba.ar       portal  localhost
```

```
#127.0.0.1      portal localhost

# The following lines are desirable for IPv6 capable hosts
::1     ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
ff02::3 ip6-allhosts
```

De tal forma que ahora ya no es `neo.exp.dc.uba.ar` si no que lo cambiamos a `portal.grid.lsc.dc.uba.ar` porque en el DNS la maquina estaba configurada como portal y para evitar molestias se deja todo igual, tanto en el DNS como en la máquina.

La parte de portal localhost, especialemte la de localhost es para que los servicios de RFT funcionen correctamente ya que así detecta el localhost de la máquina.

De ahora en adelante todo se sigue haciendo con seguridad osea que el scrip de start-stop no colocamos la parte de -nosec.

Configurar el WS GRAM

```
portal:~# cat /etc/sudoers
# /etc/sudoers
#
# This file MUST be edited with the 'visudo' command as root.
#
# See the man page for details on how to write a sudoers file.
#

Defaults        env_reset

# Host alias specification

# User alias specification
User_Alias ADMIN = dfslezak, lobo, franeijo

# Cmnd alias specification

# User privilege specification
root    ALL=(ALL) ALL
ADMIN   ALL=(ALL) ALL
globus  ALL=(bacon) NOPASSWD: /usr/local/globus-4.2.0.1/libexec/globus-gridmap-and-
execute -g /etc/grid-security/grid-mapfile /usr/local/globus-4.2.0.1/libexec/globus-
job-manager-script.pl *
globus  ALL=(bacon) NOPASSWD: /usr/local/globus-4.2.0.1/libexec/globus-gridmap-and-
execute -g /etc/grid-security/grid-mapfile /usr/local/globus-4.2.0.1/libexec/globus-
gram-local-proxy-tool *
portal:~#
```

Otro ejemplo con globusrun-ws:

```
lobo@portal:/tmp$ ls
hsperfdata_lobo  prueba            rft.xfr
hsperfdata_root  rftTest_Done.tmp  x509up_u1002
lobo@portal:/tmp$ globusrun-ws -submit -c /bin/cp /tmp/prueba /tmp/pruebacopiada
Submitting job...Done.
Job ID: uuid:fb84970a-9ae1-11dd-b440-0090278d3632
Termination time: 10/15/3008 17:52 GMT
```

```
Current job state: Active
Current job state: CleanUp
Current job state: Done
Destroying job...Done.
lobo@portal:/tmp$ cd /tmp
lobo@portal:/tmp$ ls
hsperfdata_lobo  prueba         rftTest_Done.tmp  x509up_u1002
hsperfdata_root  pruebacopiada  rft.xfr
lobo@portal:/tmp$
```

Finalmente ya queda configurada la máquina con lo necesario para formar parte de la grid.

Referencias

[1] Debian, Software in the Public Interest, Inc. [En Línea]. Disponible: www.debian.org [Consulta: marzo, 2009]

[2] The Globus Alliance, Globus, 2008. [En línea]. Disponible: http://www.globus.org/ [Consulta: marzo, 2009].

[3] Chapter 1. 4.0.x quickstart, The Globus Alliance, Globus, 1997. [En línea]. Disponible: http://www.globus.org/toolkit/docs/4.0/admin/docbook/quickstart.html#q-options [Consulta: marzo, 2009].

[4] Sun Microsystems, 1994. [En línea]. Disponible: https://cds.sun.com/is-bin/INTERSHOP.enfinity/WFS/CDS-CDS_Developer-Site/en_US/-/USD/ViewFilteredProducts-SingleVariationTypeFilter [Consulta: marzo, 2009].

[5] Archive.apache.org, 2009. [En línea]. Disponible: http://archive.apache.org/dist/ant/binaries/ [Consulta: marzo, 2009].

[6] W. Mulder, A.M. Kuipers “How To Install Globus And Write A Hello World Service,” Versión 0.22, 2004.

[7] Chapter 6. Basic Security Configuration, The Globus Alliance, Globus, 1997. [En línea]. Disponible: http://www.globus.org/toolkit/docs/4.0/admin/docbook/ch06.html [Consulta: marzo, 2009].

[8] Globus Toolkit 3.2, Installation Guide, The Globus Alliance, Globus, 1997. [En línea]. Disponible: https://www.globus.org/toolkit/docs/3.2/installation/install_config_gridftp.html [Consulta: marzo, 2009].

[9] GT 4.0 Reliable File Transfer (RFT) Service: System Administrator's Guide Installation Guide, The Globus Alliance, Globus, 1997. [En línea]. Disponible: http://www.globus.org/toolkit/docs/4.0/data/rft/admin-index.html [Consulta: marzo, 2009].

Anexo E: Crear servicios con Introduce

Este anexo ofrece una descripción del manejo de la herramienta introduce mediante un ejemplo, para la creación y despliegue de servicios Web en *Grid* [1].

Se parte del hécho que la herramienta Introduce ya esta instalada en el equipo, por tanto se comienza el proceso de la siguiente manera:

- Abra introduce mediante el siguiente comando:

```
root@cpehva:~# cd /usr/local/introduce

root@cpehva:/usr/local/introduce# ant introduce
```

En el momento de iniciar Introduce se observa una imagen como el de la figura E1.

Figura E1: Inicio de Introduce

Inmediatamente después aparece la siguiente imagen, haga clic en Create Service Skeleton, Figura E2 y E2.

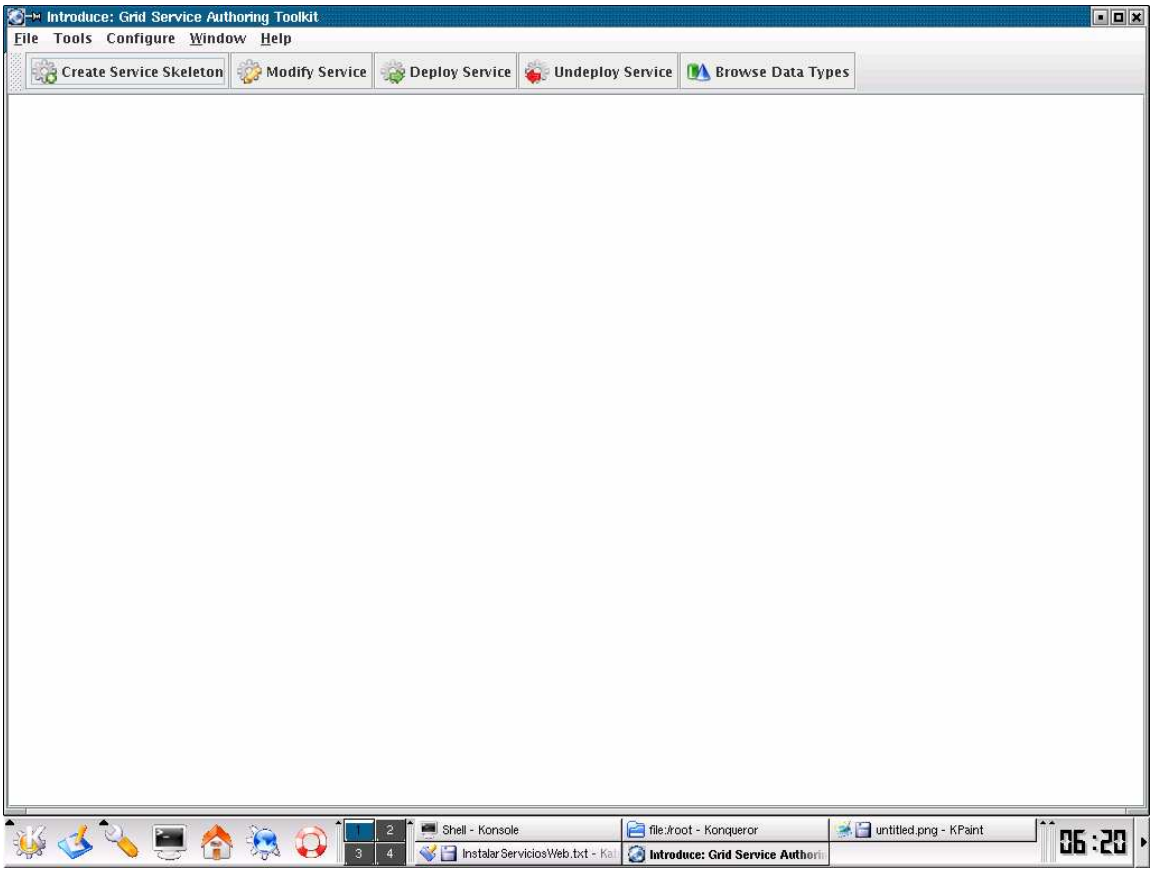

Figura E2: Inicio

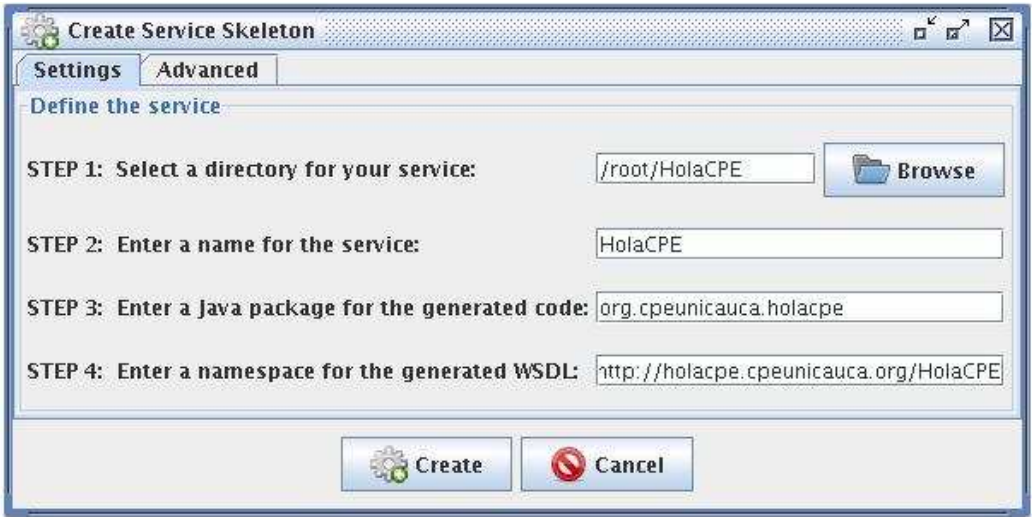

Figura E3: Create Service Skeleton

Haga clic en Create para iniciar con el proceso de creación del primer servicio web.

A continuación aparece la ventana de creación de servicios web de introduce, figura E3, la cual permite crear nuevas operaciones de los servicios y para esas operaciones utilizar tipos simples de datos (int, string, boolean, Integer, entre otros) y tipos complejos (xsd) en este caso se va a crear un servicio muy simple el cual recibe un nombre y lo retorno junto con la palabra Hola.

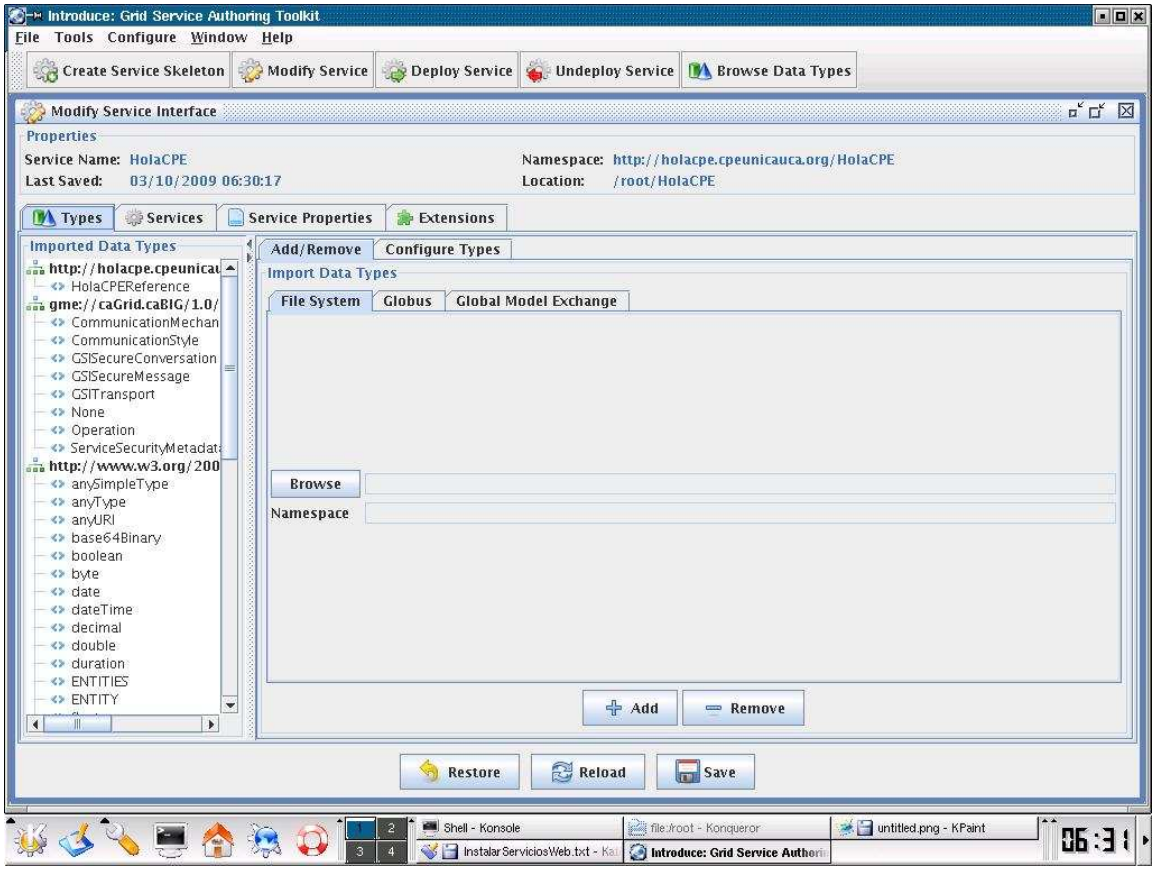

Figura E3: Creación de servicios web de introduce.

Haga clic en la pestaña Services y haga clic sobre Operations como se muestra en la siguiente figura E4.

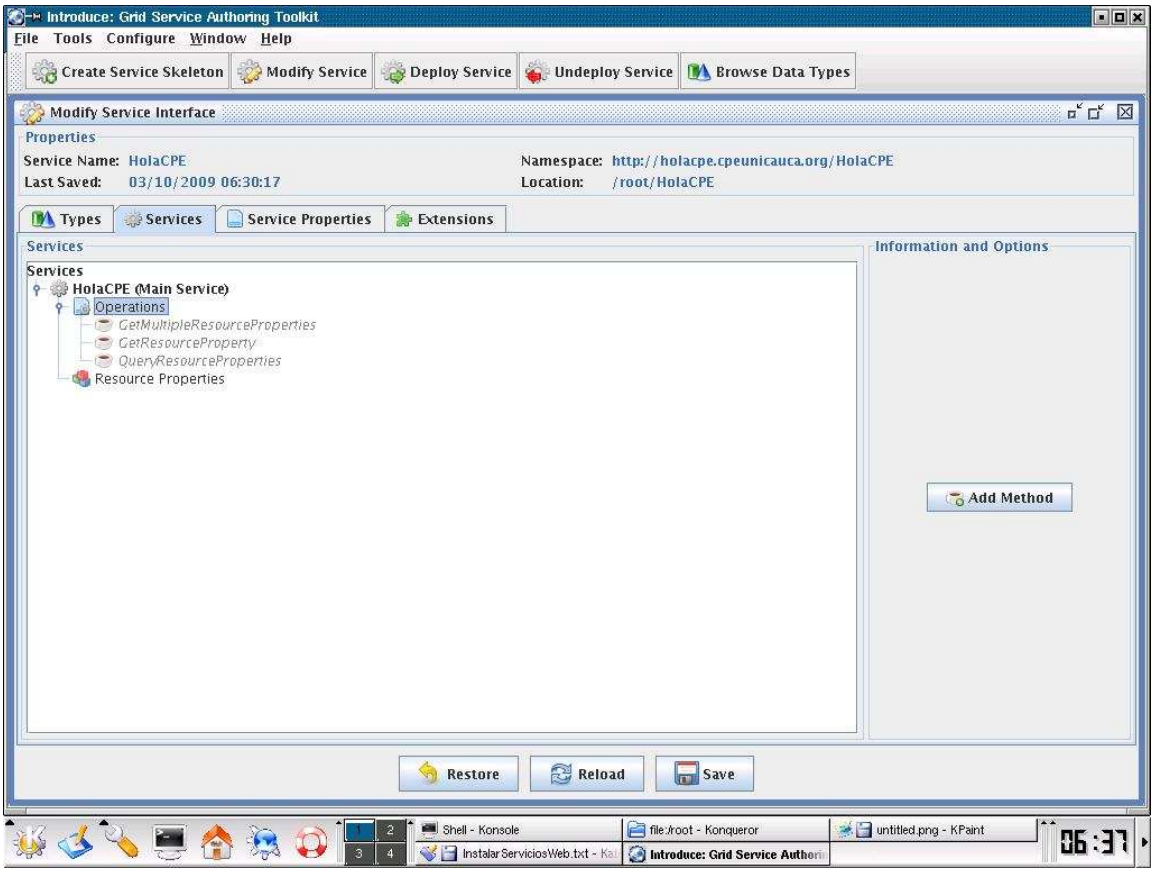

Figura E4: Services

Haga clic en el botón Add Method ver figura E5.

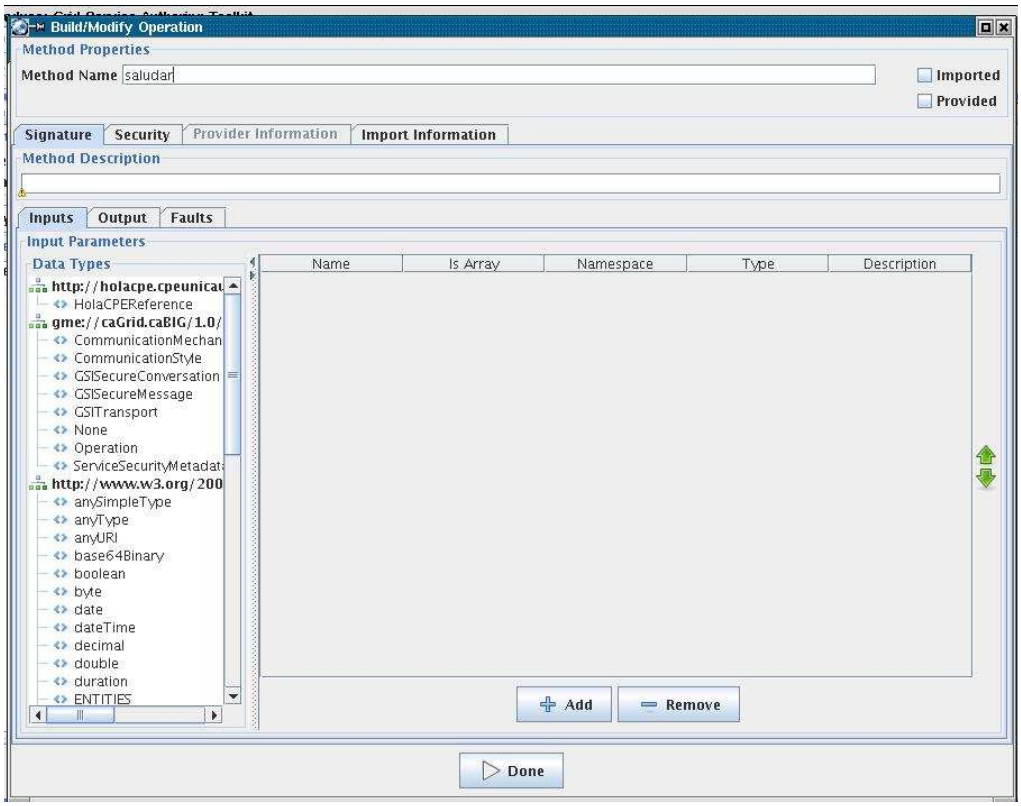

Figura E5: Add Method

Luego de la parte izquierda en la pestaña inputs seleccione el tipo de dato String y haga clic en el botón + Add figura E6.

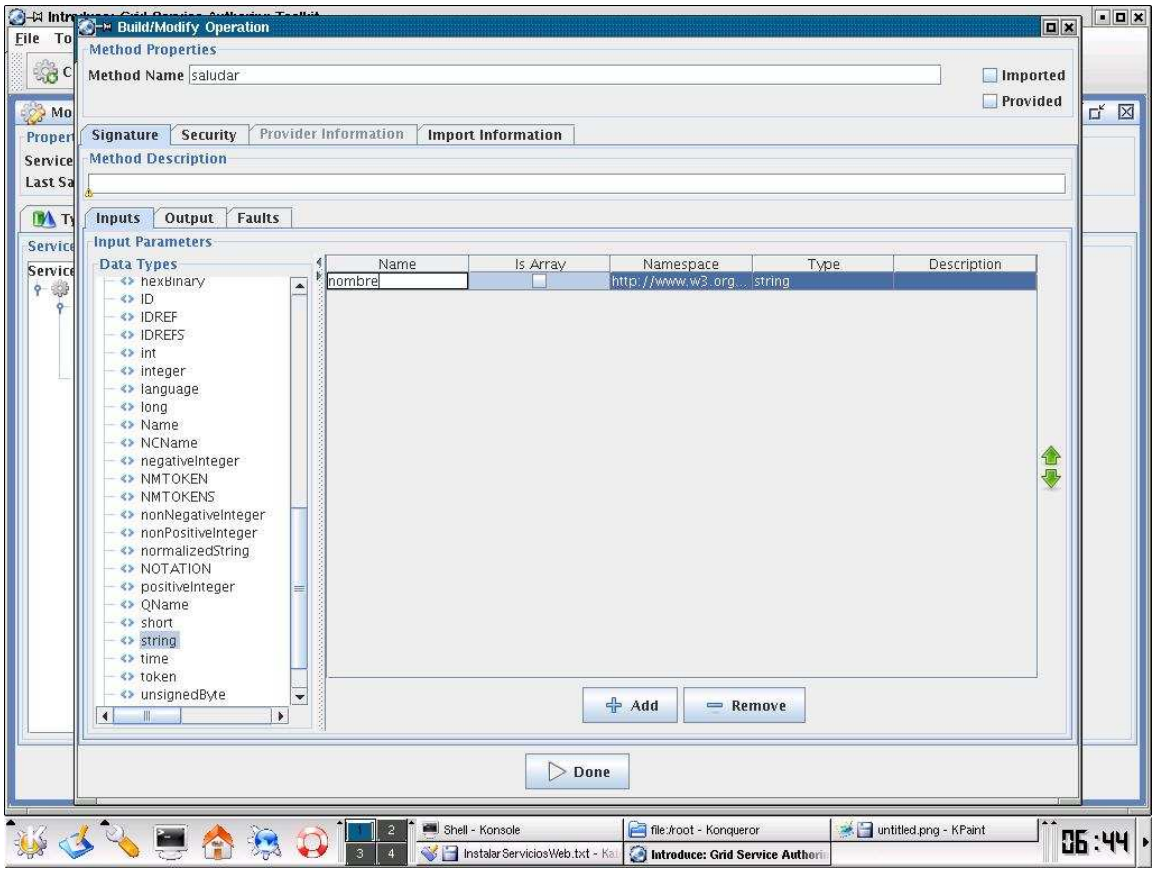

Figura E6: + Add

En name escriba nombre y haga clic en la pestaña Output, una vez ahí seleccione de la parte izquierda el tipo de dato string y haga doble clic sobre él, la ventana debe quedar como sigue figura E7.

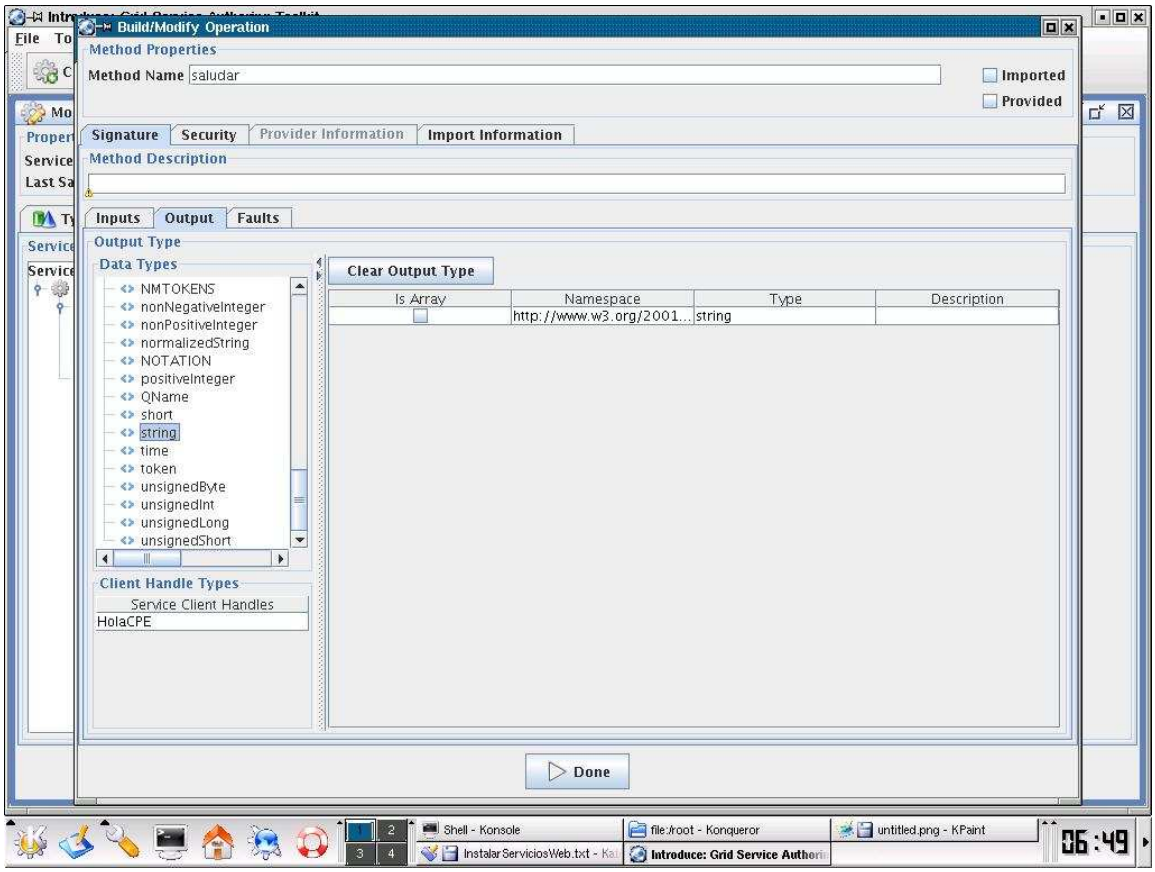

Figura E7: String.

Haga clic en Done y dentro de las operaciones del servicio aparecerá la nueva operación saludar, figura E8.

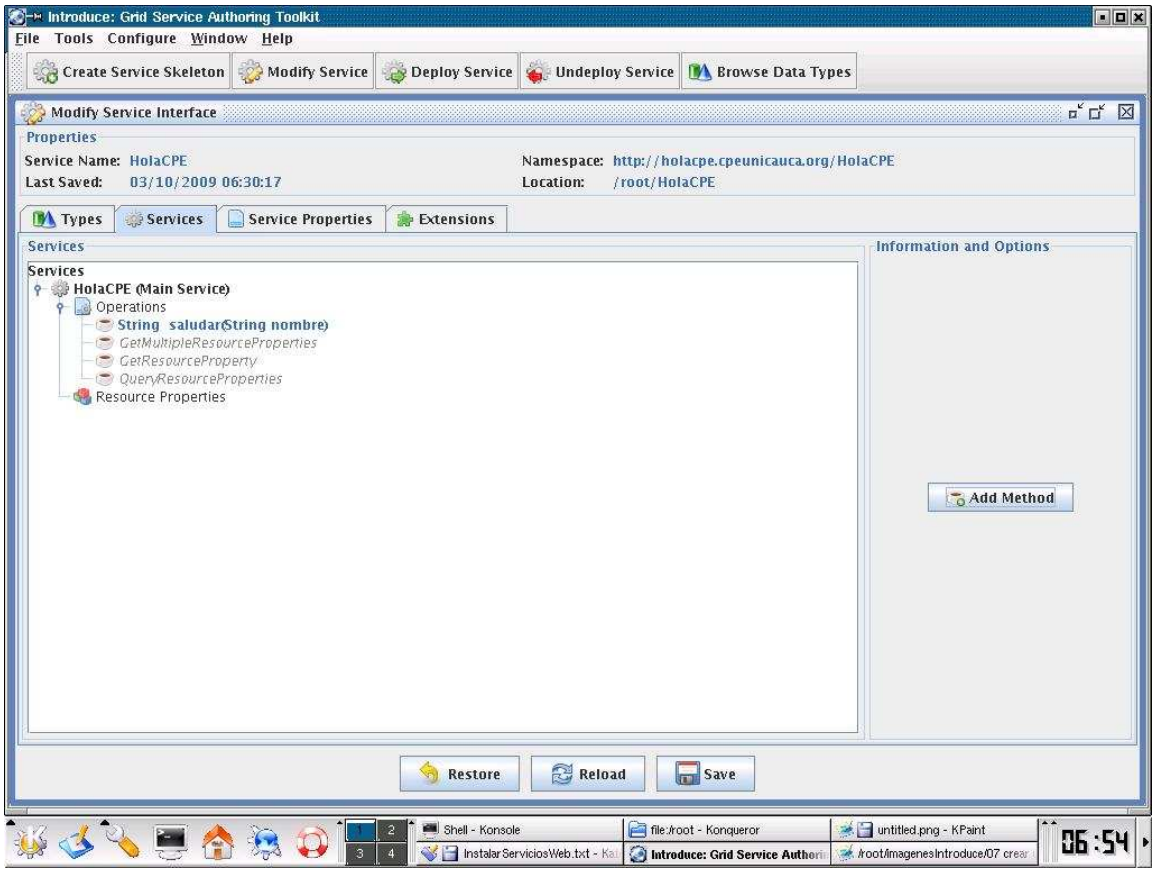

Figura E8: operación saludar.

Haga clic en guardar y sale un cuadro de diálogo de confirmación haga clic en Yes y espere a que la ventana de progreso se cierre sola.

Ingrese a la carpeta de la aplicación

`cd /root/HolaCPE` y ejecute el comando `ant all`

```
/root/HolaCPE/ant all
```

En el caso de este ejemplo se encuentra alojada `/root/HolaCPE`, se debe tomar ese proyecto y abrirlo con un IDE de java para realizar la implementación de la operación saludar. En este ejemplo se utilizará Eclipse figura E8.

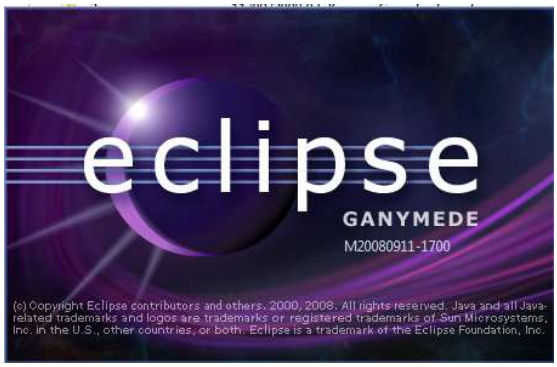

Figura E8: Inicio de Eclipse.

Abra eclipse y seleccione el menú File -> Import, seleccione luego Existing Projects into Workspace, figura E9.

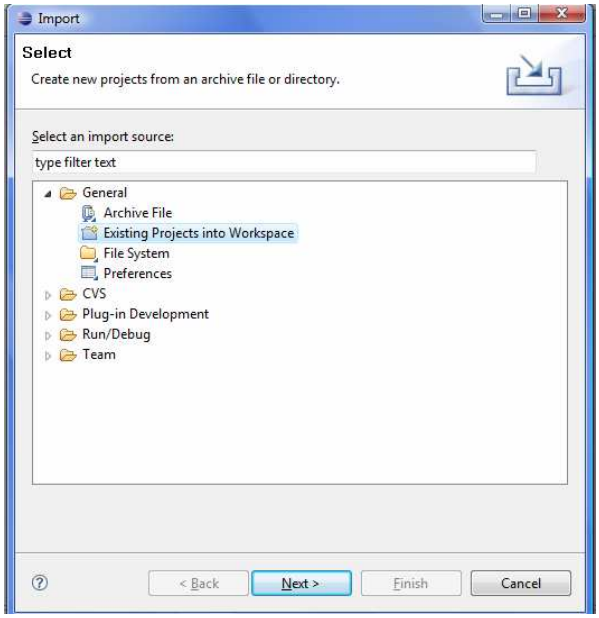

Figura E9: Existing Projects into Workspace.

Haga clic en Next y luego en Browse y seleccione la ruta del ejemplo, en este caso /root/HolaCPE, entonces haga clic en ok y luego en Finish

En el explorador de paquetes de eclipse abra el proyecto HolaCPE y busque la carpeta src y luego el paquete org.cpeunicauca.holacpe.service y abra la clase HolaCPEImpl.java. figura E10.

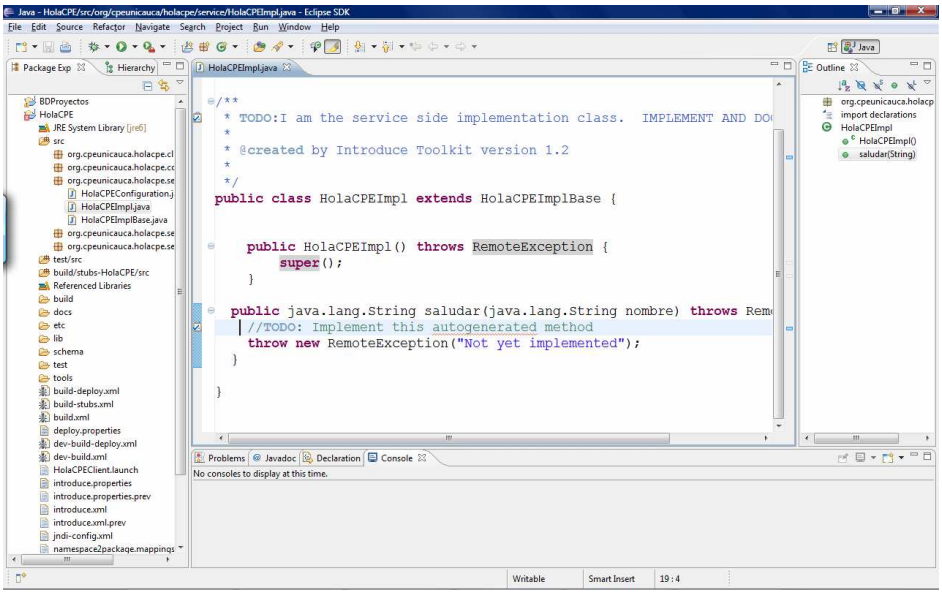

Figura E10: HolaCPEImpl.java.

En el método saludar borre las líneas

```
//TODO: Implement this autogenerated method
    throw new RemoteException("Not yet implemented");
```

y a continuación inserte el siguiente código:

```
return "Hola "+nombre;
```

Su clase debe quedar como sigue

```
package org.cpeunicauca.holacpe.service;

import java.rmi.RemoteException;

/**
 * TODO:I am the service side implementation class.  IMPLEMENT AND DOCUMENT ME
 *
 * @created by Introduce Toolkit version 1.2
 *
 */
public class HolaCPEImpl extends HolaCPEImplBase {


	public HolaCPEImpl() throws RemoteException {
		super();
	}

  public java.lang.String saludar(java.lang.String nombre) throws RemoteException {
    return "Hola "+nombre;
  }

}
```

Guarde cambios y ejecute los siguientes comandos:

`cd /root/HolaCPE` y ejecute el comando `ant all`:

```
/root/HolaCPE/ant all
```

Si todo sale bien debe obtener un mensaje de `BUILD SUCCESSFULL`

Luego regrese a introduce y haga clic en el botón Deploy Service y seleccione la carpeta `/root/HolaCPE` (Asegúrese que el contenedor de servicios web NO esté corriendo) figura E11.

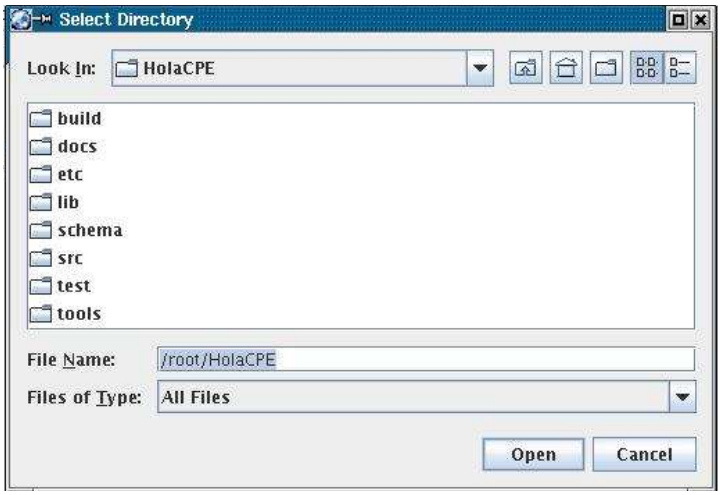

Figura E11: Deploy Service.

Haga clic en Open y a continuación clic en Deploy y espere a que la ventana de progreso desaparezca sola, figura E12.

Figura E12: Deploy.

Una vez terminada esta parte es necesario cambiar los permisos del archivo de despliegue ubicado en `/usr/local/globus-4.0.8/etc/` mediante el comando:

```
root@cpehva:~# chmod 777 -R /usr/local/globus-4.0.8/etc/cagrid_HolaCPE/
```

Finalmente iniciamos el contenedor de servicios web de globus con el comando:

```
root@cpehva:~# su globus

globus@cpehva:/root$ /etc/init.d/globus-4.0.8 start

Password:

Starting Globus container. PID: 12446

globus@cpehva:/root$
```

El archivo descriptor WSDL del servicio queda alojado en la dirección http://192.168.1.14:8443/wsrf/services/cagrid/HolaCPE?wsdl desde un equipo cliente cualquiera abra un navegador e ingrese esta dirección obtendrá el WSDL siguiente.

```
<?xml version="1.0" encoding="UTF-8" ?>
- <wsdl:definitions name="HolaCPE"
    targetNamespace="http://holacpe.cpeunicauca.org/HolaCPE/service"
    xmlns:binding="http://holacpe.cpeunicauca.org/HolaCPE/bindings"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
  <wsdl:import
    location="http://192.168.1.14:8443/share/schema/HolaCPE/HolaCPE_bindings.wsdl"
    namespace="http://holacpe.cpeunicauca.org/HolaCPE/bindings" />
- <wsdl:service name="HolaCPEService">
- <wsdl:port binding="binding:HolaCPEPortTypeSOAPBinding" name="HolaCPEPortTypePort">
  <soap:address location="http://192.168.1.14:8443/wsrf/services/cagrid/HolaCPE" />
    </wsdl:port>
    </wsdl:service>
    </wsdl:definitions>
```

Ahora lo que resta es crear un cliente para el servicio web, para el caso del presente ejemplo se utilizó el IDE NetBeans para crear un cliente para el servicio web.

En NetBeans haga clic en Archivo -> Proyecto Nuevo. De la categoría Java seleccione Aplicación Java, figura E13.

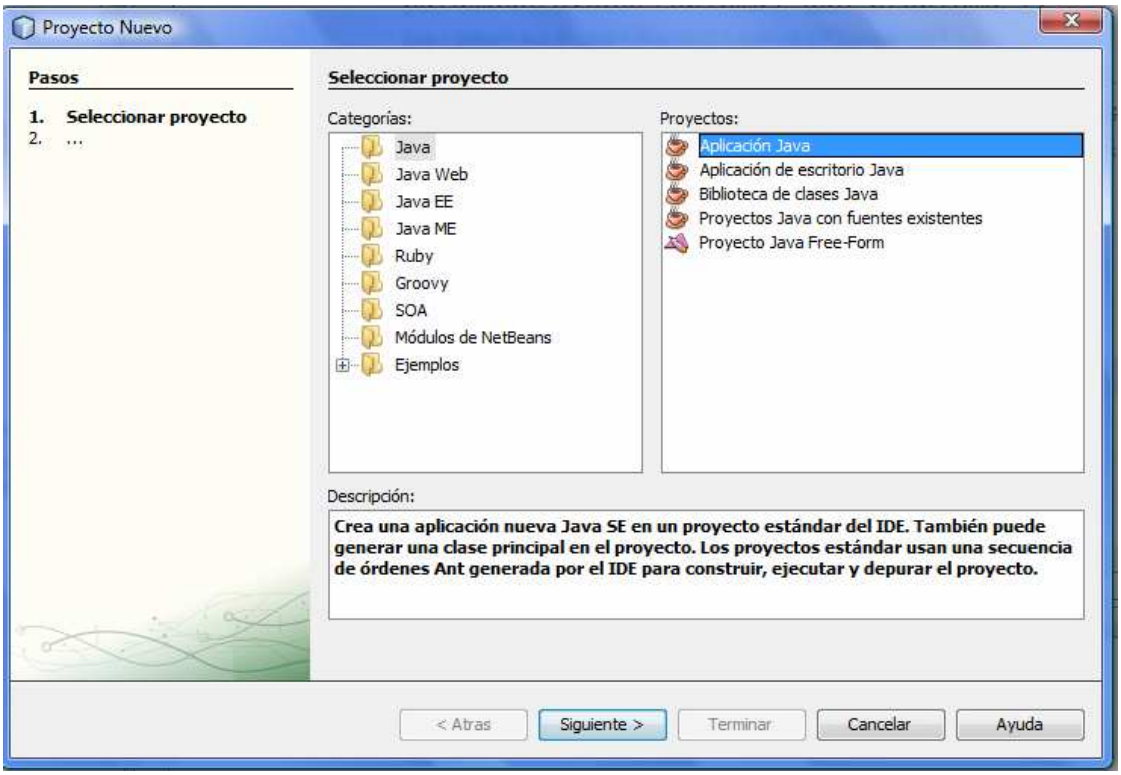

Figura E13: Aplicación Java.

Haga clic en siguiente y nombre al proyecto como ClineteHolaCPE y haga clic en Terminar, figura E14.

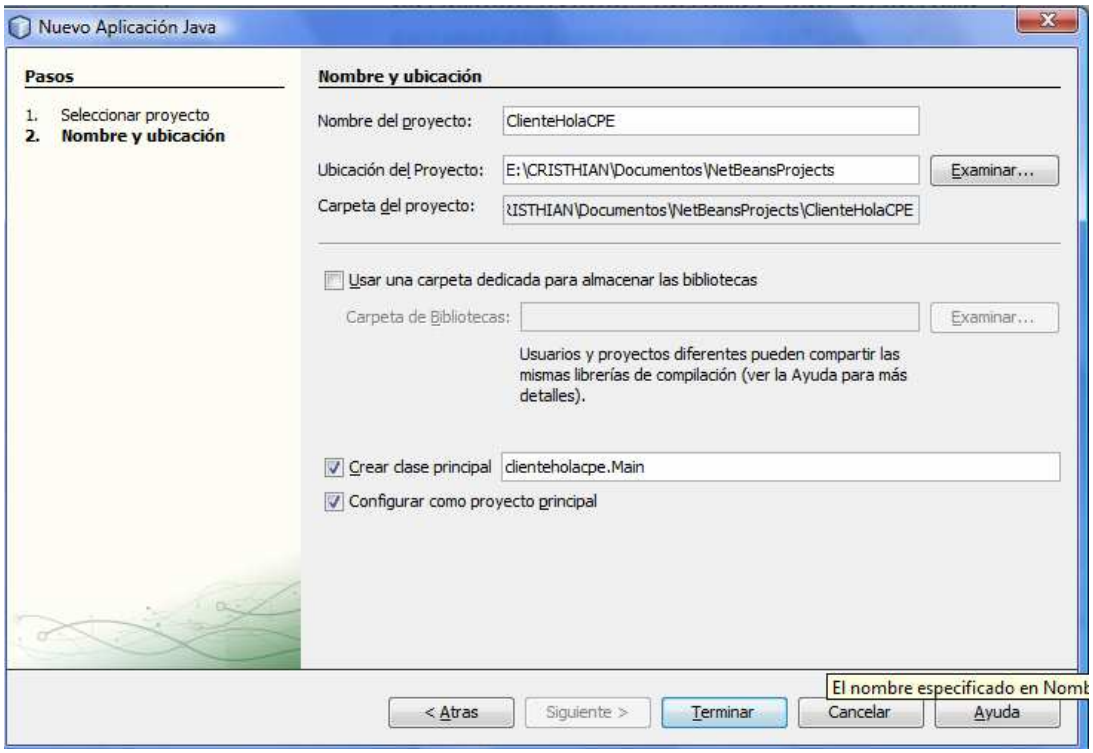

Figura E14: ClineteHolaCPE.

Una vez creado el proyecto haga clic derecho sobre el nodo del proyecto y seleccione Nuevo -> Cliente Servicio Web. Figura E15.

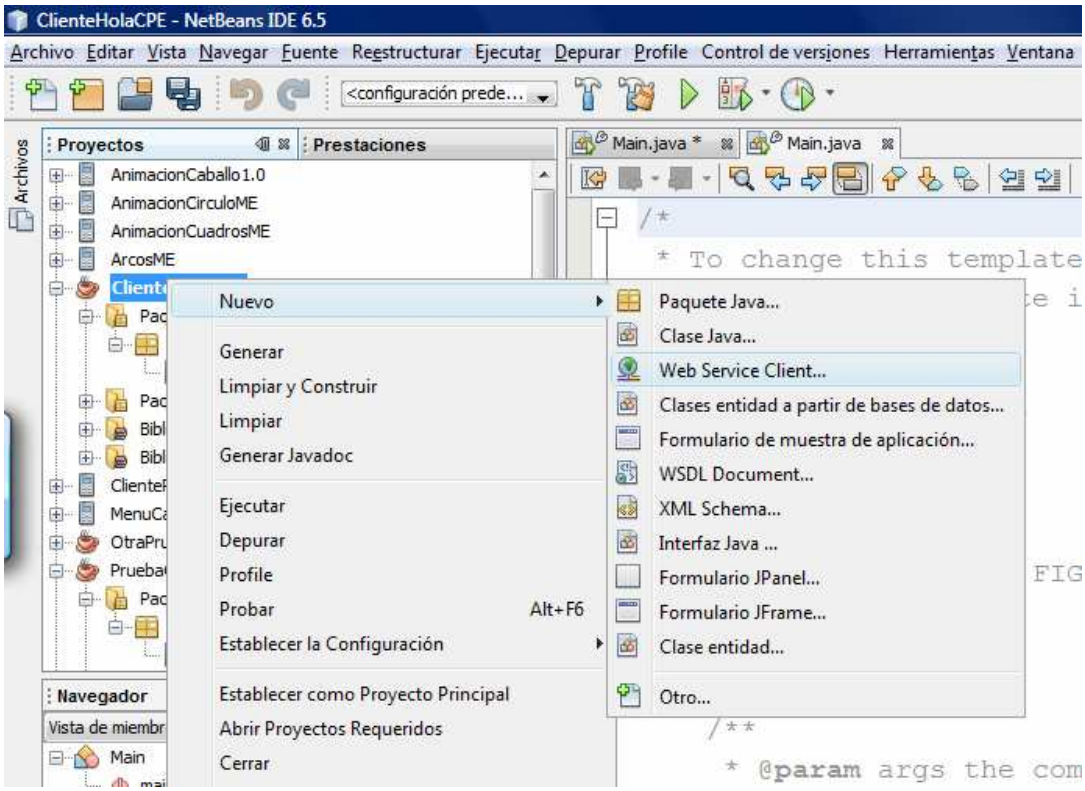

Figura E15: Cliente Servicio Web.

En la ventana emergente seleccione WSDL URL y pegue la dirección http://192.168.1.14:8443/wsrf/services/cagrid/HolaCPE?wsdl en el campo de texto y haga clic en terminar. Figura E16.

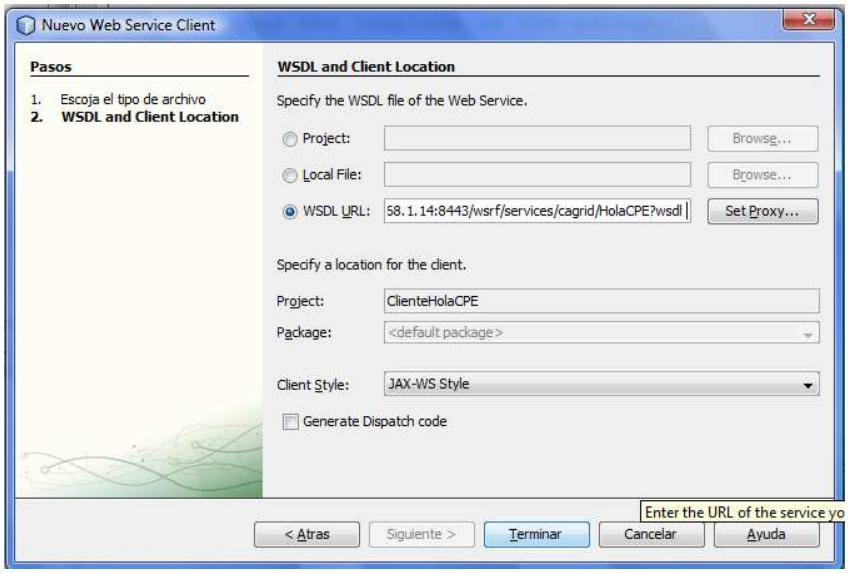

Figura E16: Terminar

A continuación abra la clase Main y dentro del método main borre la línea

```
// TODO code application logic here
```

Y haga clic derecho justo donde estaba ese texto y seleccione Web Service Client Resources -> Call Web Service Operation, figura E17.

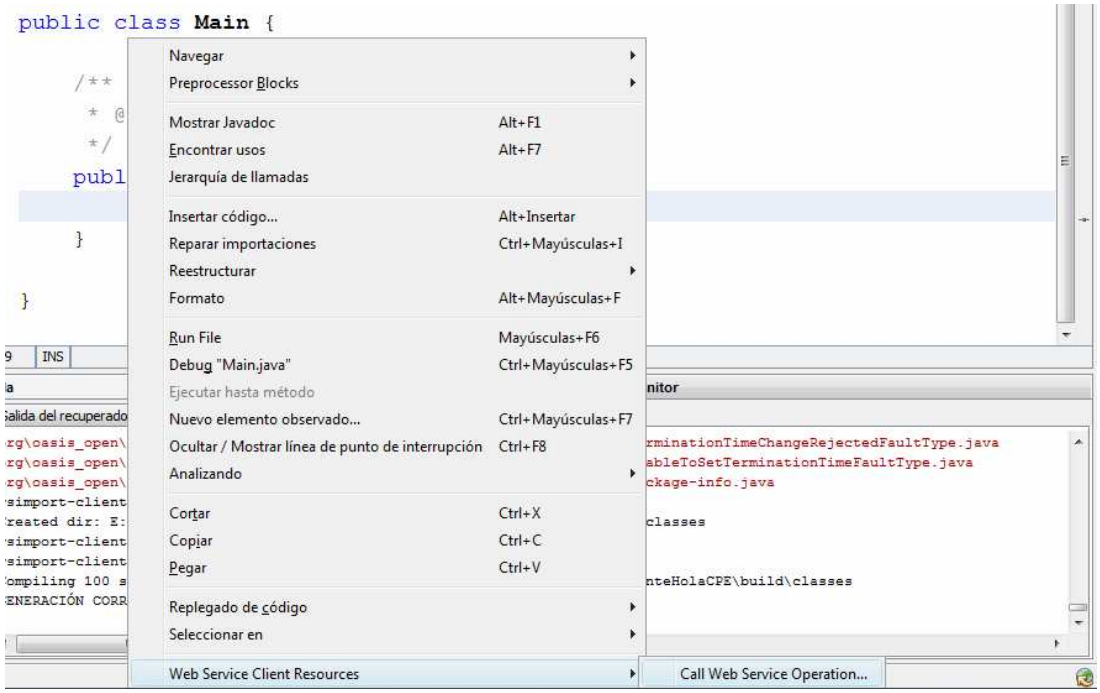

Figura E17: Call Web Service Operation

De la ventana emergente seleccione el método saludar del proyecto HolaCPE así, figura E18:

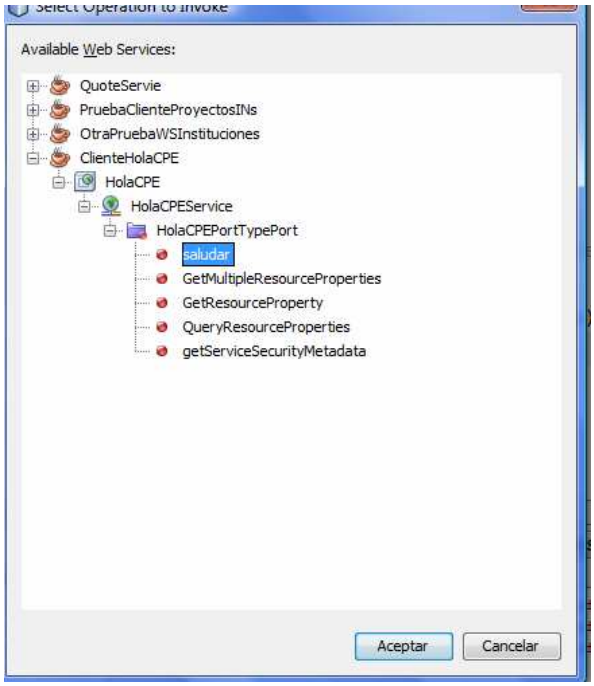

Figura E18: HolaCPE

Haga clic en aceptar y se genera el siguiente código en el método main

```
    org.cpeunicauca.holacpe.holacpe.service.HolaCPEService service = new
org.cpeunicauca.holacpe.holacpe.service.HolaCPEService();

            org.cpeunicauca.holacpe.holacpe.service.HolaCPEPortType port =
service.getHolaCPEPortTypePort();

            // TODO initialize WS operation arguments here

            org.cpeunicauca.holacpe.holacpe.SaludarRequest parameters = new
org.cpeunicauca.holacpe.holacpe.SaludarRequest();

            // TODO process result here

            org.cpeunicauca.holacpe.holacpe.SaludarResponse result =
port.saludar(parameters);

            System.out.println("Result = "+result);

        } catch (Exception ex) {

            // TODO handle custom exceptions here

        }
```

En la línea `// TODO process result here` ingrese `parameters.setNombre(“Formador CPE”);`

Y en `result` adicione `.getResponse.`

Su clase debe quedar así:

```
package clienteholacpe;


/**

 *

 * @author CRISTHIAN N FIGUEROA

 */

public class Main {


    /**

     * @param args the command line arguments

     */

    public static void main(String[] args) {


        try { // Call Web Service Operation

            org.cpeunicauca.holacpe.holacpe.service.HolaCPEService service = new
org.cpeunicauca.holacpe.holacpe.service.HolaCPEService();

            org.cpeunicauca.holacpe.holacpe.service.HolaCPEPortType port =
service.getHolaCPEPortTypePort();

            // TODO initialize WS operation arguments here

            org.cpeunicauca.holacpe.holacpe.SaludarRequest parameters = new
org.cpeunicauca.holacpe.holacpe.SaludarRequest();

            parameters.setNombre("Formador CPE");

            org.cpeunicauca.holacpe.holacpe.SaludarResponse result =
port.saludar(parameters);

            System.out.println("Resultado de la invocación al servicio web = " +
result.getResponse());

        } catch (Exception ex) {

            // TODO handle custom exceptions here

        }

    }

}
```

Ejecute su proyecto y en la consola de salida deberá observar el siguiente mensaje

```
Compiling 100 source files to
E:\CRISTHIAN\Documentos\NetBeansProjects\ClienteHolaCPE\build\classes

Compiling 1 source file to
E:\CRISTHIAN\Documentos\NetBeansProjects\ClienteHolaCPE\build\classes

compile:

run:

Resultado de la invocación al servicio web  = Hola Formador CPE

GENERACIÓN CORRECTA (tiempo total: 7 segundos)
```

Si se desea utilizar tipos complejos de datos (xsd) por favor refiérase a la guía de introduce.

Crear un servicio web en la grid de globus con acceso a Bases de Datos

En el presente ejemplo se utiliza postgreSQL como motor de bases de datos y el método de acceso a través de JDBC es DAO.

La siguiente figura E19 muestra el diagrama entidad-relación de la base de datos del ejemplo.

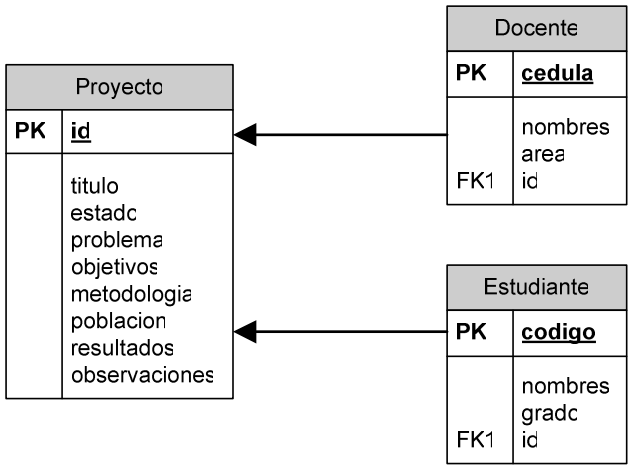

Figura E19: diagrama entidad-relación.

Primero que todo se debe crear el servicio de igual manera como se creó el ejemplo, solo que en este caso el servicio se va a llamar ProyectosPrueba. Cree el proyecto dentro de la carpeta `/root/ProyectosPrueba`. Figura E20.

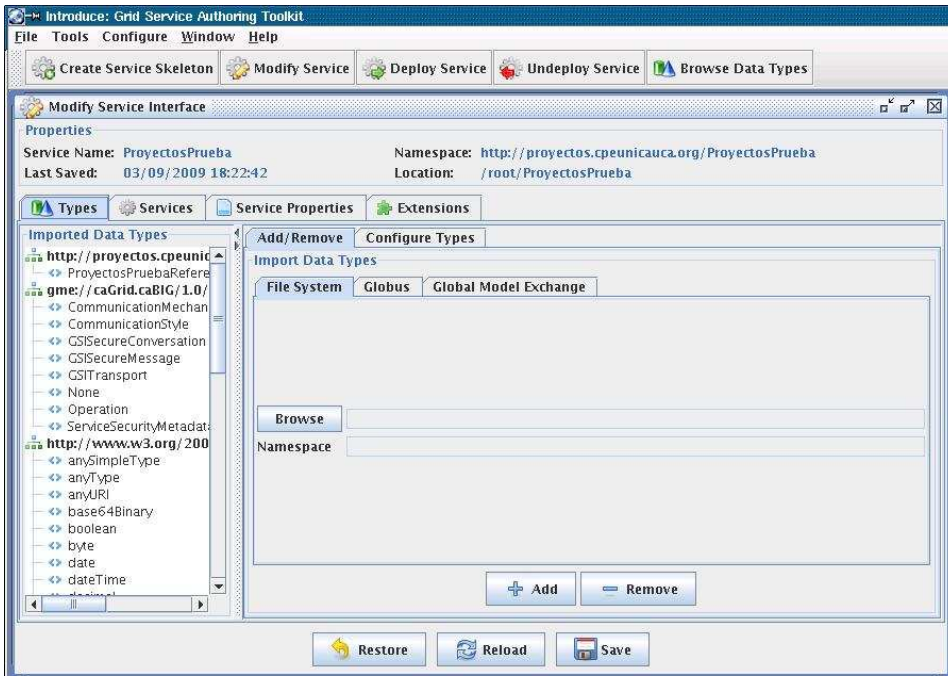

Figura E20: ProyectosPrueba.

Una vez creado el servicio se deben adicionar las operaciones siguientes, de tal manera que su servicio quede así, figura E21:

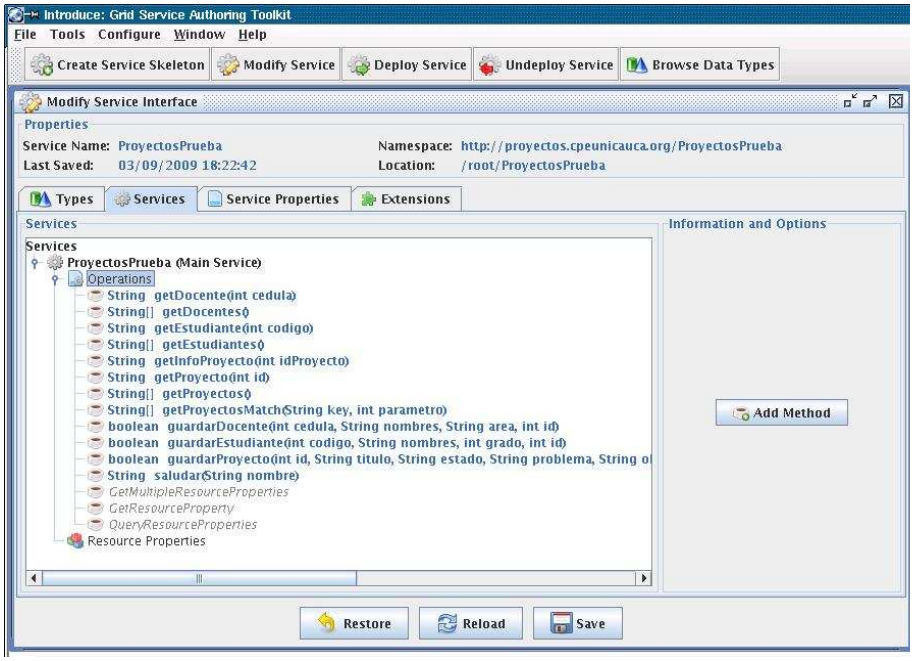

Figura E21: operaciones.

Haga clic en el botón guardar y abra eclipse para editar el proyecto de igual manera que en el ejemplo anterior. File->Import-> Existing Projects into Workspace, figura E22.

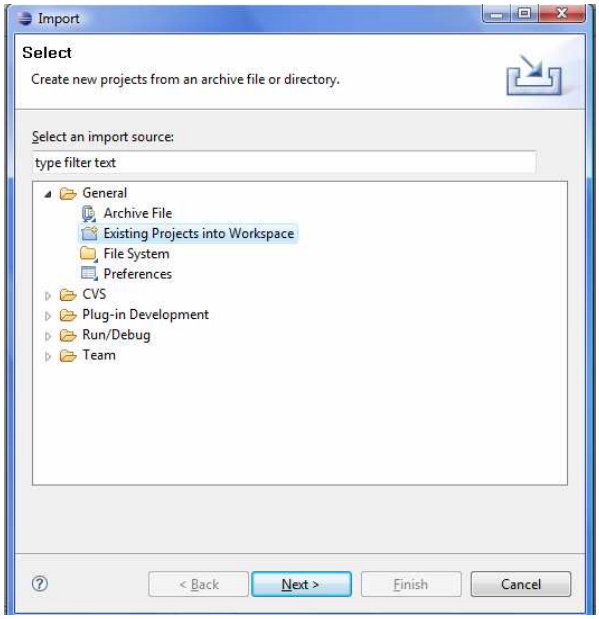

Figura E22: Existing Projects into Workspace.

En el explorador de paquetes de eclipse deberá tener lo siguiente, figura E23:

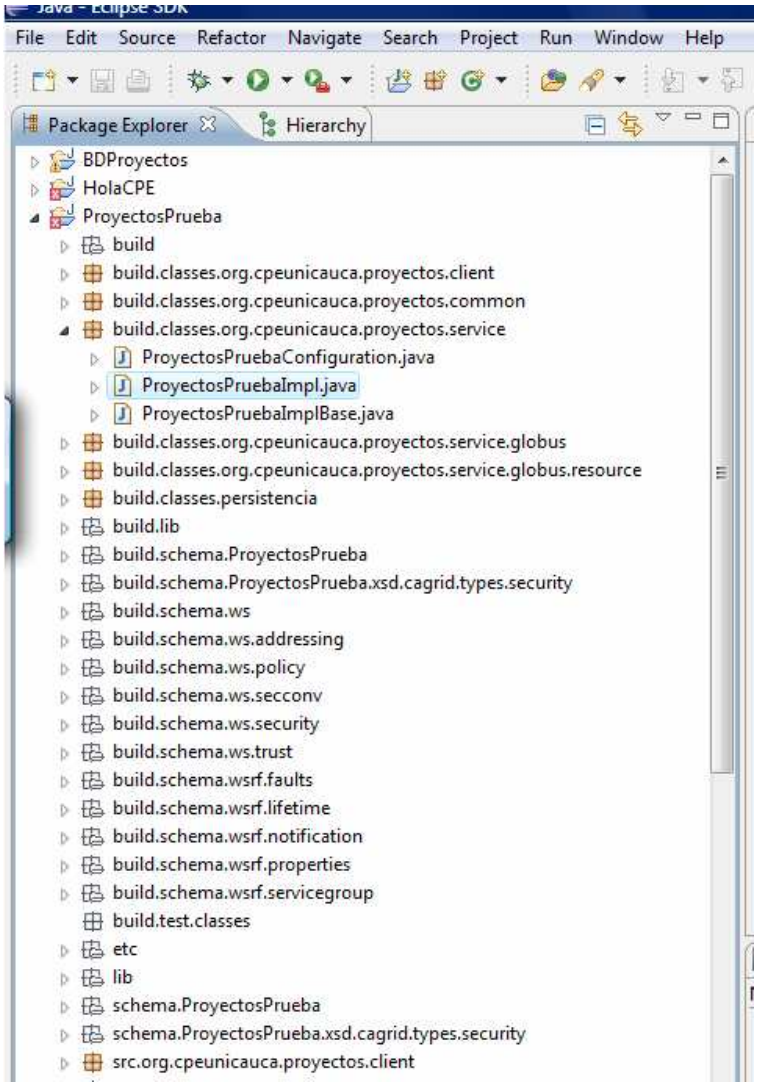

Figura E23: Explorador de Paquetes.

En este proyecto expanda el paquete build.classes.org.cpeunicauca.proyectos.service y abra la clase ProyectosPruebaImpl.java. Esta clase posee todos los métodos que se deben implementar para que el servicio web funcione correctamente. Esto servicios requieren acceder a una base de datos para esto se deben crear unas clases de acceso a datos. Dentro de un paquete que se llamará persistencia deben estar las siguientes clases de acceso a datos, figura E24.

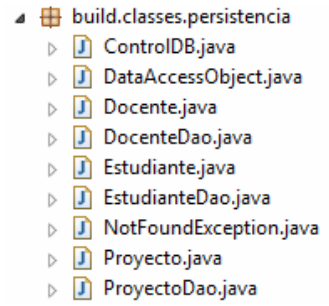

Figura E24: persistencia.

Esas clases de acceso a datos son las siguientes y no se explican pues no son objeto de este tutorial.

La primera clase ControlDB es una clase que permite controlar los métodos de acceso a datos ofrecidas por cada uno de los conocidos como valueObjects.

```
package persistencia;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.Iterator;

public class ControlDB{

	private Connection conn;
	private DataAccessObject dao;
	private DocenteDao ddao;
	private EstudianteDao edao;
	private ProyectoDao pdao;

	public ControlDB() {
		conn = null;
		ddao = new DocenteDao();
		edao = new EstudianteDao();
		pdao = new ProyectoDao();
	}

	public int conectarBD(){
		try{
			//registrar la clase del driver
			Class.forName("org.postgresql.Driver");
			//obtener el objeto de conexion

	//conn=DriverManager.getConnection("jdbc:postgresql://localhost/proyectosi","postgres","postgres");

	conn=DriverManager.getConnection("jdbc:postgresql://192.168.1.14:5432/proyectosi","postgres","postgres");

			return 0;
		}
		catch(ClassNotFoundException e){
			e.printStackTrace();
			return 1;
```

```
		}
		catch(SQLException e){
			e.printStackTrace();
			return 2;
		}
	}

	public Connection getConnection(){
		return conn;
	}

	public void guardar(Object vo) {
		try {
			if(vo.getClass() == Proyecto.class){
				pdao.create(conn, (Proyecto)vo);
			}
			else if (vo.getClass() == Estudiante.class){
				edao.create(conn, (Estudiante)vo);
			}
			else if (vo.getClass() == Docente.class){
				ddao.create(conn, (Docente)vo);
			}
		} catch (SQLException e) {
			e.printStackTrace();
		}

	}

	public String getInfoProyecto(int idProyecto){
		Proyecto proy = new Proyecto();
		String respuesta = "";
		proy.setId(idProyecto);
		try {
			pdao.load(conn, proy);
		} catch (NotFoundException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
			return e.getMessage();
		} catch (SQLException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
			return e.getMessage();
		}

		respuesta = "Titulo "+proy.getTitulo()+" ";
		respuesta += "Estado "+proy.getEstado()+" ";
		respuesta += "Objetivos "+proy.getObjetivos()+" ";
		return respuesta;
	}

	public String[] getEstudiantes(){
		String[] estudiantes;
		ArrayList lista = null;
		try {
			lista = (ArrayList)edao.loadAll(conn);
			estudiantes = new String[lista.size()];
			int i = 0;
			for (Iterator iter = lista.iterator(); iter.hasNext();) {

				Estudiante es = (Estudiante) iter.next();
				String estr = "/-/"+es.getCodigo()+"/-/"+
				es.getNombres()+"/-/"+
```

```
                        es.getGrado()+"/-/";
                        estudiantes[i] = estr;
                        i++;
                }
        } catch (SQLException e) {
                e.printStackTrace();
                estudiantes = new String[1];
                estudiantes[0] = e.getMessage();
        }
        return estudiantes;
}

public String[] getProyectos() {
        String[] proyectos;
                ArrayList lista = null;
        try {
                lista = (ArrayList)pdao.loadAll(conn);
                proyectos = new String[lista.size()];
                int i = 0;
                for (Iterator iter = lista.iterator(); iter.hasNext();) {

                        Proyecto p = (Proyecto) iter.next();
                        String prstr = "/-/"+p.getId()+"/-/"+
                        p.getTitulo()+"/-/"+
                        p.getEstado()+"/-/"+
                        p.getProblema()+"/-/"+
                        p.getObjetivos()+"/-/"+
                        p.getMetodologia()+"/-/"+
                        p.getPoblacion()+"/-/"+
                        p.getResultados()+"/-/"+
                        p.getObservaciones()+"/-/";
                        proyectos[i] = prstr;
                        i++;
                }
        } catch (SQLException e) {
                e.printStackTrace();
                proyectos = new String[1];
                proyectos[0] = e.getMessage();
        }
        return proyectos;
}

public String[] getProyectos(String key, int parametro) {
        String[] proyectos;
        Proyecto pIn = new Proyecto();
        switch (parametro) {
        case 0://por titulo
                pIn.setTitulo(key);
                break;
        case 1://por estado
                pIn.setEstado(key);
                break;
        case 2://por problema
                pIn.setProblema(key);
                break;
        case 3://por objetivos
                pIn.setObjetivos(key);
                break;
        case 4://por metodlogia
                pIn.setMetodologia(key);
                break;
        case 5://por poblacion
                pIn.setPoblacion(key);
```

```
			break;
		case 6://por resultados
			pIn.setResultados(key);
			break;
		default:
			pIn.setTitulo(key);
			break;
		}
		ArrayList lista = null;
		try {
			lista = (ArrayList)pdao.searchMatching(conn, pIn);
			proyectos = new String[lista.size()];
			int i = 0;
			for (Iterator iter = lista.iterator(); iter.hasNext();) {

				Proyecto p = (Proyecto) iter.next();
				String prstr = "/-/"+p.getId()+"/-/"+
				p.getTitulo()+"/-/"+
				p.getEstado()+"/-/"+
				p.getProblema()+"/-/"+
				p.getObjetivos()+"/-/"+
				p.getMetodologia()+"/-/"+
				p.getPoblacion()+"/-/"+
				p.getResultados()+"/-/"+
				p.getObservaciones()+"/-/";
				proyectos[i] = prstr;
				i++;
			}
		} catch (SQLException e) {
			e.printStackTrace();
			proyectos = new String[1];
			proyectos[0] = e.getMessage();
		}
		return proyectos;
	}


	public String getDocente(int cedula){
		Docente doc = new Docente(cedula);
		String respuesta = "FALLA AL BUSCAR DOCENTE";
		try {
			ddao.load(conn, doc);
			respuesta = "/-/"+doc.getCedula()+"/-/"+
			doc.getNombres()+"/-/"+
			doc.getArea()+"/-/"+
			doc.getId()+"/-/";
		} catch (NotFoundException e) {
			e.printStackTrace();
			respuesta = "NO SE ENCUENTRA";
			return e.getMessage();
		} catch (SQLException e) {
			e.printStackTrace();
			respuesta = "EXCEPCION SQL";
			return e.getMessage();
		}
		return respuesta;
	}

	public String getEstudiante(int codigo){
		Estudiante est = new Estudiante(codigo);
		String respuesta = "FALLA AL BUSCAR ESTUDIANTE";
		try {
			edao.load(conn, est);
```

```
                respuesta = "/-/"+est.getCodigo()+"/-/"+
                est.getNombres()+"/-/"+
                est.getGrado()+"/-/"+
                est.getId()+"/-/";
        } catch (NotFoundException e) {
                e.printStackTrace();
                respuesta = "NO SE ENCUENTRA";
                return e.getMessage();
        } catch (SQLException e) {
                e.printStackTrace();
                respuesta = "EXCEPCION SQL";
                return e.getMessage();
        }
        return respuesta;
}

public String getProyecto(int id){
        Proyecto p = new Proyecto(id);
        String respuesta = "FALLA AL BUSCAR PROYECTO";
        try {
                pdao.load(conn, p);
                respuesta = "/-/"+p.getId()+"/-/"+
                p.getTitulo()+"/-/"+
                p.getEstado()+"/-/"+
                p.getProblema()+"/-/"+
                p.getObjetivos()+"/-/"+
                p.getMetodologia()+"/-/"+
                p.getPoblacion()+"/-/"+
                p.getResultados()+"/-/"+
                p.getObservaciones()+"/-/";
        } catch (NotFoundException e) {
                e.printStackTrace();
                respuesta = "NO SE ENCUENTRA";
                return e.getMessage();
        } catch (SQLException e) {
                e.printStackTrace();
                respuesta = "EXCEPCION SQL";
                return e.getMessage();
        }
        return respuesta;
}

public boolean guardarDocente(int cedula, String nombres, String area, int id){
        Docente docente = new Docente();
        docente.setCedula(cedula);
        docente.setNombres(nombres);
        docente.setArea(area);
        docente.setId(id);
        return guardarDocente(docente);
}

public boolean guardarEstudiante(int codigo, String nombres, int grado, int id){
        Estudiante e = new Estudiante();
        e.setCodigo(codigo);
        e.setNombres(nombres);
        e.setGrado(grado);
        e.setId(id);
        return guardarEstudiante(e);
}

public boolean guardarProyecto(
                int id,
                String titulo,
```

```
                String estado,
                String problema,
                String objetivos,
                String metodologia,
                String poblacion,
                String resultados,
                String observaciones){
        Proyecto p = new Proyecto();
        p.setId(id);
     p.setTitulo(titulo);
     p.setEstado(estado);
     p.setProblema(problema);
     p.setObjetivos(objetivos);
     p.setMetodologia(metodologia);
     p.setPoblacion(poblacion);
     p.setResultados(resultados);
     p.setObservaciones(observaciones);
        return guardarProyecto(p);
    }

    public boolean guardarDocente(Docente docente){
        boolean creado = false;
        try {
            ddao.create(conn,docente);
            creado = true;
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return creado;
    }

    public boolean guardarProyecto(Proyecto proyecto){
        boolean creado = false;
        try {
            pdao.create(conn,proyecto);
            creado = true;
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return creado;
    }

    public boolean guardarEstudiante(Estudiante estudiante){
        boolean creado = false;
        try {
            edao.create(conn,estudiante);
            creado = true;
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return creado;
    }
}
```

Nota: las siguientes clases fueron creadas utilizando [2].

Las siguientes clases son los value Objects y los objetos de acceso a datos estos son:

Value Objects:

Docente:

```
package persistencia;


import java.io.*;
import java.sql.*;
import java.util.*;
import java.math.*;

public class Docente implements Cloneable, Serializable {

    /**
     * Persistent Instance variables. This data is directly
     * mapped to the columns of database table.
     */
    private int cedula;
    private String nombres;
    private String area;
    private int id;



    /**
     * Constructors. DaoGen generates two constructors by default.
     * The first one takes no arguments and provides the most simple
     * way to create object instance. The another one takes one
     * argument, which is the primary key of the corresponding table.
     */

    public Docente () {

    }

    public Docente (int cedulaIn) {

          this.cedula = cedulaIn;

    }


    /**
     * Get- and Set-methods for persistent variables. The default
     * behaviour does not make any checks against malformed data,
     * so these might require some manual additions.
     */

    public int getCedula() {
          return this.cedula;
    }
    public void setCedula(int cedulaIn) {
          this.cedula = cedulaIn;
    }

    public String getNombres() {
          return this.nombres;
    }
    public void setNombres(String nombresIn) {
          this.nombres = nombresIn;
    }
```

```
public String getArea() {
	return this.area;
}
public void setArea(String areaIn) {
	this.area = areaIn;
}

public int getId() {
	return this.id;
}
public void setId(int idIn) {
	this.id = idIn;
}



/**
 * setAll allows to set all persistent variables in one method call.
 * This is useful, when all data is available and it is needed to
 * set the initial state of this object. Note that this method will
 * directly modify instance variales, without going trough the
 * individual set-methods.
 */

public void setAll(int cedulaIn,
	String nombresIn,
	String areaIn,
	int idIn) {
	this.cedula = cedulaIn;
	this.nombres = nombresIn;
	this.area = areaIn;
	this.id = idIn;
}


/**
 * hasEqualMapping-method will compare two Docente instances
 * and return true if they contain same values in all persistent instance
 * variables. If hasEqualMapping returns true, it does not mean the objects
 * are the same instance. However it does mean that in that moment, they
 * are mapped to the same row in database.
 */
public boolean hasEqualMapping(Docente valueObject) {

	if (valueObject.getCedula() != this.cedula) {
		return(false);
	}
	if (this.nombres == null) {
		if (valueObject.getNombres() != null)
			return(false);
	} else if (!this.nombres.equals(valueObject.getNombres())) {
		return(false);
	}
	if (this.area == null) {
		if (valueObject.getArea() != null)
			return(false);
	} else if (!this.area.equals(valueObject.getArea())) {
		return(false);
	}
	if (valueObject.getId() != this.id) {
		return(false);
	}
```

```
            return true;
        }


        /**
         * toString will return String object representing the state of this
         * valueObject. This is useful during application development, and
         * possibly when application is writing object states in textlog.
         */
        public String toString() {
            StringBuffer out = new StringBuffer(this.getDaogenVersion());
            out.append("\nclass Docente, mapping to table docente\n");
            out.append("Persistent attributes: \n");
            out.append("cedula = " + this.cedula + "\n");
            out.append("nombres = " + this.nombres + "\n");
            out.append("area = " + this.area + "\n");
            out.append("id = " + this.id + "\n");
            return out.toString();
        }


        /**
         * Clone will return identical deep copy of this valueObject.
         * Note, that this method is different than the clone() which
         * is defined in java.lang.Object. Here, the retuned cloned object
         * will also have all its attributes cloned.
         */
        public Object clone() {
            Docente cloned = new Docente();

            cloned.setCedula(this.cedula);
            if (this.nombres != null)
                 cloned.setNombres(new String(this.nombres));
            if (this.area != null)
                 cloned.setArea(new String(this.area));
            cloned.setId(this.id);
            return cloned;
        }



        /**
         * getDaogenVersion will return information about
         * generator which created these sources.
         */
        public String getDaogenVersion() {
            return "DaoGen version 2.4.1";
        }

}
```

Estudiante:

```
package persistencia;


import java.io.*;
import java.sql.*;
import java.util.*;
```

```
import java.math.*;


public class Estudiante implements Cloneable, Serializable {

    /**
     * Persistent Instance variables. This data is directly
     * mapped to the columns of database table.
     */
    private int codigo;
    private String nombres;
    private int grado;
    private int id;



    /**
     * Constructors. DaoGen generates two constructors by default.
     * The first one takes no arguments and provides the most simple
     * way to create object instance. The another one takes one
     * argument, which is the primary key of the corresponding table.
     */

    public Estudiante () {

    }

    public Estudiante (int codigoIn) {

          this.codigo = codigoIn;

    }


    /**
     * Get- and Set-methods for persistent variables. The default
     * behaviour does not make any checks against malformed data,
     * so these might require some manual additions.
     */

    public int getCodigo() {
          return this.codigo;
    }
    public void setCodigo(int codigoIn) {
          this.codigo = codigoIn;
    }

    public String getNombres() {
          return this.nombres;
    }
    public void setNombres(String nombresIn) {
          this.nombres = nombresIn;
    }

    public int getGrado() {
          return this.grado;
    }
    public void setGrado(int gradoIn) {
          this.grado = gradoIn;
    }

    public int getId() {
          return this.id;
```

```
}
public void setId(int idIn) {
	this.id = idIn;
}



/**
 * setAll allows to set all persistent variables in one method call.
 * This is useful, when all data is available and it is needed to
 * set the initial state of this object. Note that this method will
 * directly modify instance variales, without going trough the
 * individual set-methods.
 */

public void setAll(int codigoIn,
	String nombresIn,
	int gradoIn,
	int idIn) {
	this.codigo = codigoIn;
	this.nombres = nombresIn;
	this.grado = gradoIn;
	this.id = idIn;
}


/**
 * hasEqualMapping-method will compare two Estudiante instances
 * and return true if they contain same values in all persistent instance
 * variables. If hasEqualMapping returns true, it does not mean the objects
 * are the same instance. However it does mean that in that moment, they
 * are mapped to the same row in database.
 */
public boolean hasEqualMapping(Estudiante valueObject) {

	if (valueObject.getCodigo() != this.codigo) {
		return(false);
	}
	if (this.nombres == null) {
		if (valueObject.getNombres() != null)
			return(false);
	} else if (!this.nombres.equals(valueObject.getNombres())) {
		return(false);
	}
	if (valueObject.getGrado() != this.grado) {
		return(false);
	}
	if (valueObject.getId() != this.id) {
		return(false);
	}

	return true;
}



/**
 * toString will return String object representing the state of this
 * valueObject. This is useful during application development, and
 * possibly when application is writing object states in textlog.
 */
public String toString() {
    StringBuffer out = new StringBuffer(this.getDaogenVersion());
```

```
        out.append("\nclass Estudiante, mapping to table estudiante\n");
        out.append("Persistent attributes: \n");
        out.append("codigo = " + this.codigo + "\n");
        out.append("nombres = " + this.nombres + "\n");
        out.append("grado = " + this.grado + "\n");
        out.append("id = " + this.id + "\n");
        return out.toString();
    }


    /**
     * Clone will return identical deep copy of this valueObject.
     * Note, that this method is different than the clone() which
     * is defined in java.lang.Object. Here, the retuned cloned object
     * will also have all its attributes cloned.
     */
    public Object clone() {
        Estudiante cloned = new Estudiante();

        cloned.setCodigo(this.codigo);
        if (this.nombres != null)
             cloned.setNombres(new String(this.nombres));
        cloned.setGrado(this.grado);
        cloned.setId(this.id);
        return cloned;
    }



    /**
     * getDaogenVersion will return information about
     * generator which created these sources.
     */
    public String getDaogenVersion() {
        return "DaoGen version 2.4.1";
    }

}
```

Proyecto:

```
package persistencia;


import java.io.*;
import java.sql.*;
import java.util.*;
import java.math.*;


public class Proyecto implements Cloneable, Serializable {

    /**
     * Persistent Instance variables. This data is directly
     * mapped to the columns of database table.
     */
    private int id;
    private String titulo;
    private String estado;
    private String problema;
    private String objetivos;
    private String metodologia;
```

```
private String poblacion;
private String resultados;
private String observaciones;



/**
 * Constructors. DaoGen generates two constructors by default.
 * The first one takes no arguments and provides the most simple
 * way to create object instance. The another one takes one
 * argument, which is the primary key of the corresponding table.
 */

public Proyecto () {

}

public Proyecto (int idIn) {

      this.id = idIn;

}


/**
 * Get- and Set-methods for persistent variables. The default
 * behaviour does not make any checks against malformed data,
 * so these might require some manual additions.
 */

public int getId() {
      return this.id;
}
public void setId(int idIn) {
      this.id = idIn;
}

public String getTitulo() {
      return this.titulo;
}
public void setTitulo(String tituloIn) {
      this.titulo = tituloIn;
}

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

public String getProblema() {
      return this.problema;
}
public void setProblema(String problemaIn) {
      this.problema = problemaIn;
}

public String getObjetivos() {
      return this.objetivos;
}
public void setObjetivos(String objetivosIn) {
      this.objetivos = objetivosIn;
```

```
}

public String getMetodologia() {
	return this.metodologia;
}
public void setMetodologia(String metodologiaIn) {
	this.metodologia = metodologiaIn;
}

public String getPoblacion() {
	return this.poblacion;
}
public void setPoblacion(String poblacionIn) {
	this.poblacion = poblacionIn;
}

public String getResultados() {
	return this.resultados;
}
public void setResultados(String resultadosIn) {
	this.resultados = resultadosIn;
}

public String getObservaciones() {
	return this.observaciones;
}
public void setObservaciones(String observacionesIn) {
	this.observaciones = observacionesIn;
}



/**
 * setAll allows to set all persistent variables in one method call.
 * This is useful, when all data is available and it is needed to
 * set the initial state of this object. Note that this method will
 * directly modify instance variales, without going trough the
 * individual set-methods.
 */

public void setAll(int idIn,
	String tituloIn,
	String estadoIn,
	String problemaIn,
	String objetivosIn,
	String metodologiaIn,
	String poblacionIn,
	String resultadosIn,
	String observacionesIn) {
	this.id = idIn;
	this.titulo = tituloIn;
	this.estado = estadoIn;
	this.problema = problemaIn;
	this.objetivos = objetivosIn;
	this.metodologia = metodologiaIn;
	this.poblacion = poblacionIn;
	this.resultados = resultadosIn;
	this.observaciones = observacionesIn;
}


/**
 * hasEqualMapping-method will compare two Proyecto instances
```

```
 * and return true if they contain same values in all persistent instance
 * variables. If hasEqualMapping returns true, it does not mean the objects
 * are the same instance. However it does mean that in that moment, they
 * are mapped to the same row in database.
 */
public boolean hasEqualMapping(Proyecto valueObject) {

	if (valueObject.getId() != this.id) {
		return(false);
	}
	if (this.titulo == null) {
		if (valueObject.getTitulo() != null)
			return(false);
	} else if (!this.titulo.equals(valueObject.getTitulo())) {
		return(false);
	}
	if (this.estado == null) {
		if (valueObject.getEstado() != null)
			return(false);
	} else if (!this.estado.equals(valueObject.getEstado())) {
		return(false);
	}
	if (this.problema == null) {
		if (valueObject.getProblema() != null)
			return(false);
	} else if (!this.problema.equals(valueObject.getProblema())) {
		return(false);
	}
	if (this.objetivos == null) {
		if (valueObject.getObjetivos() != null)
			return(false);
	} else if (!this.objetivos.equals(valueObject.getObjetivos())) {
		return(false);
	}
	if (this.metodologia == null) {
		if (valueObject.getMetodologia() != null)
			return(false);
	} else if (!this.metodologia.equals(valueObject.getMetodologia())) {
		return(false);
	}
	if (this.poblacion == null) {
		if (valueObject.getPoblacion() != null)
			return(false);
	} else if (!this.poblacion.equals(valueObject.getPoblacion())) {
		return(false);
	}
	if (this.resultados == null) {
		if (valueObject.getResultados() != null)
			return(false);
	} else if (!this.resultados.equals(valueObject.getResultados())) {
		return(false);
	}
	if (this.observaciones == null) {
		if (valueObject.getObservaciones() != null)
			return(false);
	} else if (!this.observaciones.equals(valueObject.getObservaciones())) {
		return(false);
	}

	return true;
}
```

```
    /**
     * toString will return String object representing the state of this
     * valueObject. This is useful during application development, and
     * possibly when application is writing object states in textlog.
     */
    public String toString() {
        StringBuffer out = new StringBuffer(this.getDaogenVersion());
        out.append("\nclass Proyecto, mapping to table proyecto\n");
        out.append("Persistent attributes: \n");
        out.append("id = " + this.id + "\n");
        out.append("titulo = " + this.titulo + "\n");
        out.append("estado = " + this.estado + "\n");
        out.append("problema = " + this.problema + "\n");
        out.append("objetivos = " + this.objetivos + "\n");
        out.append("metodologia = " + this.metodologia + "\n");
        out.append("poblacion = " + this.poblacion + "\n");
        out.append("resultados = " + this.resultados + "\n");
        out.append("observaciones = " + this.observaciones + "\n");
        return out.toString();
    }


    /**
     * Clone will return identical deep copy of this valueObject.
     * Note, that this method is different than the clone() which
     * is defined in java.lang.Object. Here, the retuned cloned object
     * will also have all its attributes cloned.
     */
    public Object clone() {
        Proyecto cloned = new Proyecto();

        cloned.setId(this.id);
        if (this.titulo != null)
             cloned.setTitulo(new String(this.titulo));
        if (this.estado != null)
             cloned.setEstado(new String(this.estado));
        if (this.problema != null)
             cloned.setProblema(new String(this.problema));
        if (this.objetivos != null)
             cloned.setObjetivos(new String(this.objetivos));
        if (this.metodologia != null)
             cloned.setMetodologia(new String(this.metodologia));
        if (this.poblacion != null)
             cloned.setPoblacion(new String(this.poblacion));
        if (this.resultados != null)
             cloned.setResultados(new String(this.resultados));
        if (this.observaciones != null)
             cloned.setObservaciones(new String(this.observaciones));
        return cloned;
    }



    /**
     * getDaogenVersion will return information about
     * generator which created these sources.
     */
    public String getDaogenVersion() {
        return "DaoGen version 2.4.1";
    }

}
```

Objetos de Acceso a Datos.

DocenteDAO:

```
package persistencia;


import java.sql.*;
import java.util.*;
import java.math.*;


public class DocenteDao {



    /**
     * createValueObject-method. This method is used when the Dao class needs
     * to create new value object instance. The reason why this method exists
     * is that sometimes the programmer may want to extend also the valueObject
     * and then this method can be overrided to return extended valueObject.
     * NOTE: If you extend the valueObject class, make sure to override the
     * clone() method in it!
     */
    public Docente createValueObject() {
          return new Docente();
    }


    /**
     * getObject-method. This will create and load valueObject contents from database
     * using given Primary-Key as identifier. This method is just a convenience method
     * for the real load-method which accepts the valueObject as a parameter. Returned
     * valueObject will be created using the createValueObject() method.
     */
    public Docente getObject(Connection conn, int cedula) throws NotFoundException,
SQLException {

          Docente valueObject = createValueObject();
          valueObject.setCedula(cedula);
          load(conn, valueObject);
          return valueObject;
    }


    /**
     * load-method. This will load valueObject contents from database using
     * Primary-Key as identifier. Upper layer should use this so that valueObject
     * instance is created and only primary-key should be specified. Then call
     * this method to complete other persistent information. This method will
     * overwrite all other fields except primary-key and possible runtime variables.
     * If load can not find matching row, NotFoundException will be thrown.
     *
     * @param conn         This method requires working database connection.
     * @param valueObject  This parameter contains the class instance to be loaded.
     *                     Primary-key field must be set for this to work properly.
     */
    public void load(Connection conn, Docente valueObject) throws NotFoundException,
SQLException {

          String sql = "SELECT * FROM docente WHERE (cedula = ? ) ";
```

```
        PreparedStatement stmt = null;

        try {
             stmt = conn.prepareStatement(sql);
             stmt.setInt(1, valueObject.getCedula());

             singleQuery(conn, stmt, valueObject);

        } finally {
            if (stmt != null)
                stmt.close();
        }
    }


    /**
     * LoadAll-method. This will read all contents from database table and
     * build a List containing valueObjects. Please note, that this method
     * will consume huge amounts of resources if table has lot's of rows.
     * This should only be used when target tables have only small amounts
     * of data.
     *
     * @param conn         This method requires working database connection.
     */
    public List loadAll(Connection conn) throws SQLException {

        String sql = "SELECT * FROM docente ORDER BY cedula ASC ";
        List searchResults = listQuery(conn, conn.prepareStatement(sql));

        return searchResults;
    }



    /**
     * create-method. This will create new row in database according to supplied
     * valueObject contents. Make sure that values for all NOT NULL columns are
     * correctly specified. Also, if this table does not use automatic surrogate-keys
     * the primary-key must be specified. After INSERT command this method will
     * read the generated primary-key back to valueObject if automatic surrogate-keys
     * were used.
     *
     * @param conn         This method requires working database connection.
     * @param valueObject  This parameter contains the class instance to be created.
     *                     If automatic surrogate-keys are not used the Primary-key
     *                     field must be set for this to work properly.
     */
    public synchronized void create(Connection conn, Docente valueObject) throws
SQLException {

        String sql = "";
        PreparedStatement stmt = null;
        ResultSet result = null;

        try {
             sql = "INSERT INTO docente ( cedula, nombres, area, "
             + "id) VALUES (?, ?, ?, ?) ";
             stmt = conn.prepareStatement(sql);

             stmt.setInt(1, valueObject.getCedula());
             stmt.setString(2, valueObject.getNombres());
             stmt.setString(3, valueObject.getArea());
             stmt.setInt(4, valueObject.getId());
```

```
                int rowcount = databaseUpdate(conn, stmt);
                if (rowcount != 1) {
                     //System.out.println("PrimaryKey Error when updating DB!");
                     throw new SQLException("PrimaryKey Error when updating DB!");
                }

            } finally {
                if (stmt != null)
                    stmt.close();
            }


    }


    /**
     * save-method. This method will save the current state of valueObject to database.
     * Save can not be used to create new instances in database, so upper layer must
     * make sure that the primary-key is correctly specified. Primary-key will indicate
     * which instance is going to be updated in database. If save can not find matching
     * row, NotFoundException will be thrown.
     *
     * @param conn         This method requires working database connection.
     * @param valueObject  This parameter contains the class instance to be saved.
     *                     Primary-key field must be set for this to work properly.
     */
    public void save(Connection conn, Docente valueObject)
            throws NotFoundException, SQLException {

            String sql = "UPDATE docente SET nombres = ?, area = ?, id = ? WHERE (cedula
= ? ) ";
            PreparedStatement stmt = null;

            try {
                stmt = conn.prepareStatement(sql);
                stmt.setString(1, valueObject.getNombres());
                stmt.setString(2, valueObject.getArea());
                stmt.setInt(3, valueObject.getId());

                stmt.setInt(4, valueObject.getCedula());

                int rowcount = databaseUpdate(conn, stmt);
                if (rowcount == 0) {
                     //System.out.println("Object could not be saved! (PrimaryKey not
found)");
                     throw new NotFoundException("Object could not be saved! (PrimaryKey
not found)");
                }
                if (rowcount > 1) {
                     //System.out.println("PrimaryKey Error when updating DB! (Many
objects were affected!)");
                     throw new SQLException("PrimaryKey Error when updating DB! (Many
objects were affected!)");
                }
            } finally {
                if (stmt != null)
                    stmt.close();
            }
    }


    /**
```

```
     * delete-method. This method will remove the information from database as
identified by
     * by primary-key in supplied valueObject. Once valueObject has been deleted it can
not
     * be restored by calling save. Restoring can only be done using create method but
if
     * database is using automatic surrogate-keys, the resulting object will have
different
     * primary-key than what it was in the deleted object. If delete can not find
matching row,
     * NotFoundException will be thrown.
     *
     * @param conn         This method requires working database connection.
     * @param valueObject  This parameter contains the class instance to be deleted.
     *                     Primary-key field must be set for this to work properly.
     */
    public void delete(Connection conn, Docente valueObject)
          throws NotFoundException, SQLException {

          String sql = "DELETE FROM docente WHERE (cedula = ? ) ";
          PreparedStatement stmt = null;

          try {
              stmt = conn.prepareStatement(sql);
              stmt.setInt(1, valueObject.getCedula());

              int rowcount = databaseUpdate(conn, stmt);
              if (rowcount == 0) {
                   //System.out.println("Object could not be deleted (PrimaryKey not
found)");
                   throw new NotFoundException("Object could not be deleted!
(PrimaryKey not found)");
              }
              if (rowcount > 1) {
                   //System.out.println("PrimaryKey Error when updating DB! (Many
objects were deleted!)");
                   throw new SQLException("PrimaryKey Error when updating DB! (Many
objects were deleted!)");
              }
          } finally {
              if (stmt != null)
                  stmt.close();
          }
    }


    /**
     * deleteAll-method. This method will remove all information from the table that
matches
     * this Dao and ValueObject couple. This should be the most efficient way to clear
table.
     * Once deleteAll has been called, no valueObject that has been created before can
be
     * restored by calling save. Restoring can only be done using create method but if
database
     * is using automatic surrogate-keys, the resulting object will have different
primary-key
     * than what it was in the deleted object. (Note, the implementation of this method
should
     * be different with different DB backends.)
     *
     * @param conn         This method requires working database connection.
     */
```

```
    public void deleteAll(Connection conn) throws SQLException {

          String sql = "DELETE FROM docente";
          PreparedStatement stmt = null;

          try {
              stmt = conn.prepareStatement(sql);
              int rowcount = databaseUpdate(conn, stmt);
          } finally {
              if (stmt != null)
                  stmt.close();
          }
    }


    /**
     * coutAll-method. This method will return the number of all rows from table that
matches
     * this Dao. The implementation will simply execute "select count(primarykey) from
table".
     * If table is empty, the return value is 0. This method should be used before
calling
     * loadAll, to make sure table has not too many rows.
     *
     * @param conn         This method requires working database connection.
     */
    public int countAll(Connection conn) throws SQLException {

          String sql = "SELECT count(*) FROM docente";
          PreparedStatement stmt = null;
          ResultSet result = null;
          int allRows = 0;

          try {
              stmt = conn.prepareStatement(sql);
              result = stmt.executeQuery();

              if (result.next())
                  allRows = result.getInt(1);
          } finally {
              if (result != null)
                  result.close();
              if (stmt != null)
                  stmt.close();
          }
          return allRows;
    }


    /**
     * searchMatching-Method. This method provides searching capability to
     * get matching valueObjects from database. It works by searching all
     * objects that match permanent instance variables of given object.
     * Upper layer should use this by setting some parameters in valueObject
     * and then  call searchMatching. The result will be 0-N objects in a List,
     * all matching those criteria you specified. Those instance-variables that
     * have NULL values are excluded in search-criteria.
     *
     * @param conn         This method requires working database connection.
     * @param valueObject  This parameter contains the class instance where search will
be based.
     *                     Primary-key field should not be set.
     */
```

```
    public List searchMatching(Connection conn, Docente valueObject) throws
SQLException {

          List searchResults;

          boolean first = true;
          StringBuffer sql = new StringBuffer("SELECT * FROM docente WHERE 1=1 ");

          if (valueObject.getCedula() != 0) {
              if (first) { first = false; }
              sql.append("AND cedula = ").append(valueObject.getCedula()).append(" ");
          }

          if (valueObject.getNombres() != null) {
              if (first) { first = false; }
              sql.append("AND nombres LIKE
'").append(valueObject.getNombres()).append("%' ");
          }

          if (valueObject.getArea() != null) {
              if (first) { first = false; }
              sql.append("AND area LIKE '").append(valueObject.getArea()).append("%'
");
          }

          if (valueObject.getId() != 0) {
              if (first) { first = false; }
              sql.append("AND id = ").append(valueObject.getId()).append(" ");
          }


          sql.append("ORDER BY cedula ASC ");

          // Prevent accidential full table results.
          // Use loadAll if all rows must be returned.
          if (first)
               searchResults = new ArrayList();
          else
               searchResults = listQuery(conn, conn.prepareStatement(sql.toString()));

          return searchResults;
    }


    /**
     * getDaogenVersion will return information about
     * generator which created these sources.
     */
    public String getDaogenVersion() {
        return "DaoGen version 2.4.1";
    }


    /**
     * databaseUpdate-method. This method is a helper method for internal use. It will
execute
     * all database handling that will change the information in tables. SELECT queries
will
     * not be executed here however. The return value indicates how many rows were
affected.
     * This method will also make sure that if cache is used, it will reset when data
changes.
     *
```

```
     * @param conn         This method requires working database connection.
     * @param stmt         This parameter contains the SQL statement to be excuted.
     */
    protected int databaseUpdate(Connection conn, PreparedStatement stmt) throws
SQLException {

          int result = stmt.executeUpdate();

          return result;
    }



    /**
     * databaseQuery-method. This method is a helper method for internal use. It will
execute
     * all database queries that will return only one row. The resultset will be
converted
     * to valueObject. If no rows were found, NotFoundException will be thrown.
     *
     * @param conn         This method requires working database connection.
     * @param stmt         This parameter contains the SQL statement to be excuted.
     * @param valueObject  Class-instance where resulting data will be stored.
     */
    protected void singleQuery(Connection conn, PreparedStatement stmt, Docente
valueObject)
          throws NotFoundException, SQLException {

          ResultSet result = null;

          try {
              result = stmt.executeQuery();

              if (result.next()) {

                   valueObject.setCedula(result.getInt("cedula"));
                   valueObject.setNombres(result.getString("nombres"));
                   valueObject.setArea(result.getString("area"));
                   valueObject.setId(result.getInt("id"));

              } else {
                    //System.out.println("Docente Object Not Found!");
                    throw new NotFoundException("Docente Object Not Found!");
              }
          } finally {
              if (result != null)
                  result.close();
              if (stmt != null)
                  stmt.close();
          }
    }


    /**
     * databaseQuery-method. This method is a helper method for internal use. It will
execute
     * all database queries that will return multiple rows. The resultset will be
converted
     * to the List of valueObjects. If no rows were found, an empty List will be
returned.
     *
     * @param conn         This method requires working database connection.
     * @param stmt         This parameter contains the SQL statement to be excuted.
```

```
     */
    protected List listQuery(Connection conn, PreparedStatement stmt) throws
SQLException {

          ArrayList searchResults = new ArrayList();
          ResultSet result = null;

          try {
              result = stmt.executeQuery();

              while (result.next()) {
                   Docente temp = createValueObject();

                   temp.setCedula(result.getInt("cedula"));
                   temp.setNombres(result.getString("nombres"));
                   temp.setArea(result.getString("area"));
                   temp.setId(result.getInt("id"));

                   searchResults.add(temp);
              }

          } finally {
              if (result != null)
                  result.close();
              if (stmt != null)
                  stmt.close();
          }

          return (List)searchResults;
    }


}
```

EstudianteDAO:

```
package persistencia;


import java.sql.*;
import java.util.*;
import java.math.*;


public class EstudianteDao {



    /**
     * createValueObject-method. This method is used when the Dao class needs
     * to create new value object instance. The reason why this method exists
     * is that sometimes the programmer may want to extend also the valueObject
     * and then this method can be overrided to return extended valueObject.
     * NOTE: If you extend the valueObject class, make sure to override the
     * clone() method in it!
     */
    public Estudiante createValueObject() {
          return new Estudiante();
    }


    /**
```

```
     * getObject-method. This will create and load valueObject contents from database
     * using given Primary-Key as identifier. This method is just a convenience method
     * for the real load-method which accepts the valueObject as a parameter. Returned
     * valueObject will be created using the createValueObject() method.
     */
    public Estudiante getObject(Connection conn, int codigo) throws NotFoundException,
SQLException {

          Estudiante valueObject = createValueObject();
          valueObject.setCodigo(codigo);
          load(conn, valueObject);
          return valueObject;
    }


    /**
     * load-method. This will load valueObject contents from database using
     * Primary-Key as identifier. Upper layer should use this so that valueObject
     * instance is created and only primary-key should be specified. Then call
     * this method to complete other persistent information. This method will
     * overwrite all other fields except primary-key and possible runtime variables.
     * If load can not find matching row, NotFoundException will be thrown.
     *
     * @param conn         This method requires working database connection.
     * @param valueObject  This parameter contains the class instance to be loaded.
     *                     Primary-key field must be set for this to work properly.
     */
    public void load(Connection conn, Estudiante valueObject) throws NotFoundException,
SQLException {

          String sql = "SELECT * FROM estudiante WHERE (codigo = ? ) ";
          PreparedStatement stmt = null;

          try {
               stmt = conn.prepareStatement(sql);
               stmt.setInt(1, valueObject.getCodigo());

               singleQuery(conn, stmt, valueObject);

          } finally {
              if (stmt != null)
                  stmt.close();
          }
    }


    /**
     * LoadAll-method. This will read all contents from database table and
     * build a List containing valueObjects. Please note, that this method
     * will consume huge amounts of resources if table has lot's of rows.
     * This should only be used when target tables have only small amounts
     * of data.
     *
     * @param conn         This method requires working database connection.
     */
    public List loadAll(Connection conn) throws SQLException {

          String sql = "SELECT * FROM estudiante ORDER BY codigo ASC ";
          List searchResults = listQuery(conn, conn.prepareStatement(sql));

          return searchResults;
    }
```

```
    /**
     * create-method. This will create new row in database according to supplied
     * valueObject contents. Make sure that values for all NOT NULL columns are
     * correctly specified. Also, if this table does not use automatic surrogate-keys
     * the primary-key must be specified. After INSERT command this method will
     * read the generated primary-key back to valueObject if automatic surrogate-keys
     * were used.
     *
     * @param conn         This method requires working database connection.
     * @param valueObject  This parameter contains the class instance to be created.
     *                     If automatic surrogate-keys are not used the Primary-key
     *                     field must be set for this to work properly.
     */
    public synchronized void create(Connection conn, Estudiante valueObject) throws
SQLException {

          String sql = "";
          PreparedStatement stmt = null;
          ResultSet result = null;

          try {
               sql = "INSERT INTO estudiante ( codigo, nombres, grado, "
               + "id) VALUES (?, ?, ?, ?) ";
               stmt = conn.prepareStatement(sql);

               stmt.setInt(1, valueObject.getCodigo());
               stmt.setString(2, valueObject.getNombres());
               stmt.setInt(3, valueObject.getGrado());
               stmt.setInt(4, valueObject.getId());

               int rowcount = databaseUpdate(conn, stmt);
               if (rowcount != 1) {
                    //System.out.println("PrimaryKey Error when updating DB!");
                    throw new SQLException("PrimaryKey Error when updating DB!");
               }

          } finally {
              if (stmt != null)
                  stmt.close();
          }


    }


    /**
     * save-method. This method will save the current state of valueObject to database.
     * Save can not be used to create new instances in database, so upper layer must
     * make sure that the primary-key is correctly specified. Primary-key will indicate
     * which instance is going to be updated in database. If save can not find matching
     * row, NotFoundException will be thrown.
     *
     * @param conn         This method requires working database connection.
     * @param valueObject  This parameter contains the class instance to be saved.
     *                     Primary-key field must be set for this to work properly.
     */
    public void save(Connection conn, Estudiante valueObject)
          throws NotFoundException, SQLException {

          String sql = "UPDATE estudiante SET nombres = ?, grado = ?, id = ? WHERE
(codigo = ? ) ";
```

```
            PreparedStatement stmt = null;

            try {
                stmt = conn.prepareStatement(sql);
                stmt.setString(1, valueObject.getNombres());
                stmt.setInt(2, valueObject.getGrado());
                stmt.setInt(3, valueObject.getId());

                stmt.setInt(4, valueObject.getCodigo());

                int rowcount = databaseUpdate(conn, stmt);
                if (rowcount == 0) {
                    //System.out.println("Object could not be saved! (PrimaryKey not
found)");
                    throw new NotFoundException("Object could not be saved! (PrimaryKey
not found)");
                }
                if (rowcount > 1) {
                    //System.out.println("PrimaryKey Error when updating DB! (Many
objects were affected!)");
                    throw new SQLException("PrimaryKey Error when updating DB! (Many
objects were affected!)");
                }
            } finally {
                if (stmt != null)
                    stmt.close();
            }
        }


        /**
         * delete-method. This method will remove the information from database as
identified by
         * by primary-key in supplied valueObject. Once valueObject has been deleted it can
not
         * be restored by calling save. Restoring can only be done using create method but
if
         * database is using automatic surrogate-keys, the resulting object will have
different
         * primary-key than what it was in the deleted object. If delete can not find
matching row,
         * NotFoundException will be thrown.
         *
         * @param conn         This method requires working database connection.
         * @param valueObject  This parameter contains the class instance to be deleted.
         *                     Primary-key field must be set for this to work properly.
         */
        public void delete(Connection conn, Estudiante valueObject)
                throws NotFoundException, SQLException {

            String sql = "DELETE FROM estudiante WHERE (codigo = ? ) ";
            PreparedStatement stmt = null;

            try {
                stmt = conn.prepareStatement(sql);
                stmt.setInt(1, valueObject.getCodigo());

                int rowcount = databaseUpdate(conn, stmt);
                if (rowcount == 0) {
                    //System.out.println("Object could not be deleted (PrimaryKey not
found)");
                    throw new NotFoundException("Object could not be deleted!
(PrimaryKey not found)");
```

```
            }
            if (rowcount > 1) {
                //System.out.println("PrimaryKey Error when updating DB! (Many
objects were deleted!)");
                throw new SQLException("PrimaryKey Error when updating DB! (Many
objects were deleted!)");
            }
        } finally {
            if (stmt != null)
                stmt.close();
        }
    }


    /**
     * deleteAll-method. This method will remove all information from the table that
matches
     * this Dao and ValueObject couple. This should be the most efficient way to clear
table.
     * Once deleteAll has been called, no valueObject that has been created before can
be
     * restored by calling save. Restoring can only be done using create method but if
database
     * is using automatic surrogate-keys, the resulting object will have different
primary-key
     * than what it was in the deleted object. (Note, the implementation of this method
should
     * be different with different DB backends.)
     *
     * @param conn         This method requires working database connection.
     */
    public void deleteAll(Connection conn) throws SQLException {

        String sql = "DELETE FROM estudiante";
        PreparedStatement stmt = null;

        try {
            stmt = conn.prepareStatement(sql);
            int rowcount = databaseUpdate(conn, stmt);
        } finally {
            if (stmt != null)
                stmt.close();
        }
    }


    /**
     * coutAll-method. This method will return the number of all rows from table that
matches
     * this Dao. The implementation will simply execute "select count(primarykey) from
table".
     * If table is empty, the return value is 0. This method should be used before
calling
     * loadAll, to make sure table has not too many rows.
     *
     * @param conn         This method requires working database connection.
     */
    public int countAll(Connection conn) throws SQLException {

        String sql = "SELECT count(*) FROM estudiante";
        PreparedStatement stmt = null;
        ResultSet result = null;
        int allRows = 0;
```

```
            try {
                stmt = conn.prepareStatement(sql);
                result = stmt.executeQuery();

                if (result.next())
                    allRows = result.getInt(1);
            } finally {
                if (result != null)
                    result.close();
                if (stmt != null)
                    stmt.close();
            }
            return allRows;
        }


        /**
         * searchMatching-Method. This method provides searching capability to
         * get matching valueObjects from database. It works by searching all
         * objects that match permanent instance variables of given object.
         * Upper layer should use this by setting some parameters in valueObject
         * and then  call searchMatching. The result will be 0-N objects in a List,
         * all matching those criteria you specified. Those instance-variables that
         * have NULL values are excluded in search-criteria.
         *
         * @param conn         This method requires working database connection.
         * @param valueObject  This parameter contains the class instance where search will
be based.
         *                     Primary-key field should not be set.
         */
        public List searchMatching(Connection conn, Estudiante valueObject) throws
SQLException {

            List searchResults;

            boolean first = true;
            StringBuffer sql = new StringBuffer("SELECT * FROM estudiante WHERE 1=1 ");

            if (valueObject.getCodigo() != 0) {
                if (first) { first = false; }
                sql.append("AND codigo = ").append(valueObject.getCodigo()).append(" ");
            }

            if (valueObject.getNombres() != null) {
                if (first) { first = false; }
                sql.append("AND nombres LIKE
'").append(valueObject.getNombres()).append("%' ");
            }

            if (valueObject.getGrado() != 0) {
                if (first) { first = false; }
                sql.append("AND grado = ").append(valueObject.getGrado()).append(" ");
            }

            if (valueObject.getId() != 0) {
                if (first) { first = false; }
                sql.append("AND id = ").append(valueObject.getId()).append(" ");
            }


            sql.append("ORDER BY codigo ASC ");
```

```
            // Prevent accidential full table results.
            // Use loadAll if all rows must be returned.
            if (first)
                searchResults = new ArrayList();
            else
                searchResults = listQuery(conn, conn.prepareStatement(sql.toString()));

            return searchResults;
        }


        /**
         * getDaogenVersion will return information about
         * generator which created these sources.
         */
        public String getDaogenVersion() {
            return "DaoGen version 2.4.1";
        }


        /**
         * databaseUpdate-method. This method is a helper method for internal use. It will
execute
         * all database handling that will change the information in tables. SELECT queries
will
         * not be executed here however. The return value indicates how many rows were
affected.
         * This method will also make sure that if cache is used, it will reset when data
changes.
         *
         * @param conn          This method requires working database connection.
         * @param stmt          This parameter contains the SQL statement to be excuted.
         */
        protected int databaseUpdate(Connection conn, PreparedStatement stmt) throws
SQLException {

            int result = stmt.executeUpdate();

            return result;
        }



        /**
         * databaseQuery-method. This method is a helper method for internal use. It will
execute
         * all database queries that will return only one row. The resultset will be
converted
         * to valueObject. If no rows were found, NotFoundException will be thrown.
         *
         * @param conn          This method requires working database connection.
         * @param stmt          This parameter contains the SQL statement to be excuted.
         * @param valueObject   Class-instance where resulting data will be stored.
         */
        protected void singleQuery(Connection conn, PreparedStatement stmt, Estudiante
valueObject)
            throws NotFoundException, SQLException {

            ResultSet result = null;

            try {
                result = stmt.executeQuery();
```

```
            if (result.next()) {

                 valueObject.setCodigo(result.getInt("codigo"));
                 valueObject.setNombres(result.getString("nombres"));
                 valueObject.setGrado(result.getInt("grado"));
                 valueObject.setId(result.getInt("id"));

            } else {
                  //System.out.println("Estudiante Object Not Found!");
                  throw new NotFoundException("Estudiante Object Not Found!");
            }
        } finally {
            if (result != null)
                result.close();
            if (stmt != null)
                stmt.close();
        }
    }


    /**
     * databaseQuery-method. This method is a helper method for internal use. It will
execute
     * all database queries that will return multiple rows. The resultset will be
converted
     * to the List of valueObjects. If no rows were found, an empty List will be
returned.
     *
     * @param conn         This method requires working database connection.
     * @param stmt         This parameter contains the SQL statement to be excuted.
     */
    protected List listQuery(Connection conn, PreparedStatement stmt) throws
SQLException {

        ArrayList searchResults = new ArrayList();
        ResultSet result = null;

        try {
            result = stmt.executeQuery();

            while (result.next()) {
                 Estudiante temp = createValueObject();

                 temp.setCodigo(result.getInt("codigo"));
                 temp.setNombres(result.getString("nombres"));
                 temp.setGrado(result.getInt("grado"));
                 temp.setId(result.getInt("id"));

                 searchResults.add(temp);
            }

        } finally {
            if (result != null)
                result.close();
            if (stmt != null)
                stmt.close();
        }

        return (List)searchResults;
    }


}
```

ProyectosDAO:

```
package persistencia;


import java.sql.*;
import java.util.*;
import java.math.*;


public class ProyectoDao {



    /**
     * createValueObject-method. This method is used when the Dao class needs
     * to create new value object instance. The reason why this method exists
     * is that sometimes the programmer may want to extend also the valueObject
     * and then this method can be overrided to return extended valueObject.
     * NOTE: If you extend the valueObject class, make sure to override the
     * clone() method in it!
     */
    public Proyecto createValueObject() {
          return new Proyecto();
    }


    /**
     * getObject-method. This will create and load valueObject contents from database
     * using given Primary-Key as identifier. This method is just a convenience method
     * for the real load-method which accepts the valueObject as a parameter. Returned
     * valueObject will be created using the createValueObject() method.
     */
    public Proyecto getObject(Connection conn, int id) throws NotFoundException,
SQLException {

          Proyecto valueObject = createValueObject();
          valueObject.setId(id);
          load(conn, valueObject);
          return valueObject;
    }


    /**
     * load-method. This will load valueObject contents from database using
     * Primary-Key as identifier. Upper layer should use this so that valueObject
     * instance is created and only primary-key should be specified. Then call
     * this method to complete other persistent information. This method will
     * overwrite all other fields except primary-key and possible runtime variables.
     * If load can not find matching row, NotFoundException will be thrown.
     *
     * @param conn         This method requires working database connection.
     * @param valueObject  This parameter contains the class instance to be loaded.
     *                     Primary-key field must be set for this to work properly.
     */
    public void load(Connection conn, Proyecto valueObject) throws NotFoundException,
SQLException {

          String sql = "SELECT * FROM proyecto WHERE (id = ? ) ";
          PreparedStatement stmt = null;
```

```
            try {
                 stmt = conn.prepareStatement(sql);
                 stmt.setInt(1, valueObject.getId());

                 singleQuery(conn, stmt, valueObject);

            } finally {
                if (stmt != null)
                    stmt.close();
            }
        }


        /**
         * LoadAll-method. This will read all contents from database table and
         * build a List containing valueObjects. Please note, that this method
         * will consume huge amounts of resources if table has lot's of rows.
         * This should only be used when target tables have only small amounts
         * of data.
         *
         * @param conn         This method requires working database connection.
         */
        public List loadAll(Connection conn) throws SQLException {

            String sql = "SELECT * FROM proyecto ORDER BY id ASC ";
            List searchResults = listQuery(conn, conn.prepareStatement(sql));

            return searchResults;
        }



        /**
         * create-method. This will create new row in database according to supplied
         * valueObject contents. Make sure that values for all NOT NULL columns are
         * correctly specified. Also, if this table does not use automatic surrogate-keys
         * the primary-key must be specified. After INSERT command this method will
         * read the generated primary-key back to valueObject if automatic surrogate-keys
         * were used.
         *
         * @param conn         This method requires working database connection.
         * @param valueObject  This parameter contains the class instance to be created.
         *                     If automatic surrogate-keys are not used the Primary-key
         *                     field must be set for this to work properly.
         */
        public synchronized void create(Connection conn, Proyecto valueObject) throws
SQLException {

            String sql = "";
            PreparedStatement stmt = null;
            ResultSet result = null;

            try {
                 sql = "INSERT INTO proyecto ( id, titulo, estado, "
                 + "problema, objetivos, metodologia, "
                 + "poblacion, resultados, observaciones) VALUES (?, ?, ?, ?, ?, ?, ?, ?,
?) ";
                 stmt = conn.prepareStatement(sql);

                 stmt.setInt(1, valueObject.getId());
                 stmt.setString(2, valueObject.getTitulo());
                 stmt.setString(3, valueObject.getEstado());
```

```
            stmt.setString(4, valueObject.getProblema());
            stmt.setString(5, valueObject.getObjetivos());
            stmt.setString(6, valueObject.getMetodologia());
            stmt.setString(7, valueObject.getPoblacion());
            stmt.setString(8, valueObject.getResultados());
            stmt.setString(9, valueObject.getObservaciones());

            int rowcount = databaseUpdate(conn, stmt);
            if (rowcount != 1) {
                //System.out.println("PrimaryKey Error when updating DB!");
                throw new SQLException("PrimaryKey Error when updating DB!");
            }

        } finally {
            if (stmt != null)
                stmt.close();
        }


    }


    /**
     * save-method. This method will save the current state of valueObject to database.
     * Save can not be used to create new instances in database, so upper layer must
     * make sure that the primary-key is correctly specified. Primary-key will indicate
     * which instance is going to be updated in database. If save can not find matching
     * row, NotFoundException will be thrown.
     *
     * @param conn         This method requires working database connection.
     * @param valueObject  This parameter contains the class instance to be saved.
     *                     Primary-key field must be set for this to work properly.
     */
    public void save(Connection conn, Proyecto valueObject)
        throws NotFoundException, SQLException {

        String sql = "UPDATE proyecto SET titulo = ?, estado = ?, problema = ?, "
            + "objetivos = ?, metodologia = ?, poblacion = ?, "
            + "resultados = ?, observaciones = ? WHERE (id = ? ) ";
        PreparedStatement stmt = null;

        try {
            stmt = conn.prepareStatement(sql);
            stmt.setString(1, valueObject.getTitulo());
            stmt.setString(2, valueObject.getEstado());
            stmt.setString(3, valueObject.getProblema());
            stmt.setString(4, valueObject.getObjetivos());
            stmt.setString(5, valueObject.getMetodologia());
            stmt.setString(6, valueObject.getPoblacion());
            stmt.setString(7, valueObject.getResultados());
            stmt.setString(8, valueObject.getObservaciones());

            stmt.setInt(9, valueObject.getId());

            int rowcount = databaseUpdate(conn, stmt);
            if (rowcount == 0) {
                //System.out.println("Object could not be saved! (PrimaryKey not
found)");
                throw new NotFoundException("Object could not be saved! (PrimaryKey
not found)");
            }
            if (rowcount > 1) {
```

```
                    //System.out.println("PrimaryKey Error when updating DB! (Many
objects were affected!)");
                    throw new SQLException("PrimaryKey Error when updating DB! (Many
objects were affected!)");
                }
            } finally {
                if (stmt != null)
                    stmt.close();
            }
    }


    /**
     * delete-method. This method will remove the information from database as
identified by
     * by primary-key in supplied valueObject. Once valueObject has been deleted it can
not
     * be restored by calling save. Restoring can only be done using create method but
if
     * database is using automatic surrogate-keys, the resulting object will have
different
     * primary-key than what it was in the deleted object. If delete can not find
matching row,
     * NotFoundException will be thrown.
     *
     * @param conn         This method requires working database connection.
     * @param valueObject  This parameter contains the class instance to be deleted.
     *                     Primary-key field must be set for this to work properly.
     */
    public void delete(Connection conn, Proyecto valueObject)
            throws NotFoundException, SQLException {

            String sql = "DELETE FROM proyecto WHERE (id = ? ) ";
            PreparedStatement stmt = null;

            try {
                stmt = conn.prepareStatement(sql);
                stmt.setInt(1, valueObject.getId());

                int rowcount = databaseUpdate(conn, stmt);
                if (rowcount == 0) {
                    //System.out.println("Object could not be deleted (PrimaryKey not
found)");
                    throw new NotFoundException("Object could not be deleted!
(PrimaryKey not found)");
                }
                if (rowcount > 1) {
                    //System.out.println("PrimaryKey Error when updating DB! (Many
objects were deleted!)");
                    throw new SQLException("PrimaryKey Error when updating DB! (Many
objects were deleted!)");
                }
            } finally {
                if (stmt != null)
                    stmt.close();
            }
    }


    /**
     * deleteAll-method. This method will remove all information from the table that
matches
```

```
     * this Dao and ValueObject couple. This should be the most efficient way to clear
table.
     * Once deleteAll has been called, no valueObject that has been created before can
be
     * restored by calling save. Restoring can only be done using create method but if
database
     * is using automatic surrogate-keys, the resulting object will have different
primary-key
     * than what it was in the deleted object. (Note, the implementation of this method
should
     * be different with different DB backends.)
     *
     * @param conn         This method requires working database connection.
     */
    public void deleteAll(Connection conn) throws SQLException {

          String sql = "DELETE FROM proyecto";
          PreparedStatement stmt = null;

          try {
              stmt = conn.prepareStatement(sql);
              int rowcount = databaseUpdate(conn, stmt);
          } finally {
              if (stmt != null)
                  stmt.close();
          }
    }


    /**
     * coutAll-method. This method will return the number of all rows from table that
matches
     * this Dao. The implementation will simply execute "select count(primarykey) from
table".
     * If table is empty, the return value is 0. This method should be used before
calling
     * loadAll, to make sure table has not too many rows.
     *
     * @param conn         This method requires working database connection.
     */
    public int countAll(Connection conn) throws SQLException {

          String sql = "SELECT count(*) FROM proyecto";
          PreparedStatement stmt = null;
          ResultSet result = null;
          int allRows = 0;

          try {
              stmt = conn.prepareStatement(sql);
              result = stmt.executeQuery();

              if (result.next())
                  allRows = result.getInt(1);
          } finally {
              if (result != null)
                  result.close();
              if (stmt != null)
                  stmt.close();
          }
          return allRows;
    }
```

```
    /**
     * searchMatching-Method. This method provides searching capability to
     * get matching valueObjects from database. It works by searching all
     * objects that match permanent instance variables of given object.
     * Upper layer should use this by setting some parameters in valueObject
     * and then  call searchMatching. The result will be 0-N objects in a List,
     * all matching those criteria you specified. Those instance-variables that
     * have NULL values are excluded in search-criteria.
     *
     * @param conn         This method requires working database connection.
     * @param valueObject  This parameter contains the class instance where search will
be based.
     *                     Primary-key field should not be set.
     */
    public List searchMatching(Connection conn, Proyecto valueObject) throws
SQLException {

          List searchResults;

          boolean first = true;
          StringBuffer sql = new StringBuffer("SELECT * FROM proyecto WHERE 1=1 ");

          if (valueObject.getId() != 0) {
              if (first) { first = false; }
              sql.append("AND id = ").append(valueObject.getId()).append(" ");
          }

          if (valueObject.getTitulo() != null) {
              if (first) { first = false; }
              sql.append("AND titulo LIKE
'%").append(valueObject.getTitulo()).append("%' ");
          }

          if (valueObject.getEstado() != null) {
              if (first) { first = false; }
              sql.append("AND estado LIKE
'").append(valueObject.getEstado()).append("%' ");
          }

          if (valueObject.getProblema() != null) {
              if (first) { first = false; }
              sql.append("AND problema LIKE
'").append(valueObject.getProblema()).append("%' ");
          }

          if (valueObject.getObjetivos() != null) {
              if (first) { first = false; }
              sql.append("AND objetivos LIKE
'").append(valueObject.getObjetivos()).append("%' ");
          }

          if (valueObject.getMetodologia() != null) {
              if (first) { first = false; }
              sql.append("AND metodologia LIKE
'").append(valueObject.getMetodologia()).append("%' ");
          }

          if (valueObject.getPoblacion() != null) {
              if (first) { first = false; }
              sql.append("AND poblacion LIKE
'").append(valueObject.getPoblacion()).append("%' ");
          }
```

```
            if (valueObject.getResultados() != null) {
                if (first) { first = false; }
                sql.append("AND resultados LIKE
'").append(valueObject.getResultados()).append("%' ");
            }

            if (valueObject.getObservaciones() != null) {
                if (first) { first = false; }
                sql.append("AND observaciones LIKE
'").append(valueObject.getObservaciones()).append("%' ");
            }


            sql.append("ORDER BY id ASC ");

            // Prevent accidential full table results.
            // Use loadAll if all rows must be returned.
            if (first)
                 searchResults = new ArrayList();
            else
                 searchResults = listQuery(conn, conn.prepareStatement(sql.toString()));

            return searchResults;
        }


        /**
         * getDaogenVersion will return information about
         * generator which created these sources.
         */
        public String getDaogenVersion() {
            return "DaoGen version 2.4.1";
        }


        /**
         * databaseUpdate-method. This method is a helper method for internal use. It will
execute
         * all database handling that will change the information in tables. SELECT queries
will
         * not be executed here however. The return value indicates how many rows were
affected.
         * This method will also make sure that if cache is used, it will reset when data
changes.
         *
         * @param conn         This method requires working database connection.
         * @param stmt         This parameter contains the SQL statement to be excuted.
         */
        protected int databaseUpdate(Connection conn, PreparedStatement stmt) throws
SQLException {

            int result = stmt.executeUpdate();

            return result;
        }



        /**
         * databaseQuery-method. This method is a helper method for internal use. It will
execute
         * all database queries that will return only one row. The resultset will be
converted
```

```
     * to valueObject. If no rows were found, NotFoundException will be thrown.
     *
     * @param conn         This method requires working database connection.
     * @param stmt         This parameter contains the SQL statement to be excuted.
     * @param valueObject  Class-instance where resulting data will be stored.
     */
    protected void singleQuery(Connection conn, PreparedStatement stmt, Proyecto
valueObject)
          throws NotFoundException, SQLException {

          ResultSet result = null;

          try {
              result = stmt.executeQuery();

              if (result.next()) {

                   valueObject.setId(result.getInt("id"));
                   valueObject.setTitulo(result.getString("titulo"));
                   valueObject.setEstado(result.getString("estado"));
                   valueObject.setProblema(result.getString("problema"));
                   valueObject.setObjetivos(result.getString("objetivos"));
                   valueObject.setMetodologia(result.getString("metodologia"));
                   valueObject.setPoblacion(result.getString("poblacion"));
                   valueObject.setResultados(result.getString("resultados"));
                   valueObject.setObservaciones(result.getString("observaciones"));

              } else {
                    //System.out.println("Proyecto Object Not Found!");
                    throw new NotFoundException("Proyecto Object Not Found!");
              }
          } finally {
              if (result != null)
                  result.close();
              if (stmt != null)
                  stmt.close();
          }
    }


    /**
     * databaseQuery-method. This method is a helper method for internal use. It will
execute
     * all database queries that will return multiple rows. The resultset will be
converted
     * to the List of valueObjects. If no rows were found, an empty List will be
returned.
     *
     * @param conn         This method requires working database connection.
     * @param stmt         This parameter contains the SQL statement to be excuted.
     */
    protected List listQuery(Connection conn, PreparedStatement stmt) throws
SQLException {

          ArrayList searchResults = new ArrayList();
          ResultSet result = null;

          try {
              result = stmt.executeQuery();

              while (result.next()) {
                   Proyecto temp = createValueObject();
```

```
                    temp.setId(result.getInt("id"));
                    temp.setTitulo(result.getString("titulo"));
                    temp.setEstado(result.getString("estado"));
                    temp.setProblema(result.getString("problema"));
                    temp.setObjetivos(result.getString("objetivos"));
                    temp.setMetodologia(result.getString("metodologia"));
                    temp.setPoblacion(result.getString("poblacion"));
                    temp.setResultados(result.getString("resultados"));
                    temp.setObservaciones(result.getString("observaciones"));

                    searchResults.add(temp);
                }

            } finally {
                if (result != null)
                    result.close();
                if (stmt != null)
                    stmt.close();
            }

            return (List)searchResults;
        }


}
```

DataAccessObject:

```
package persistencia;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.util.List;

abstract class DataAccessObject {

       abstract Object createValueObject();

       abstract Object getObject(Connection conn, int id_Object)
                    throws NotFoundException, SQLException;

       abstract void load(Connection conn, Object valueObject)
                    throws NotFoundException, SQLException;

       abstract List loadAll(Connection conn) throws SQLException;

       abstract void create(Connection conn, Object valueObject)
                    throws SQLException;

       abstract void save(Connection conn, Object valueObject)
                    throws NotFoundException, SQLException;

       abstract void delete(Connection conn, Object valueObject)
                    throws NotFoundException, SQLException;

       abstract void deleteAll(Connection conn) throws SQLException;

       abstract int countAll(Connection conn) throws SQLException;

       abstract List searchMatching(Connection conn, Object valueObject)
```

```
                    throws SQLException;

        abstract String getDaogenVersion();

        abstract int databaseUpdate(Connection conn, PreparedStatement stmt)
                    throws SQLException;

        abstract void singleQuery(Connection conn, PreparedStatement stmt,
                    Object valueObject) throws NotFoundException, SQLException;

        abstract List listQuery(Connection conn, PreparedStatement stmt)
                    throws SQLException;

        abstract int getNextID(Connection conn) throws SQLException;
}
```

NotFoundException:

```
package persistencia;


import java.util.*;


public class NotFoundException extends Exception {

   /**
    * Constructor for NotFoundException. The input message is
    * returned in toString() message.
    */
    public NotFoundException(String msg) {
           super(msg);
    }

}
```

A continuación se debe compilar el proyecto tal y como se hizo en el primer ejemplo. Guarde cambios y ejecute los siguientes comandos:

```
root@cpehva:~# cd ProyectosPrueba/

root@cpehva:~/ProyectosPrueba# ant all
```

Si todo sale bien debe obtener un mensaje de `BUILD SUCCESSFULL`

```
BUILD SUCCESSFUL

Total time: 29 seconds

root@cpehva:~/ProyectosPrueba#


root@cpehva:~# chmod 777 -R /usr/local/globus-4.0.8/etc/cagrid_ProyectosPrueba/

root@cpehva:~#
```

Luego regresar a introduce y hacer clic en Deploy Service, figura E25.

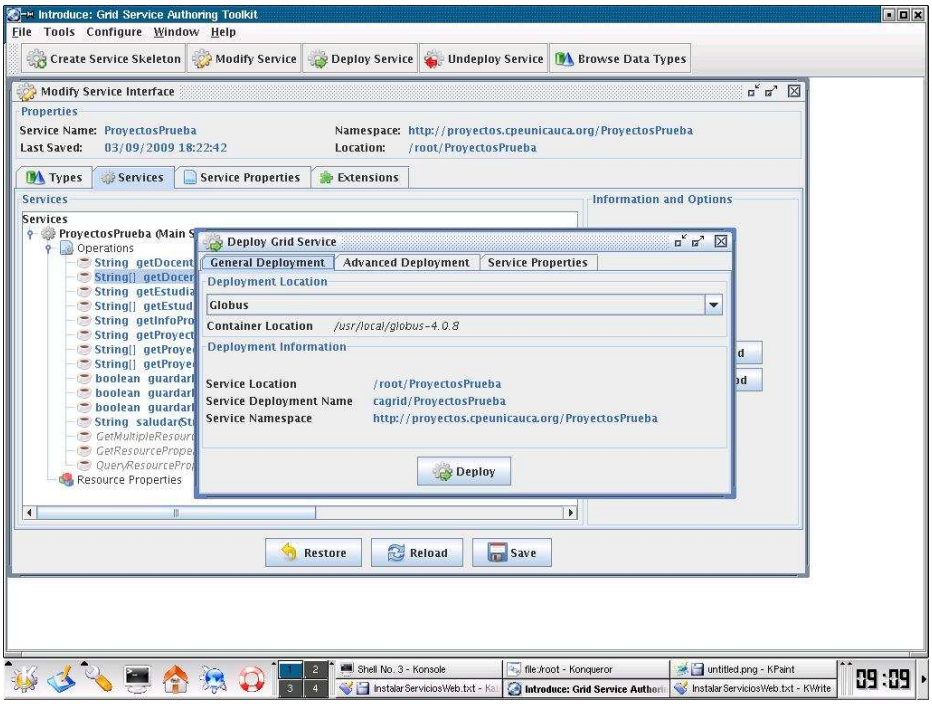

Figura E25: Deploy Service.

Una vez terminada esta parte es necesario cambiar los permisos del archivo de despliegue ubicado en `/usr/local/globus-4.0.8/etc/` mediante el comando:

```
root@cpehva:~# chmod 777 -R /usr/local/globus-4.0.8/etc/cagrid_ProyectosPrueba/
```

Finalmente iniciamos el contenedor de servicios web de globus con el comando:

```
root@cpehva:~# su globus

globus@cpehva:/root$ /etc/init.d/globus-4.0.8 start

Password:

Starting Globus container. PID: 12446

globus@cpehva:/root$
```

El archivo descriptor WSDL del servicio queda alojado en la dirección: http://192.168.1.14:8443/wsrf/services/cagrid/ProyectosPrueba?wsdl

Desde un equipo cliente cualquiera abra un navegador e ingrese esta dirección obtendrá el WSDL.

La creación del cliente para consumir el servicio es idéntico al anterior ejemplo solo que cambia la wsdl por la del proyecto acutal, ejemplo:

http://192.168.1.14:8443/wsrf/services/cagrid/ProyectosPrueba?wsdl

NOTA: Para que los servicios web puedan acceder a la base de datos es necesario modificar el archvio `pg_hba.conf` de tal manera que se permita las conexiones externas al motor de bases de datos.

Referencias

[1] caGrid, Introduce 2009 [En Línea]. Disponible: http://www.cagrid.org/display/introduce/Home [Consulta: marzo, 2009].

[2] TitanicLinux.Net, 2003 [En línea]. Disponible: www.titaniclinux.net [Consulta: marzo, 2009].