Práctica 0: Entorno software para Sistemas Informáticos II

Índice

Objetivos

El objetivo de esta práctica es enumerar el software en el que se desarrollarán las prácticas de la asignatura. Se asume que la mayor parte de este software ya ha sido cubierto por otras asignaturas. En caso de no ser así, se ofrecen apéndices con información extendida:

Se estudiará una introducción o repaso a los siguientes aspectos:

El servidor de aplicaciones ***Oracle Glassfish Server V4.0***. De este las siguientes interfaces:

- En línea de comandos: *asadmin*, para administración del servidor web y *ant*, para la construcción de proyectos en Java
- Como interfaz web: *Admin Console* (consola de administración del servidor).

El DBMS ***PostgreSQL 8.4.8***. De este las siguientes interfaces:

- En línea de comandos: *psql, createdb, dropdb,…* para conexión, creación y borrado de bases de datos
- Como interfaz gráfica: *Tora*

Jakarta JMeter v2.5.1 como herramienta gráfica de generación de carga y test.

El entorno de virtualización ***VMWare Player v6.0.2***, incluyendo una máquina virtual ***Ubuntu Server 10.04.3 LTS*** con todas las aplicaciones anteriores instaladas.

El servidor web Apache como balanceador de carga externo al servidor de aplicaciones

No es necesario entregar nada al completar la práctica. El objetivo es sólo que el estudiante **sea capaz de reconocer, instalar y utilizar cada uno de estos elementos**. La dedicación estimada para esta práctica es de 2 horas presenciales y 3 horas no presenciales.

Entorno software

Para realizar las prácticas en el laboratorio **se podrá utilizar únicamente la partición de Linux**. Los programas que es necesario utilizar ya se encuentran instalados en dicha partición. Sin embargo, habrá algunos pasos de configuración del sistema que será preciso ejecutar cada vez que se inicie el ordenador, ya que este no permite almacenar estos archivos tras realizar dicho reinicio. Estos pasos se indicarán cuando se explique la aplicación que los requiere.

A lo largo de las prácticas se distinguirá entre los siguientes entornos, habituales en aplicaciones empresariales:

- **Entorno de desarrollo:** El entorno software donde trabajan los desarrolladores de las aplicaciones y realizan las pruebas funcionales de las mismas. ***En nuestro caso, el entorno de desarrollo son PCs de las aulas (x86 con Ubuntu Desktop 14.04 LTS).***
- **Entorno de producción:** Es el entorno donde se ejecuta la aplicación para prestar el servicio para el que está diseñada a los usuarios finales. Normalmente es un entorno **remoto**, con fuertes restricciones de **seguridad en el acceso** y **cambios**, ubicado en la infraestructura de producción del cliente, y con grandes capacidades de monitorización y gestión. ***En estas prácticas no contamos con este entorno, pero lo simularemos empleando una máquina virtual VMWare Player v6.0.2 ejecutando Ubuntu Server 10.04.3 LTS dentro del mismo PC***

Nota: Existen varias versiones de Glassfish y de Java EE. En todos los enunciados de las prácticas, “Glassfish” se refiere a Oracle Glassfish Server v4.0, y “Java EE” a Java EE 8, a no ser que se indique explícitamente lo contrario.

Introducción a Glassfish y Java EE

Oracle Glassfish Server V4.0 es un servidor de aplicaciones (*application server*) desarrollado íntegramente en Java, con tecnología Java EE (antes J2EE). Se trata de una aplicación que ofrece entornos de ejecución para *servlets*, JSPs y EJBs; servicios generales de seguridad para todos los elementos; capacidad de replicación y distribución de aplicaciones entre múltiples servidores; y herramientas centralizadas de configuración, administración y control. En estas prácticas se asume que el alumno está familiarizado con él. No obstante puede revisarse el Apéndice I para ampliar información.

En este laboratorio estudiaremos Glassfish desde el punto de vista del arquitecto de software, no tanto desde la perspectiva de un desarrollador Java.

La siguiente figura detalla los diferentes elementos que componen una instancia del servidor de GlassFish. Algunos de estos elementos serán vistos con mayor detalle a lo largo de las siguientes prácticas:

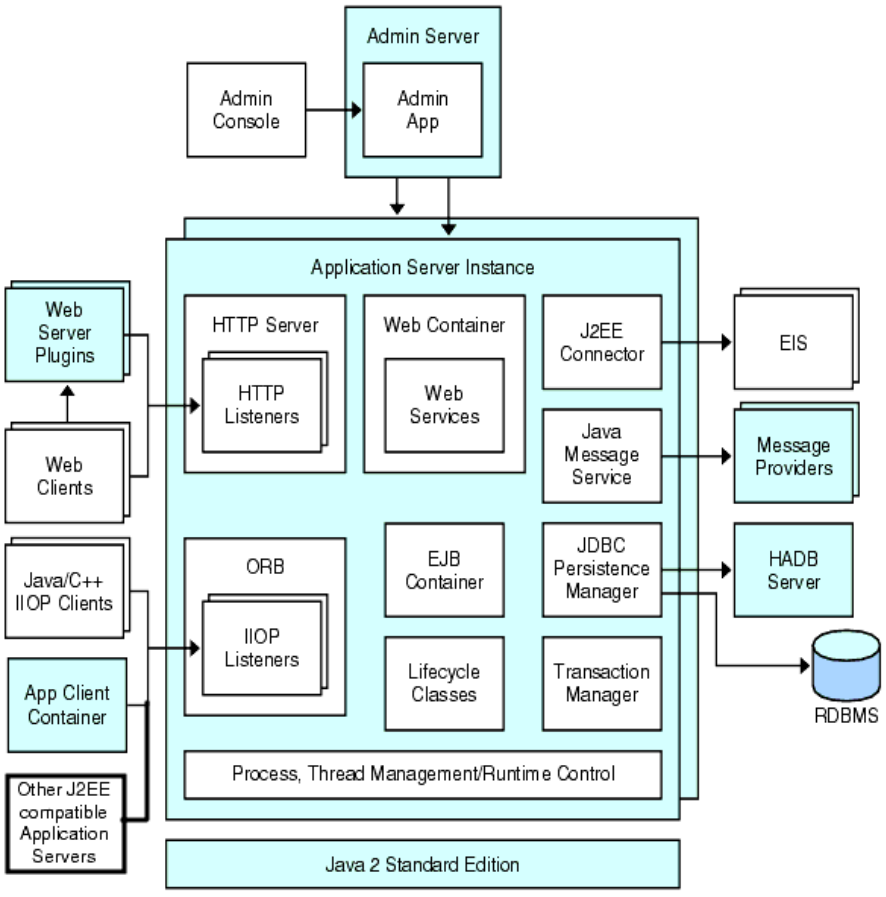

Figura 1: Estructura interna completa de una instancia del servidor Glassfish

En lo que sigue, nos referiremos al directorio donde se encuentra instalado el Glassfish como *<install-dir-j2ee>*. En los laboratorios, este directorio es:

```
/usr/local/glassfish-4.1/glassfish
```

En la máquina virtual descrita más adelante, este directorio es:

```
/opt/glassfish4/glassfish
```

En los laboratorios es necesario declarar la variable de entorno J2EE_HOME con el valor de dicho directorio de instalación. Esta variable indica donde encontrar el comando asadmin y los paquetes de la plataforma J2EE. Habría que definirla mediante el comando:

```
export J2EE_HOME=/usr/local/glassfish-4.1/glassfish
```

(un fallo común es olvidar exportar esta variable de entorno, apareciendo entonces errores al intentar compilar, por no encontrarse los paquetes de la plataforma J2EE)

La herramienta ant

Ant es el equivalente para Java de *make*; el fichero *build.xml* es el equivalente a los *Makefiles.* Al igual que *make* se ejecuta (normalmente) en un directorio que contiene un *Makefile*, *ant* se ejecuta en un directorio con *build.xml* (aunque como con *make,* estas opciones por defecto pueden ser cambiadas mediante argumentos de línea de comandos). La forma de usar *ant* es semejante a la de *make* en que su argumento se refiere a un objetivo (*target*) del fichero de construcción del proyecto; aunque en *ant* se habla de *tasks* (tareas) en vez de *targets.* Por ejemplo, *ant setup* ejecuta la tarea *setup* del *build.xml, ant run* la tarea *run*.

Existen sin embargo significativas diferencias entre *ant* y *make*:

- *ant* está implementado completamente en Java, y entiende la estructura por defecto de los proyectos Java, incluida la estructura de directorios asociada a los paquetes Java.
- *ant* usa ficheros xml para la construcción de proyectos, lo cual le da una semántica mucho más "limpia".
- Los dos aspectos anteriores permiten (con ciertas matizaciones, nada es perfecto) mucha más portabilidad en la definición de los proyectos, y el uso de un solo fichero de construcción para múltiples plataformas.
- Al contrario que *make*, donde los objetivos se construyen mediante secuencias de comandos arbitarias de la shell del sistema, las tareas en *ant* están también en Java, lo que de nuevo permite la independencia de plataforma y también la extensibilidad de *ant* mediante la definición de nuevas tareas.

PostgreSQL

En el entorno de las prácticas se empleará PostgreSQL como gestor de bases de datos.

A lo largo del curso se asume que el alumno ya conoce y maneja dicho gestor de bases de datos, así como las siguientes herramientas de línea de comandos:

- psql: Cliente de línea de comandos para acceder a la base de datos. Por ejemplo, para conectarnos a las base de datos "DB" con el usuario de PostgreSQL *alumnodb* previamente creado, escribiremos:
  ```
  psql –U alumnodb DB
  ```
- createdb: Comando para crear una nueva base de datos. Al igual que el anterior, empleará el parámetro –U para indicar qué usuario se quiere emplear. Para crear la base de datos "DB" se deberá ejecutar:
  ```
  createdb –U alumnodb DB
  ```
- dropdb: Comando para eliminar una nueva base de datos. Sigue el patrón de los comandos anteriores. Para borrar la base de datos "DB" se deberá ejecutar:
  ```
  dropdb –U alumnodb DB
  ```

Conexión JDBC con PostgreSQL

En prácticas sucesivas emplearemos el *driver* JDBC para conectar a dicho gestor, que se encuentra instalado en la máquina virtual:

```
/opt/glassfish4/glassfish/domains/domain1/libs/postgresql-jdbc4.jar
```

En sucesivas prácticas se proporcionará también este driver para hacer el despliegue. Este driver nos proporcionará entre otras las siguientes clases de interés para posteriores prácticas:

```
org.postgresql.Driver
```

```
org.postgresql.ds.PGConnectionPoolDataSource
```

TORA

Tora es el acrónimo de *Toolkit for* Oracle. Se trata de una potente herramienta de administración y desarrollo con bases de datos, para conectar a múltiples DBMS.

A lo largo de estas prácticas podremos optar por cualquier herramienta gráfica, aunque a continuación presentamos *tora* como una alternativa para ello.

Antes de probar la conexión con Tora siguiendo los pasos descritos a continuación, es necesario que creemos una base de datos de nombre DB para el usuario alumnodb en la máquina virtual. Para ello, deberemos utilizar el comando createdb descrito anteriormente. El usuario alumnodb ya está creado en PostgreSQL en la máquina virtual.

Conexión desde Tora

Para conectar al gestor de bases de datos iniciaremos la aplicación tora desde el menú de aplicaciones, o bien escribiendo *tora* desde la línea de comandos. A continuación se nos mostrará un diálogo para acceder a la base de datos, similar a este:

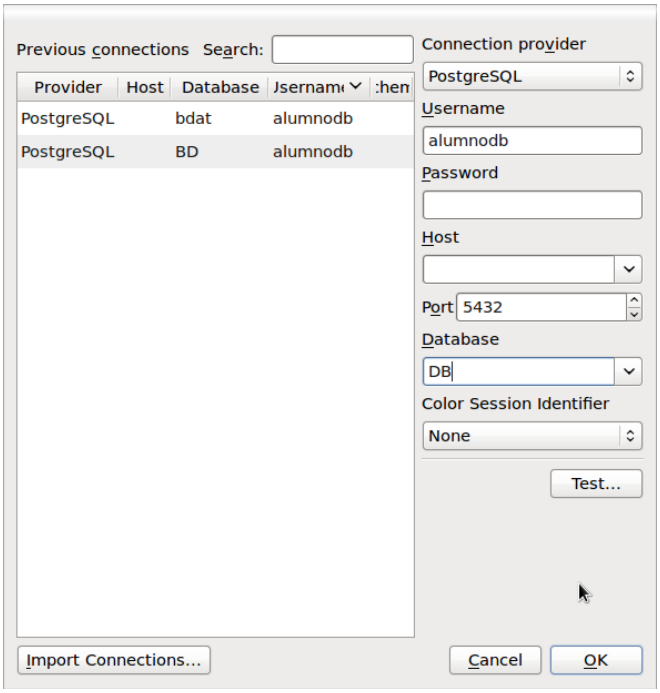

Figura 2: *Conectando a PostgreSQL con Tora*

Deberemos:

- Seleccionar el tipo de DBMS en el desplegable “Connection provider” (en nuestro caso, PostgreSQL)
- Indicar el nombre de usuario: *alumnodb*
- Indicar la contraseña, en nuestro caso, vacía
- Indicar el host de conexión: si la conexión es local, deberemos dejar en blanco dicho cuadro de texto. Préstese especial atención a que *localhost* no es un valor válido en la configuración de los laboratorios.
- Pulsar OK

Si todo va bien, accederemos a la aplicación con una pantalla similar a esta:

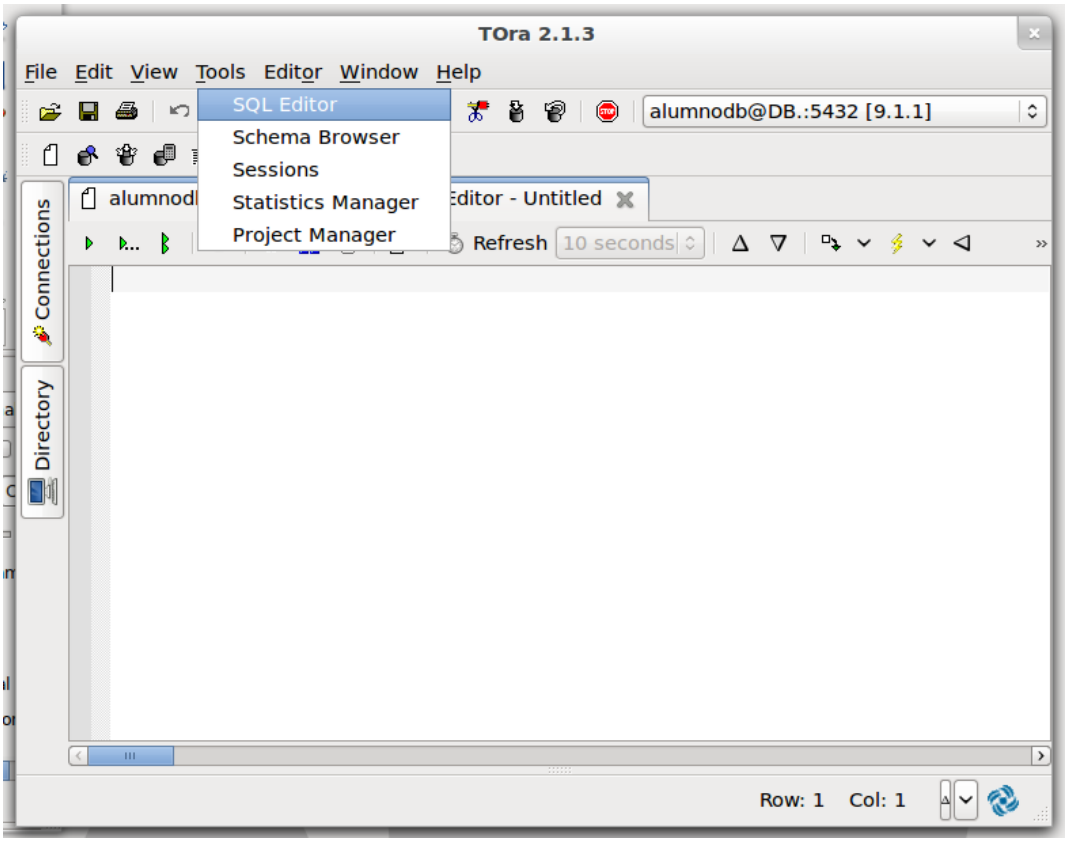

Figura 3: Vista de editor SQL en Tora

Haremos uso de las siguientes **dos vistas** de la base de datos:

- Editor SQL (*SQL Editor*): Permite ejecutar consultas SQL. Esto se realiza seleccionando la consulta y pulsando en los iconos triangulares de la esquina superior izquierda, para ejecutar una consulta (bajo el cursor) o todas las consultas a partir del cursor. También se puede hacer lo mismo seleccionando texto y pulsando *Ctrl-Enter*
- Navegador del esquema (*Schema browser*). Permite revisar la estructura de tablas y columnas, realizando modificaciones en dicho esquema. Se accede pulsando *Tools -> Schema browser*

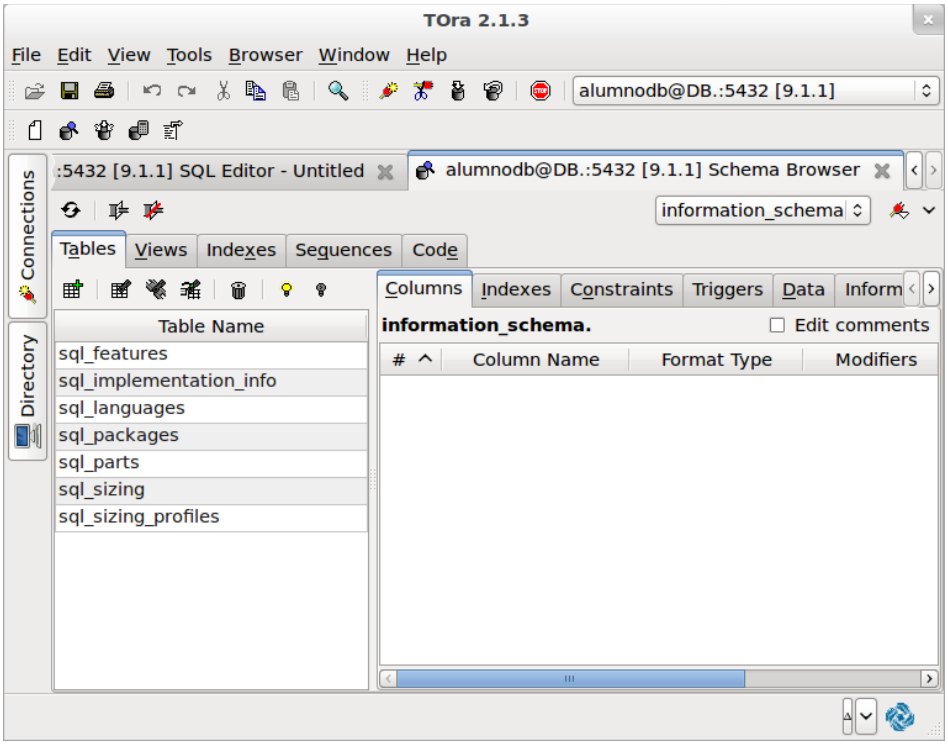

Figura 4: Vista de esquema en Tora

El software de virtualización VMWare Player

Este software de virtualización es usado profesionalmente para virtualizar sistemas operativos completos, es decir, ejecutar concurrentemente varias instancias separadas del sistema operativo sobre el mismo hardware con mínimo impacto sobre el rendimiento del mismo. Distinguimos dos entidades fundamentales:

- **Sistema anfitrión o *host***: aquel bajo el que se está ejecutando el propio software de virtualización (en el caso de los laboratorios: Ubuntu Desktop 14.04 LTS)
- **Sistema invitado o *guest***: aquel que es virtualizado y ejecutado *dentro* de la máquina virtual. En nuestro caso se trata de una instalación de Ubuntu Server 10.04.3 LTS con todo lo necesario para poder realizar las prácticas de esta asignatura. Habitualmente nos referiremos a éste simplemente como la *Máquina Virtual* o VM.

La manera habitual de conectar con la VM es utilizando TCP/IP. Ésta incluye interfaces de red "virtuales" para realizar esta función. Se distinguen tres tipos:

- **Interfaz host-only**: una interfaz de muy bajo *overhead* que permite la conexión únicamente con el PC host. En nuestras prácticas no la usaremos por simplicidad y porque no es realista en cuanto a su rendimiento
- **Interfaz NAT**: permite que la VM se conecte al exterior utilizando la propia interfaz del PC host y mediante traducción de direcciones de red. Sin embargo, en principio no permite la operación inversa (conexiones entrantes, desde el PC host hacia la VM). Nuestra configuración incluye una de estas "tarjetas" (eth0) para su propio mantenimiento de actualizaciones.
- **Interfaz puente o bridge**: es una interfaz de red con un funcionamiento idéntico a la del propio PC. Utiliza su propia dirección MAC (configurable por el usuario) y todos los PCs (virtuales o no) que compartan segmento de red podrán ver dicha dirección MAC en sus tablas arp y, por tanto, acceder a la máquina virtual. En la VM del laboratorio utilizaremos esta interfaz de red para acceder a la VM, por HTTP o SSH para tareas administrativas.

El software VMWare Player se encuentra en la opción del menú (Aplicaciones -> Herramientas del sistema), o bien ejecutando "vmplayer" en la línea de comandos.

Importante: La virtualización anidada (ejecutar una máquina virtual dentro de otra máquina virtual) no es posible con VMWare ni en la arquitectura x86… al menos a la fecha.

Importación del servicio virtualizado

Para simplificar las cosas, se ha preparado una máquina virtual (en adelante VM). El formato empleado es un único fichero si2srv.tgz de aproximadamente 1.1GB, y que tras *importarse* puede ocupar entre 1.5GB y 2GB en función de las modificaciones que se le haga.

La imagen se puede descargar desde la página moodle del curso.

Los pasos para importar la imagen son los siguientes:

1. En un terminal, descomprimir la imagen virtual sobre el directorio $HOME del usuario o cualquiera donde este tenga permisos de escritura. Por ejemplo, la siguiente secuencia de comandos nos descomprimirá la imagen virtual en $HOME/si2srv/

   ```
   cd $HOME

   tar –xzvf /opt/si2/si2srv.tgz
   ```

2. Iniciar VMWare Player
3. Abrir la imagen virtual (File -> Open) y acceder hasta el fichero $HOME/si2srv/si2srv.vmx

Importante: Los PCs del laboratorio eliminan todo contenido del usuario con cada reinicio. Por ese motivo, el alumno deberá transportar la imagen importada en su propio medio de almacenamiento externo (ver más adelante)

Configuración de red

Importante: Este paso es fundamental, ya que todas las imágenes virtuales de los PC de laboratorio son idénticas en cuanto a sus direcciones MAC, lo cual llevará a conflictos de duplicación de direcciones salvo que se tomen precauciones.

Antes de usar la imagen virtual, debemos realizar los siguientes pasos:

1. Seleccionar la imagen importada (si2srv)
2. Pulsar en el botón "Edit virtual machine settings"
3. Revisar las interfaces de red en el recuadro izquierdo: debiera haber dos: Network Adapter (NAT) y Network Adapter 2 (Bridged)

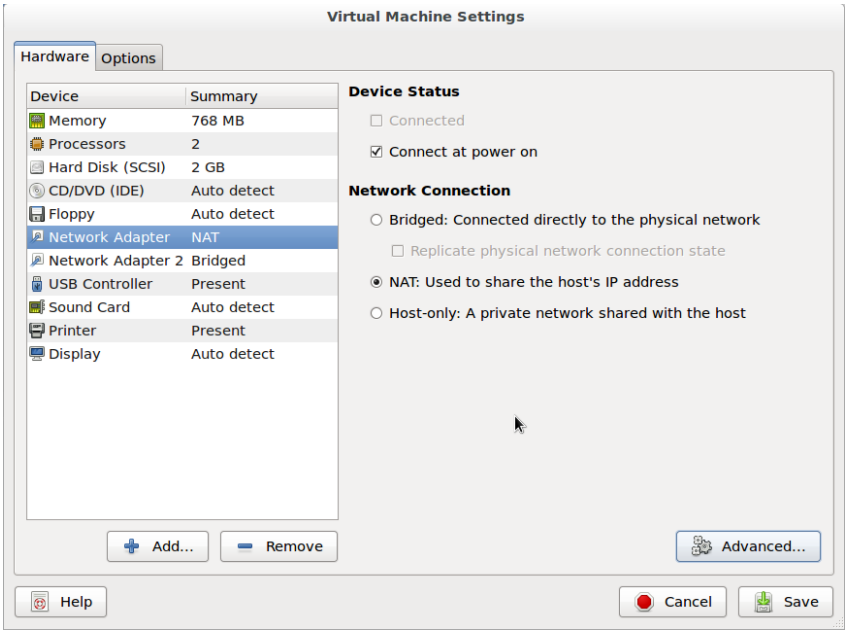

Figura 5: Propiedades de la máquina virtual, interfaces de red

4. Seleccionar la primera interfaz (NAT) y pulsar en opciones avanzadas

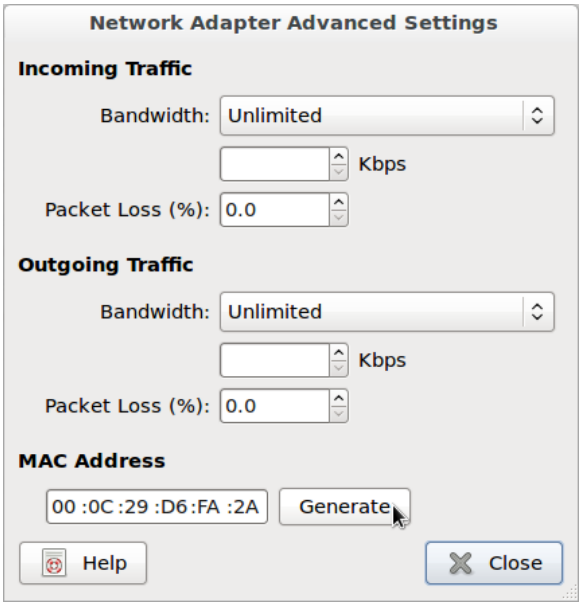

Figura 6: Propiedades avanzadas de una interfaz de red

5. Pulsamos el botón "*Generate*" junto a "*Mac Address*". Esto debería generar una dirección MAC nueva y aleatoria.

6. **Se repite la operación con el segundo adaptador (interfaz *bridged*)**

Nota: Otra opción para cambiar la dirección MAC es utilizar el script que se proporciona junto con la máquina virtual, en la carpeta donde se encuentra el archivo .vmxl. Dicho script asigna una dirección MAC en particular a la máquina virtual al pasarle como argumentos el grupo, la pareja y el nº de PC que se quiere utilizar para desplegar la máquina virtual.

`./si2fixMAC.sh <grupo> <pareja> <PC>`

(por ejemplo: `./si2fixMAC.sh 2312 12 2` ; nótese que **no se deben añadir ceros a la izquierda**: '2' y no '02')

Transportar la máquina virtual

La VM generada puede ser transportada opcionalmente en un medio externo (USB) del propio alumno. Para ello basta con que se copie la carpeta si2srv al medio externo y se recupere de vuelta cuando sea necesario.

Importante: Antes de copiar la carpeta con la máquina virtual, ésta debe estar parada. Para detenerla, hacer login en la máquina virtual y ejecutar (por ejemplo):

```
sudo halt
```

Se puede reducir el tamaño de la imagen comprimiéndola, por ejemplo con `gzip`.

Ejecutar la máquina virtual

Para comenzar, abrimos la VM si2srv y pulsamos el botón "*Play virtual machine*"

Tras el proceso habitual de inicio de un sistema Ubuntu Server, se os mostrará el *prompt* de inicio de sesión de línea de comandos:

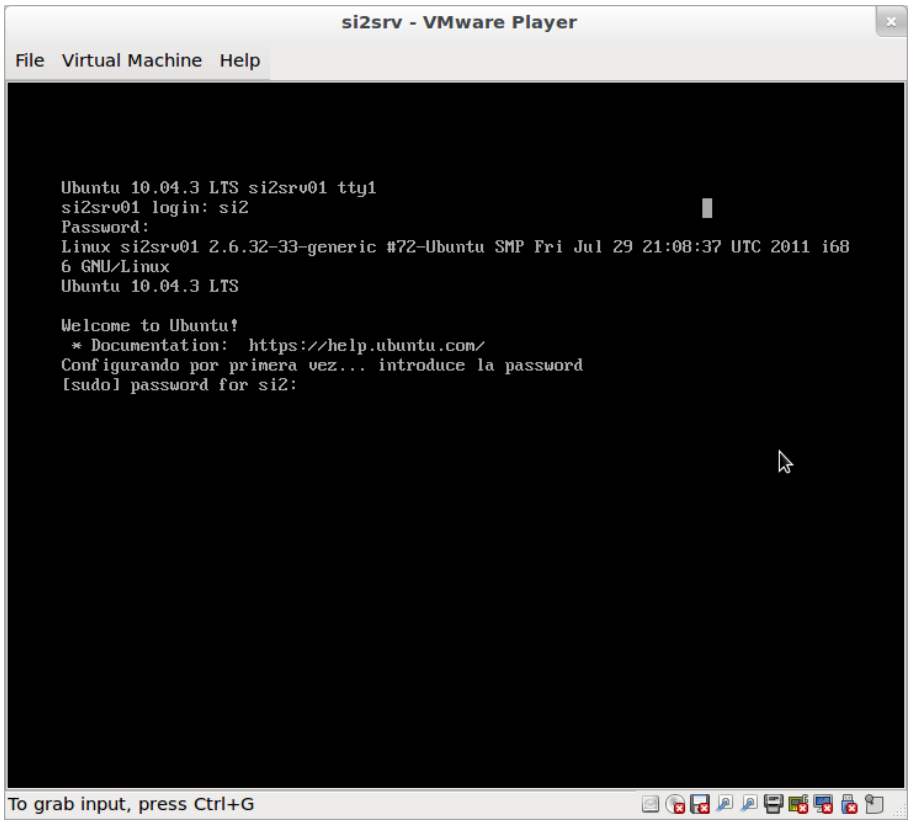

Figura 7: Consola de si2srv

El usuario a emplear es *si2* y su contraseña *2017sid0s* (tener en cuenta que la última o de d0s es un cero).

La primera vez que se inicie esta imagen (o cada vez que se haga en el laboratorio a partir de la imagen .tgz) habrá que completar un asistente de configuración cuyo objetivo es **asignar una dirección IP única y sin conflictos**. Durante el uso del asistente podemos movernos por las opciones usando el **ratón**. **Se**

deberá usar la combinación de teclas Ctrl-Alt para abandonar dicha máquina virtual y retornar el ratón al PC host.

Los pasos serán los siguientes:

1. Elegir el grupo de prácticas de entre los mostrados.

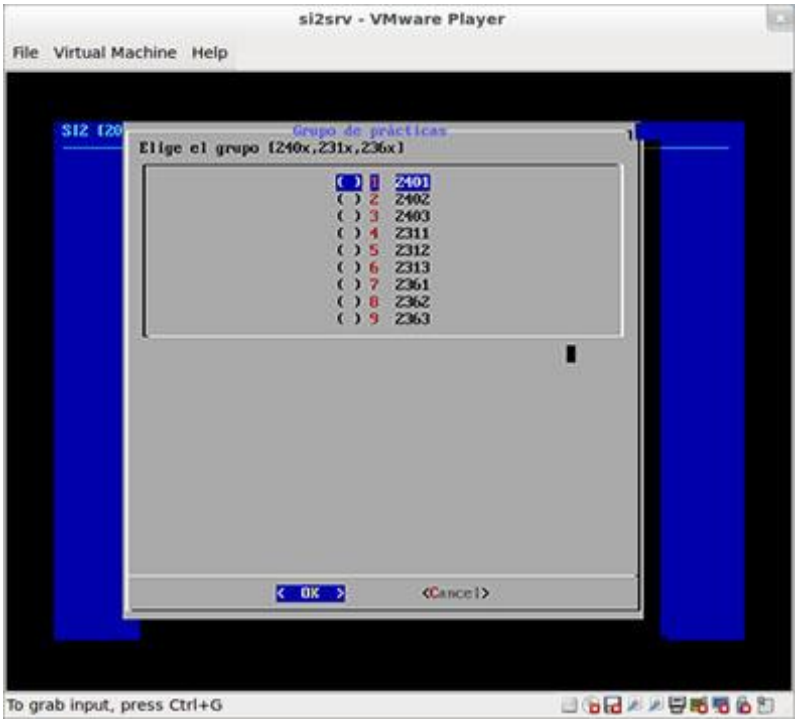

Figura 8: Selección de grupo

2. Elegir pareja dentro del grupo. Teclearla, **sin ceros a la izquierda** (‘1’ y no ‘01’), y luego pulsar *Entrar*.

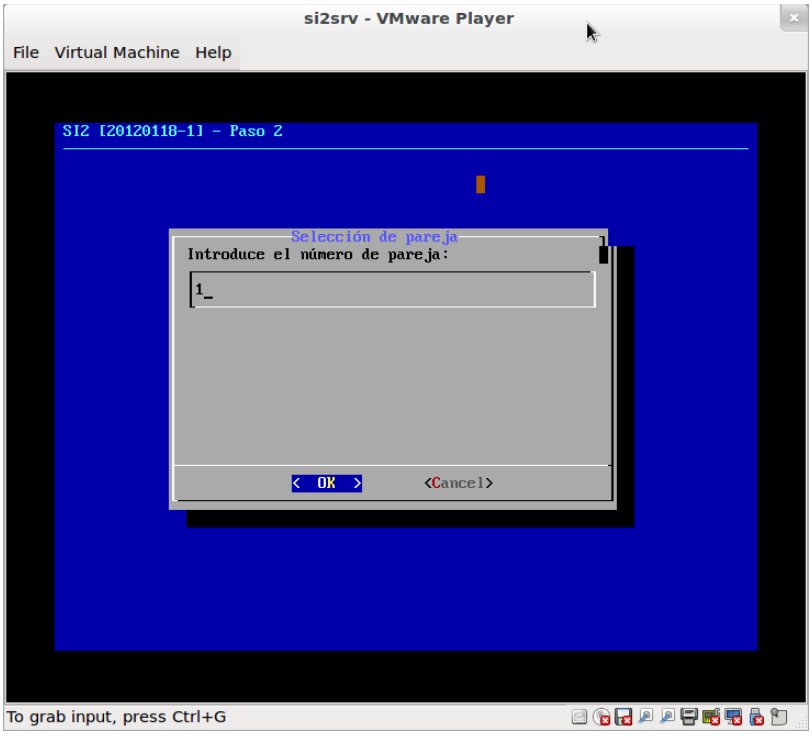

Figura 9: Selección de pareja

3. Elegir número de PC virtual dentro de la pareja.. Pulsaremos *Entrar*.

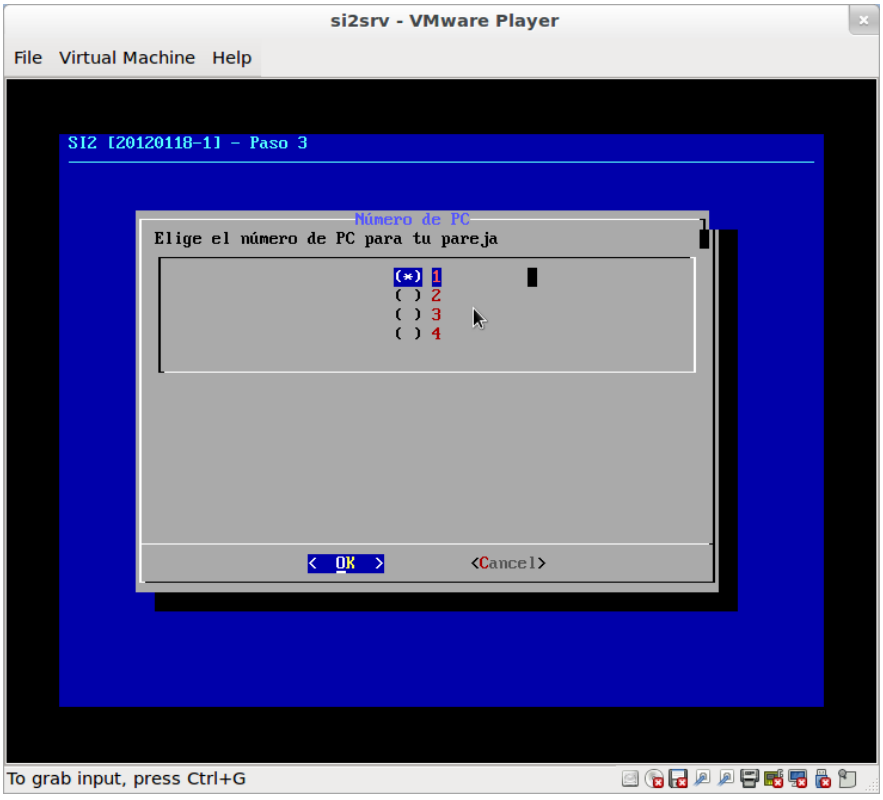

Figura 10: Selección de PC de la pareja

4. Verificar que la dirección que se ha asignado es de la forma 10.X.Y.Z donde X.Y.Z tienen relación con el grupo, pareja y número de PC seleccionado, respectivamente.

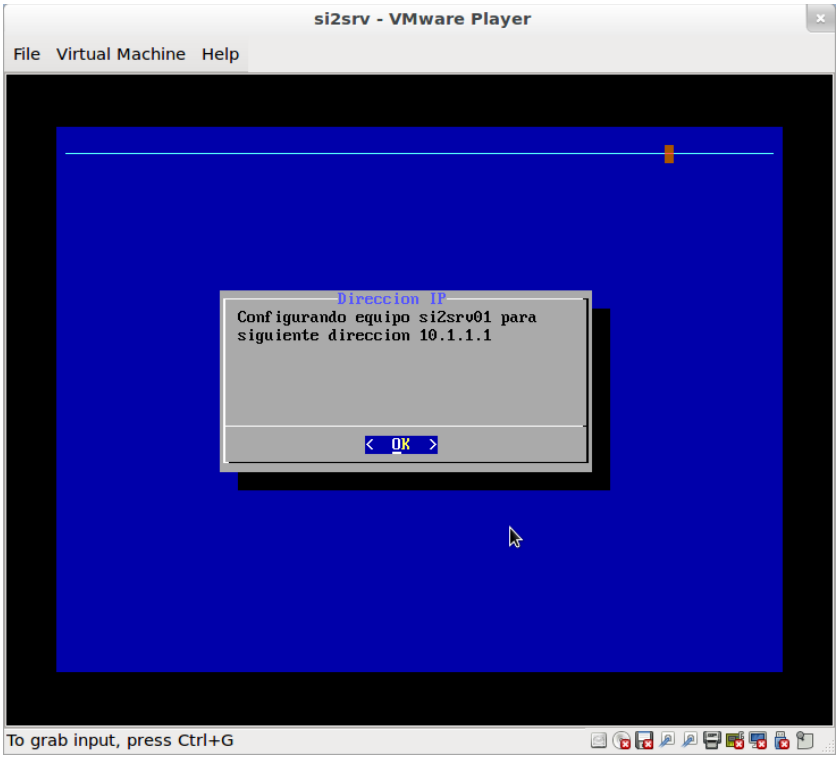

Figura 11: Direcciones IP asignadas

5. Pulsar *Entrar*, lo cual devuelve al prompt del sistema.

6. Reiniciar la máquina virtual con el comando:

```
sudo reboot
```

Acceso remoto a la máquina virtual

Importante: No conviene utilizar la máquina virtual desde la consola de VMWare, debido a su bajo rendimiento sin aceleración 2D y al hecho de que el teclado está en inglés. Si se quiere hacer operaciones sobre la misma, se deberá utilizar acceso SSH contra dicha máquina virtual. Para ello se ejecutará desde un terminal del PC host:

Antes de acceder será necesario asignar a la interfaz de red del Host una dirección IP en el rango 10.X.Y.Z:

```
ifconfig eth0:0
      eth0:0 Link encap:Ethernet HWaddr 00:aa:bb:cc:dd:ee
      inet addr:10.10.X.Y Bcast:10.255.255.255 Mask:255.0.0.0
      UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
```

Para ello podemos ejecutar el siguiente comando en los PCs de los laboratorios:

```
sudo /opt/si2/virtualip.sh eth0
```

Tras comprobar que la nueva interfaz está disponible con el comando `ifconfig`, es recomendable comprobar que la máquina virtual es accesible con el comando ping:

`ping 10.X.Y.Z` (donde 10.X.Y.Z será la dirección asignada anteriormente)

Si todo ha ido bien, accederemos de forma remota a la máquina virtual con el comando:

`ssh si2@10.X.Y.Z` (donde 10.X.Y.Z será la dirección asignada anteriormente)

La contraseña será la indicada anteriormente.

Iniciar y detener el servidor Glassfish en la máquina virtual

- **Iniciar el servidor:** Con el siguiente comando en un terminal:

  ```
  asadmin start-domain domain1
  ```

- **Detener el servidor:** con el siguiente comando:

  ```
  asadmin stop-domain domain1
  ```

- **Acceder al servidor:** Las aplicaciones serán accesibles a partir de http://10.X.Y.Z:8080/app donde app es el contexto de la aplicación que queremos ejecutar
- **Acceder al servidor o consola de administración:** Se deberá acceder[1]a http://10.X.Y.Z:4848

Importante: en los laboratorios y en la máquina virtual el comando **asadmin** se encuentra en el PATH (ruta de búsqueda de programas). En otras instalaciones podría ser necesario anteponer al comando la ruta absoluta a <install-dir-j2ee>/bin.

En la VM facilitada no es el caso, pero en otras instalaciones del servidor glassfish puede que sea necesario habilitar la administración remota con el mandato:

```
asadmin enable-secure-admin
```

Acceso al servidor por SSL

Acceder a la página https://10.X.Y.Z:8181 (nótese el http**s**). Debido a que Glassfish usa un certificado de seguridad autofirmado (en lugar de estar firmado por una autoridad de certificación externa fiable o CA, *Certification Authority*, como por ejemplo Verisign, que es lo necesario para tramitar pagos en Internet o servicios seguros con la Administración), un buen navegador como Firefox no lo aceptará de inmediato y solicitará todo tipo de confirmaciones de que se desea proceder. Tras confirmar la excepción, aparecerá una página similar a la de http://10.X.Y.Z:8080 con la importante diferencia de que accede en modo seguro y encriptado (via SSL: Secure Sockets Layer)

Otros aspectos de Glassfish

Los diferentes dominios se encuentran en

<*install-dir-j2ee*>/domains.

El dominio por defecto es *domain1*, que se encuentra en

<*install-dir-j2ee*>/domains/domain1.

Cada dominio tiene un perfil por defecto, Enterprise, Cluster, o Developer; y puede correr una o más instancias de servidor (por defecto, es decir, el "servidor predeterminado" que se invoca desde el menú, se llama "server" y está asociado a *domain1*).

El log de errores se encuentra en

[1] **Importante:** En la máquina virtual (usuario:***admin***, contraseña: ***adminadmin***)

<*install-dir-j2ee*>/domains/domain1/logs/server.log.

Importante: Se recomienda revisar los logs ante cualquier incidencia en el despliegue o ejecución de la aplicación, para obtener más información sobre qué está pasando.

La página que se presenta al acceder a http://10.X.Y.Z:8080 es la página *index.html* del "document root folder" del servidor virtual por defecto de este dominio. Se encuentra en:

<*install-dir-j2ee*>`/domains/domain1/docroot/index.html`.

Las aplicaciones *web* se instalan por defecto en el directorio:

<*install-dir-j2ee*>`/domains/domain1/applications/j2ee-modules/`

Para acceder a un *servlet*, el URL del navegador deberá apuntar a:

http://10.X.Y.Z:8080/contexto_aplicación_web/nombre_del_servlet

asadmin tiene multitud de opciones que poco a poco se irán presentando. Puede ejecutarse directamente o de forma interactiva. Para el modo interactivo, simplemente teclear *asadmin* en línea de comandos para entrar en una *shell* propia en la que se pueden introducir cualquiera de sus comandos. Ya se ha visto *start-domain, stop-domain* (sin argumentos, se aplican al dominio por defecto)*, help* lista todos los comandos disponibles; *exit* permite salir de la consola de *asadmin* y volver a la shell del sistema; *list-domains* permite verificar que domain1 está ejecutándose.

Selección del Idioma

Todas las capturas de pantalla que se presentan en los enunciados de prácticas están en inglés. Para poder ver las opciones de la misma forma en los laboratorios, se recomienda configurar el navegador para que use el inglés de forma predeterminada. Para ello en Firefox se deberá seguir la siguiente secuencia de menús:

Editar->Preferencias->Contenido->Idioma

Elegir idioma inglés, subirlo como opción preferente (flecha de arriba)

Aceptar

Apéndice I: Arquitectura de GlassFish

A continuación un pequeño glosario de términos que iremos revisando a lo largo de sucesivas prácticas

Servidor de aplicaciones. Plataforma de software para la ejecución de aplicaciones del lado servidor y servicios Web.

Java EE (Java Platform Enterprise Edition). API y entorno de ejecución construido sobre Java SE (Java Standard Edition, que contiene la funcionalidad básica de Java) para el desarrollo y ejecución de aplicaciones de red distribuidas, multicapa, escalables y seguras.

Servlets. Clases Java que siguen la Java Servlet API, protocolo por el que una clase Java procesa y responde a peticiones http (*http requests*). Es la alternativa Java a la programación CGI, ASP, etc.

JSPs. Java Server Pages, permiten mezclar contenido estático y dinámico en páginas html. En contraste, los servlets, programas CGI, etc. requieren que el programa genere la página html completa. Los JSPs son semejantes en este aspecto al uso de php dentro de páginas html con la etiqueta <?php>, pero usan la etiqueta <%=>. Se ejecutan por completo en el servidor, compilando el código en un servlet que es lo único a lo que accede el cliente (la compilación se realiza la primera vez que se accede a una JSP, lo cual lógicamente se supone que hará el desarrollador para probar la página antes de ponerla en producción). Permiten separar la parte de diseño (html) de la lógica de la aplicación.

Recurso JDBC (Java DataBase Connectivity). Recurso empleado para conexión con servidores de bases de datos (DBMS).

EJBs (Enterprise Java Beans). Componentes reusables Java para la lógica de negocio de la aplicación en el servidor.

Recurso JMS (Java Message Service). Recurso empleado para comunicación con otros procesos de forma desacoplada. También nos referiremos a estos como colas de mensajes o MQ (*Message Queues*)

Arquitectura de una aplicación en Java EE

La arquitectura de una aplicación Java EE suele articularse en torno a tres capas: la capa cliente, la capa intermedia, y la capa de datos. Las aplicaciones de Java EE se concentran en la **capa intermedia** o ***middleware***, donde a su vez podemos distinguir la capa web y la capa de negocio (la "lógica" de la aplicación). La primera gestiona la interacción entre la capa cliente y la capa de negocio, e incluye los servlets y JSPs, y también otros componentes como las Java Server Faces, centrados en la interfaz de usuario, y Java Beans. La segunda gestiona la funcionalidad principal de la aplicación, con EJBs y otros componentes, y a su vez interacciona con la capa de datos, donde pueden estar incluidos todo tipo de bases de datos y otros sistemas empresariales.

Un servidor de aplicaciones de Java EE, como Glassfish, aloja un conjunto de componentes de la aplicación que corresponden a cada capa. El servidor de aplicaciones proporciona servicios a estos componentes en la forma de *contenedores*:

- Contenedores de Java EE: proporcionan la interfaz entre el componente y la funcionalidad definida por la plataforma de Java EE para ese componente.
- El contenedor Web: interfaz entre los componentes web (como servlets, JSPs y páginas de Java Server Faces) y el servidor web. Gestiona el ciclo de vida del componente, envía peticiones a los componentes de aplicación y proporciona una interfaz para obtener datos sobre estas peticiones y su contexto.
- El contenedor de Cliente de Aplicación: interfaz entre los clientes de aplicación y el servidor de aplicaciones, que se ejecuta en la máquina cliente.
- El contenedor EJB: interfaz entre las EJBs y el servidor, que se ejecuta en el servidor.

Para más detalles sobre esta arquitectura, puede consultarse la segunda y tercera sección del tutorial *"Your First Cup: An Introduction to the Java EE Platform"*, que está incluido con la instalación de Glassfish y puede encontrarse también en la sección de Bibliografía *recomendada*.

Un tutorial mucho más completo sobre Java EE, que servirá de material de referencia para las prácticas es el Java EE Tutorial (de nuevo, ver la Bibliografía *recomendada*).

Arquitectura de Glassfish

La arquitectura de Glassfish se presenta en la figura 12. Se detallan a continuación los conceptos básicos necesarios para entenderla.

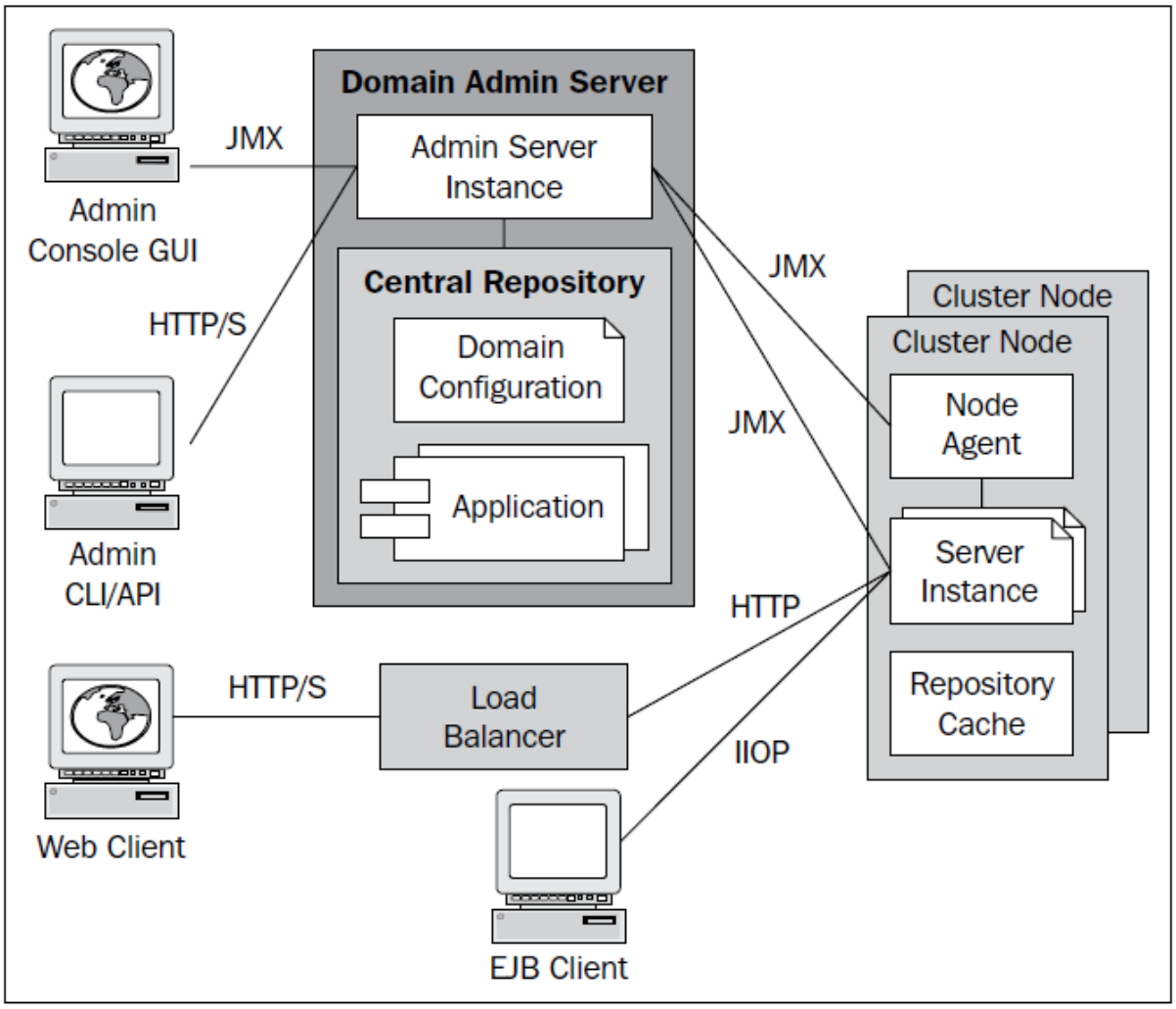

Figura 12: Arquitectura de Glassfish

Dominios administrativos

Un dominio administrativo (o, simplemente, dominio) es un grupo de una o múltiples instancias de servidor (*server instances*) administradas de forma conjunta. Una instancia de servidor pertenece siempre a un único dominio, pero las instancias de servidor de un único dominio pueden ejecutarse en diferentes hosts físicos. Cada dominio tiene un servidor de administración del dominio, y una o múltiples instancias de servidor.

Servidor de Administración de Dominio (DAS)

El Servidor de Administración de Dominio *(Domain Administration Server, DAS)* de un dominio es una instancia especial de servidor, normalmente dedicada a alojar aplicaciones administrativas, como la aplicación web de la Admin Console y otras herramientas de administración. El DAS es responsable de autenticar al administrador, gestionar peticiones de tipo administrativo, y comunicarse con otras instancias del servidor en el dominio para ejecutar tareas administrativas. Toda esta comunicación se basa en la Java Management Extension (JMX).

Clusters

Un *cluster* es simplemente una colección, con un nombre asociado, de instancias de servidor que comparten las mismas aplicaciones, recursos, e información de configuración.

Node agents

Un *node agent* es un proceso ligero que se ejecuta en cada uno de los hosts físicos donde se ejecutan instancias de GlassFish Server, incluyendo el ordenador donde se encuentra el DAS. El *node agent* es responsable, entre otras, de las siguientes tareas: crear instancias de servidor en el host en el que reside, iniciarlas y pararlas (instruido por el DAS); obtener información de diagnóstico; y sincronizar el almacén o repositorio de configuración local de cada instancia de servidor con el almacén de configuración central del DAS.

Instancias de servidor

Una instancia de servidor es un GlassFish Server que se ejecuta en un único proceso de la Java Virtual Machine (JVM). La instancia de servidor implementa contenedores para los componentes Java EE, incluyendo el contenedor web para los módulos de aplicación web y el contenedor EJB para los módulos EJB. Además, la instancia de servidor proporciona la capacidad de habilitar el acceso de clientes y la gestión de recursos.

Desde el punto de vista de las aplicaciones a desarrollar en estas prácticas, los elementos que entran en juego son, de forma simplificada, los siguientes:

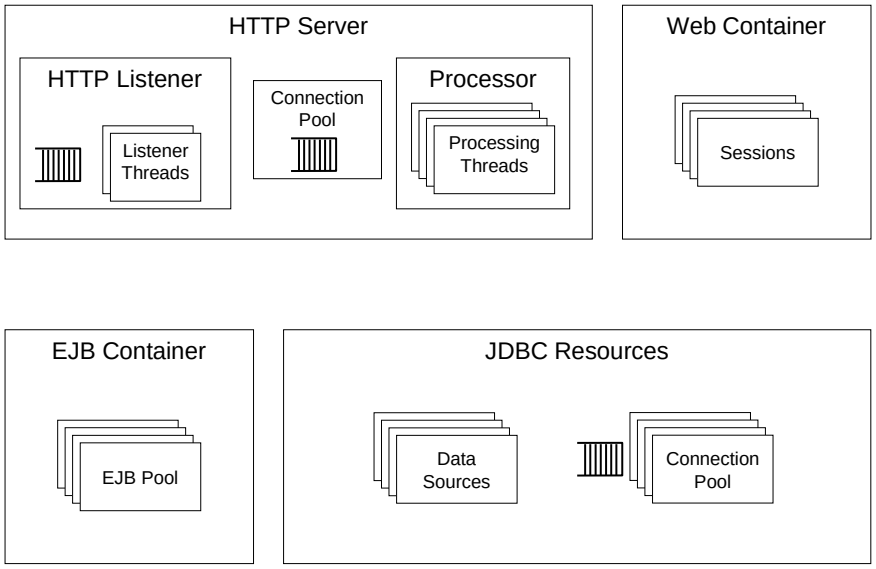

Figura 13: Elementos que entran en juego en el desarrollo de una aplicación que se ejecute sobre Glassfish

El **servicio HTTP** es el proceso que se encarga de la ejecución de peticiones de los usuarios a través del protocolo HTTP. Consta de dos elementos:

- Los subprocesos de recepción de llamadas, *HTTP listeners*. Esperan la recepción de peticiones en el puerto correspondiente. Una vez aceptadas las peticiones, las pasan a la *Connection Pool*, donde esperan a ser atendidas por el servidor.
- El servidor HTTP, propiamente dicho. Extrae las peticiones de la *Connection Pool* y realiza el proceso solicitado sobre los hilos de proceso.

El **contenedor Web** es el contenedor de los objetos necesarios para la ejecución de servlets y JSPs.

El **contenedor EJB** es el contenedor de los Enterprise Java Beans.

Los **recursos JDBC** son los que centralizan el acceso de la aplicación a las bases de datos (pero no son el propio gestor de bases de datos, que se ejecuta en un proceso independiente).

Los **recursos JMS** son los que centralizan el acceso de la aplicación a colas de mensajes. Glassfish incluye un gestor de colas de mensaje, aunque otros gestores externos pueden ser utilizados (ej: IBM WebSphere MQ)

Apéndice II: Instalación del entorno software

Importante: Todo el software que se describe a continuación **ya está instalado en los laboratorios** así como en la máquina virtual que se distribuye a los alumnos. No obstante, se incluye un breve recorrido sobre los procedimientos de instalación de los mismos, como ayuda **para que el alumno pueda reproducirlo** en su equipo personal.

Java SE 8

Es necesario como punto de partida contar con una versión del entorno de desarrollo de Java. Puede ser descargado:

http://www.oracle.com/technetwork/java/javase/downloads/index.html

No obstante, en distribuciones basadas en apt (Debian/Ubuntu), puede ser instalado (como root) mediante el comando:

```
sudo add-apt-repository ppa:webupd8team/java
sudo apt-get update
sudo apt-get install oracle-java8-installer
sudo apt-get install oracle-java8-set-default
```

Glassfish v4.0

Se puede descargar esta versión de:

http://download.java.net/glassfish/4.0/release/glassfish-4.0-unix.sh

La página principal de descargas para Glassfish, con la última versión, 4.1, se encuentra en:

https://glassfish.java.net/download.htm

Las versiones anteriores se hallan en:

https://glassfish.java.net/download-archive.html

A lo largo de las prácticas haremos referencia siempre a la versión en inglés, con sus opciones, parámetros y menús (donde proceda).

Importante: Las prácticas únicamente funcionan **únicamente** en la versión **v4.0**. **No se permite el uso de versiones distintas a ésta para la realización de las prácticas.**

La instalación se realiza ejecutando (como usuario root, por ejemplo mediante el mandato `sudo`):

```
sudo sh glassfish-4.0-unix.sh
```

El instalador solicitará el directorio de instalación, en nuestro caso: **/usr/local/glassfish-4.0**

Puesto que la ruta de instalación puede variar, a partir de ahora, y en el resto de prácticas nos referiremos a este directorio como ***<install-dir-j2ee>***.

Conviene dar permisos de escritura a todos los usuarios:

```
sudo chmod -R a+rwX /usr/local/glassfish-4.0
```

PostgreSQL y TORA

Ambos paquetes se encuentran disponibles en los repositorios estándar de Ubuntu. Para instalarlos ejecutaremos:

```
sudo apt-get install postgresql tora libqt4-sql-psql libpg-java
```

Otras herramientas

Hay otro software de monitorización instalado en cada PC que son de utilidad, tales como openssh-server nmon, sysstat, la herramienta de compilación ant, el editor vim y el intérprete para el lenguaje de scripting *regina-rexx* que emplearemos en prácticas sucesivas:

```
sudo apt-get install nmon sysstat regina-rexx unzip openssh-server vim ant
```

Jakarta JMeter 2.5.1

Este programa es una herramienta de generación de carga y testing que usaremos en posteriores prácticas. De momento, si queremos instalarlo en nuestro PC particular, descargaremos el paquete jakarta-jmeter-2.5.1.tgz de:

http://jmeter.apache.org/download_jmeter.cgi

Eligiendo de entre las opciones la versión binaria 2.5.1.tgz:

http://archive.apache.org/dist/jakarta/jmeter/binaries/jakarta-jmeter-2.5.1.tgz

A continuación lo descomprimimos en la ruta de nuestra elección, por ejemplo, la carpeta personal, mediante el comando:

```
cd $HOME
tar –xzvf jakarta-jmeter-2.5.1.tgz
```

Para ejecutarlo lanzamos el script **jmeter.sh**:

```
~/jakarta-jmeter/bin/jmeter.sh
```

VMWare Player v6.0.2

Descargamos el paquete de:

https://www.vmware.com/tryvmware/index.php

Se requiere registro, pero el software es gratuito. Hay diversas opciones de instalación en función del sistema operativo, distribución y versión de la misma (32b, 64b). En los laboratorios se emplea la versión x86 32 bits VMware-Player-6.0.2-1744117.i386.bundle

Deberemos ejecutar dicho fichero .bundle como root:

```
sudo sh VMware-Player-6.0.2-1744117.i386.bundle
```

Conexiones contra la VM si2srv desde el PC host

La máquina virtual levanta una interfaz bridge en el rango IP 10.X.Y.Z

Por tanto, para acceder a ésta, el PC host requiere una IP en el mismo rango.

Todos los PC's del laboratorio realizan dicha tarea al ejecutar el script virtualip.sh, con direcciones 10.10.a.b siendo a.b coincidentes en numeración con la IP del propio PC en el rango habitual de los laboratorios.

Para facilitar la tarea al alumno en su propio PC, se proporciona el siguiente script *virtualip.sh* (se encuentra disponible en la carpeta /opt/si2 de cada PC del laboratorio).

El alumno deberá ejecutar (como root, en Ubuntu mediante el mandato `sudo`) simplemente el comando:

```
sudo sh /opt/si2/virtualip.sh <interfaz>
```

Donde *<interfaz>* es la interfaz de red sobre la que montaremos el bridge. Ejemplos: eth0, wlan0. Pongamos por caso que nuestra interfaz de red wifi es: wlan0. Entonces deberemos ejecutar

```
sudo sh /opt/si2/virtualip.sh wlan0
```

Sólo es necesario ejecutarlo una vez tras iniciar la red del propio PC.

Entrega

Como ya se ha indicado, no es necesario entregar nada en esta práctica. Sin embargo, se espera que el alumno instale las herramientas necesarias en su ordenador personal.

Bibliografía recomendada

Java EE

Your First Cup: An Introduction to the Java EE Platform:

http://www.oracle.com/pls/topic/lookup?ctx=javaee&id=JEEFC

Java EE Tutorial:

http://www.oracle.com/pls/topic/lookup?ctx=javaee&id=JEETT

Guidelines, Patterns, and Code for End-to-End Java Applications:

http://www.oracle.com/technetwork/java/namingconventions-139351.html

Glassfish

https://glassfish.java.net

Documentación*:*

https://glassfish.java.net/documentation.html

GlassFish Enterprise Server 4.0 Quick Start Guide:

https://glassfish.java.net/docs/4.0/quick-start-guide.pdf

GlassFish Enterprise Server 4.0 Administration Guide:

https://glassfish.java.net/docs/4.0/administration-guide.pdf

Ant

http://codefeed.com/tutorial/ant_intro.html

http://ant.apache.org/manual/