Despliegue de Aplicaciones Web

2ª Edición

Álvaro García Sánchez
Javier Sanz Rodríguez

Técnico Superior en **Desarrollo de Aplicaciones Web**

Despliegue de Aplicaciones Web

2ª Edición

Garceta
grupo editorial

Despliegue de Aplicaciones Web. 2ª Edición

Álvaro García Sánchez
Javier Sanz Rodríguez

ISBN: 978-84-1622-861-4
IBERGARCETA PUBLICACIONES, S.L., Madrid 2016

Edición: 2.ª
Impresión: 1.ª
N.º de páginas: 414
Formato: 20 x 26 cm

Reservados los derechos para todos los países de lengua española. De conformidad con lo dispuesto en el artículo 270 y siguientes del código penal vigente, podrán ser castigados con penas de multa y privación de libertad quienes reprodujeren o plagiaren, en todo o en parte, una obra literaria, artística o científica fijada en cualquier tipo de soporte sin la preceptiva autorización. Ninguna parte de esta publicación, incluido el diseño de la cubierta, puede ser reproducida, almacenada o trasmitida de ninguna forma, ni por ningún medio, sea éste electrónico, químico, mecánico, electro-óptico, grabación, fotocopia o cualquier otro, sin la previa autorización escrita por parte de la editorial.

Diríjase a CEDRO (Centro Español de Derechos Reprográficos), www.cedro.org, si necesita fotocopiar o escanear algún fragmento de esta obra.

COPYRIGHT © 2016 IBERGARCETA PUBLICACIONES, S.L.
info@garceta.es

Despliegue de Aplicaciones Web. 2ª Edición

© Álvaro García Sánchez, Javier Sanz Rodríguez

2.ª edición, 1.ª impresión
OI: 297/2016
ISBN: 978-84-1622-861-4
Deposito Legal: M-25863-2016
Imagen de cubierta: © Andrejs Pidjass - Fotolia

Impresión:Print House, marca registrada de Coplar, S.A.

IMPRESO EN ESPAÑA - PRINTED IN SPAIN

Nota sobre enlaces a páginas web ajenas: Este libro puede incluir referencias a sitios web gestionados por terceros y ajenos a IBERGARCETA PUBLICACIONES, S.L., que se incluyen sólo con finalidad informativa. IBERGARCETA PUBLICACIONES, S.L., no asume ningún tipo de responsabilidad por los daños y perjuicios derivados del uso de los datos personales que pueda hacer un tercero encargado del mantenimiento de las páginas web ajenas a IBERGARCETA PUBLICACIONES, S.L., y del funcionamiento, accesibilidad y mantenimiento de los sitios web no gestionados por IBERGARCETA PUBLICACIONES, S.L., directamente. Las referencias se proporcionan en el estado en que se encuentran en el momento de publicación sin garantías, expresas o implícitas, sobre la información que se proporcione en ellas.

PRÓLOGO

Este libro está diseñado para ser el libro de prácticas del módulo profesional ***Despliegue de Aplicaciones Web*** del segundo curso del ciclo formativo de grado superior DAW (**Diseño de Aplicaciones Web**). Debido a la amplitud de temas que abarca, también resulta de utilidad para los módulos profesionales ***Implantación de Aplicaciones Web*** del segundo curso del ciclo formativo de grado superior ASIR (**Administración de Sistemas Informáticos en Red**) y ***Desarrollo Web en Entorno Servidor*** del segundo curso del ciclo formativo de grado superior DAW (**Diseño de Aplicaciones Web**).

Partiendo de nuestra experiencia como profesores de ciclos formativos y profesores de Universidad, hemos desarrollado esta herramienta complementaria a las clases del módulo de formación profesional, que también será interesante para lectores independientes que quieran profundizar en las diferentes tecnologías implicadas en el ***Despliegue de Aplicaciones Web***.

Los objetivos principales que se alcanzarán utilizando el libro serán los siguientes:

- Conocer los diferentes tipos de arquitecturas web así como los principales servidores web y servidores de aplicaciones.
- Utilizar las herramientas necesarias para el despliegue y gestión de aplicaciones web.
- Configurar servicios DNS (*Domain Name System*) en función de los requerimientos de las aplicaciones web desplegadas.
- Administrar el servidor *Apache* realizando su instalación, configuración, despliegue y operación.
- Desplegar aplicaciones WAMP(*Windows*, *Apache*, *MySQL* y *PHP*) y LAMP(*Linux*, *Apache*, *MySQL* y *PHP*).
- Desarrollar y desplegar aplicaciones JavaEE basadas en *Servlets* y JSP (*Java Server Pages*).
- Administrar el servidor *Tomcat* realizando su instalación, configuración, despliegue y operación.
- Integrar los servidores *Apache* y *Tomcat*.
- Configurar servicios de directorio LDAP integrándolos con *Apache* y *Tomcat*.

Esta obra tiene una orientación fundamentalmente práctica. Consiste en una secuencia de prácticas resueltas que se desarrollan sobre diferentes máquinas virtuales con sistemas operativos variados, tanto libres como propietarios.

En la segunda edición de esta obra, se han actualizado las versiones de los sistemas operativos y servicios utilizados, se han ampliado las prácticas de desarrollo en JavaEE con el *framework Spring MVC* y mejorado las prácticas sobre despliegue de aplicaciones en la nube sobre el PaaS *OpenShift*.

La teoría de cada uno de los capítulos del libro se suministrará al profesor que adopte esta obra como libro de texto.

Índice general

1. Arquitecturas y tecnologías web **1**
1.1. Búsqueda de información sobre tecnologías web 2

2. Conceptos básicos de TCP/IP **3**
2.1. Conexión de las máquinas virtuales a la red del aula 4
2.2. Máquinas virtuales en la red IP del aula 5
2.3. Puertos y conexiones 10

3. Servicio de nombres de dominio (DNS) **13**
3.1. Funcionamiento del servicio DNS 14
3.2. Servidor DNS en *Microsoft Windows 2008 Server*. Instalación y configuración del servidor DNS como solo *cache* 15
3.3. Servidor DNS en *Microsoft Windows 2008 Server*. Configuración del servidor como primario (maestro) para una zona de resolución directa 17
3.4. Servidor DNS en *Microsoft Windows 2008 Server*. Configuración del servidor como primario (maestro) para una zona de resolución inversa 21
3.5. Servidor DNS en *Microsoft Windows 2012 Server*. Instalación y configuración del servidor DNS como solo *cache* 23
3.6. Servidor DNS en *Microsoft Windows 2012 Server*. Configuración del servidor como primario (maestro) para una zona de resolución directa 25
3.7. Servidor DNS en *Microsoft Windows 2012 Server*. Configuración del servidor como primario (maestro) para una zona de resolución inversa 29
3.8. Cliente DNS en las otras máquinas 31

4. WWW y protocolo HTTP **35**
4.1. Protocolo HTTP 36

5. Administración de Apache **39**
5.1. Instalación del servidor web *Apache 2.4* en *Linux* 40
5.2. Instalación del servidor web *Apache 2.2* en *Windows* 43
5.3. Ficheros de configuración y directivas en *Linux* 48
5.4. Ficheros de configuración y directivas en *Windows* 52
5.5. Configuración básica en *Linux* 56
5.6. Configuración básica en *Windows* 64
5.7. Módulos en *Linux* 69
5.8. Módulos en *Windows* 71
5.9. Control de acceso por IP y nombre de dominio 74
5.10. Autenticación y autorización *Basic* y *Digest* 75
5.11. Ficheros .htaccess 77
5.12. Ficheros de registros (*logs*) 80
5.13. Módulos *mod_status* y *mod_info* 80
5.14. *Webalizer* 83
5.15. Alojamiento virtual de sitios web en Internet 85
5.16. Alojamiento virtual de sitios web en *Linux* 87
5.17. Alojamiento virtual de sitios web en *Windows* 93
5.18. HTTPS y certificados digitales 97

5.19. Servidor virtual HTTPS por defecto en *Linux* 101
5.20. Servidor virtual HTTPS en *Linux* 105
5.21. Servidor virtual HTTPS por defecto en *Windows* 111
5.22. WebDav 116

6. Despliegue en entornos WAMP-LAMP **121**
6.1. Instalación de un servidor WAMP 122
6.2. Instalación de un servidor LAMP 131
6.3. Despliegue en un servidor local 136
6.4. Servidor *vsftpd* en *Linux* 141
6.5. Servidor *OpenSSH* en *Linux* 145
6.6. Despliegue de aplicaciones en un servidor remoto 147
6.7. Despliegue en un servidor de *hosting* 152
6.8. Despliegue en la nube (OpenShift) - 1 162
6.9. Despliegue en la nube (OpenShift) - 2 172

7. Aplicaciones JavaEE **181**
7.1. Instalación de *Tomcat 7.0* en *Windows* 182
7.2. Instalación y configuración básica de *Eclipse* 188
7.3. Integración de *Eclipse* con *Tomcat* 190
7.4. Despliegue de *Servlets* 198
7.5. *Servlet* con acceso a base de datos con *JDBC* 205
7.6. Sesiones con *Servlets* 215
7.7. Conceptos adicionales sobre *Servlets* 220
7.8. *Pool de conexiones con JDNI y Servlets 3.0* 229
7.9. *Java Server Pages* 237
7.10. *Arquitectura MVC* 244
7.11. Instalación de *Spring Framework* en *Windows* 256
7.12. *Spring* MVC 261

8. Administración de Tomcat **267**
8.1. Instalación de *Apache Tomcat 7.0* en *Linux* 268
8.2. Arquitectura y ficheros de configuración 273
8.3. Despliegue manual de aplicaciones web 277
8.4. Despliegue de aplicaciones en el *ROOT Context* 286
8.5. Instalación de *Tomcat Web Manager* 287
8.6. Administración con *Tomcat Web Manager* 290
8.7. Despliegue de *OpenCms* 296
8.8. Despliegue de aplicaciones con *Apache Ant* 301
8.9. Sesiones 308
8.10. Autenticación y autorización (*MemoryRealm*) 312
8.11. Autenticación y autorización (*UserDatabaseRealm*) 315
8.12. Autenticación y autorización (*JDBCRealm*) 316
8.13. *Valves* 322
8.14. Filtros 326
8.15. Ficheros de registros (*logs*) 328
8.16. Alojamiento virtual 329
8.17. *Tomcat Host Manager* 335
8.18. Seguridad SSL 337
8.19. Despliegue en la nube (OpenShift) - 3 341

8.20. Despliegue en la nube (OpenShift) - 4 345

9. Integración de Apache y Tomcat **349**
9.1. Integración de *Apache* y *Tomcat* usando mod_proxy 350
9.2. Integración de *Apache* y *Tomcat* usando mod_jk 357
9.3. Cluster con *Apache* y *Tomcat* (balanceo de carga) 360

10. Servicios de directorio. LDAP **365**
10.1. Instalación de *OpenLDAP 2.4* en *Linux* 366
10.2. Operaciones sobre LDAP 369
10.3. Clientes LDAP 371
10.4. Configuración del servidor LDAP 382
10.5. Autenticación y autorización LDAP en *Apache* 384
10.6. Autenticación y autorización LDAP en *Tomcat* (*JNDIRealm*) 387

11. Administración de GlassFish **391**
11.1. Instalación de *GlassFish 4.0* en *Windows* 392
11.2. Herramientas de administración de*GlassFish* 398
11.3. Despliegue de aplicaciones web en *GlassFish* 401

Capítulo 1

Arquitecturas y tecnologías web

1.1. Búsqueda de información sobre tecnologías web

Busca información sobre tecnologías utilizadas en el desarrollo de aplicaciones web y describe brevemente las tecnologías más utilizadas.

- Clientes (navegadores).
- Servidores
  - Servidores webs.
  - Servidores de aplicaciones.
  - Sistemas gestores de bases de datos.
- Desarrollo/Lenguajes
  - Cliente.
  - Servidor.
  - *Frameworks.*

Capítulo 2

Conceptos básicos de TCP/IP

2.1. Conexión de las máquinas virtuales a la red del aula

Para realizar las prácticas del curso utilizaremos tres máquinas virtuales creadas con el software de virtualización *VirtualBox*. Las máquinas tienen instalados los sistemas operativos: *Windows7*, *Ubuntu Server 14.04* y *Windows 2008 Server* o *Windows 2012 Server*. Puedes elegir utilizar Windows 2008 o 2012 (ten en cuenta que la versión 2012 necesita mayores recursos). Esta máquina se utilizará principalmente como servidor DNS y la configuración es muy similar en las dos versiones.

El nombre de las máquinas es:

- **DesarrolloW7**
- **ServidorLinux**
- **ServidorW2008** o **ServidorW2012**

El objetivo de la práctica es conectar las máquinas proporcionadas a la red de área local del aula, tal y como se muestra en la Figura 2.1.

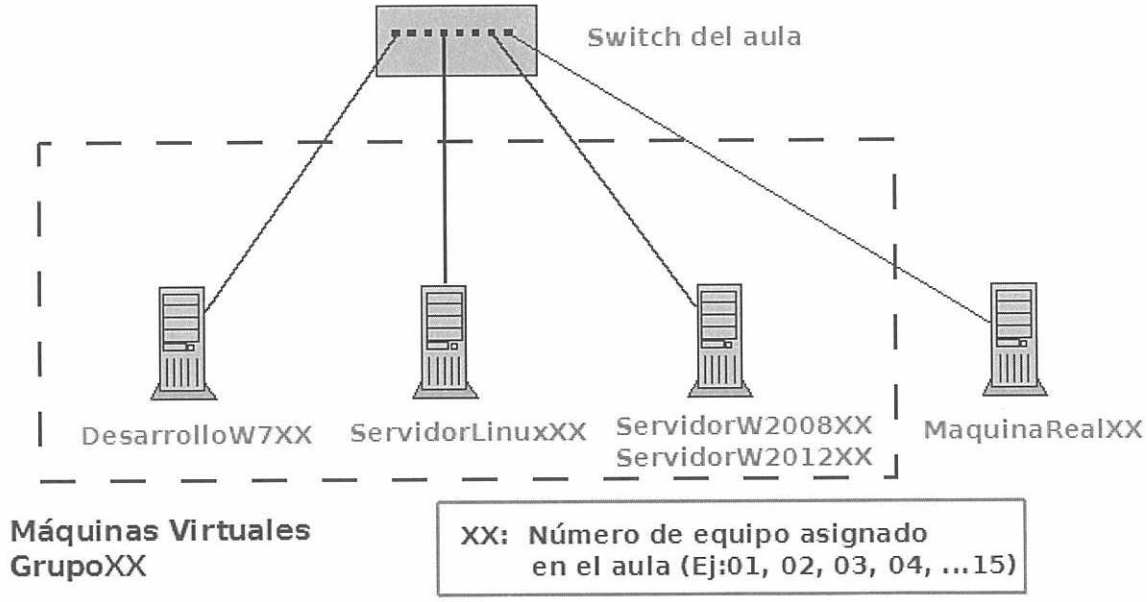


Figura 2.1: Máquinas virtuales conectadas a la red de aula

1. Inicia *VirtualBox*.
2. Agrega las tres máquinas virtuales. En el menú de *VirtualBox* accede a **Máquina, Agregar** y selecciona la máquina correspondiente.
3. Una vez que tengas las máquinas agregadas conéctalas a la red del aula. En cada una de ellas:
   3.1. Pulsa el botón derecho del ratón y selecciona **Configuración**.
   3.2. Accede a **Red** y configura el **Adaptador1** conectado a: **Adaptador puente**.
   3.3. Pincha en **Aceptar** para aplicar los cambios.
4. Cambia la dirección MAC de las tarjetas de red para evitar direcciones MAC duplicadas en la red del aula, Figura 2.2.

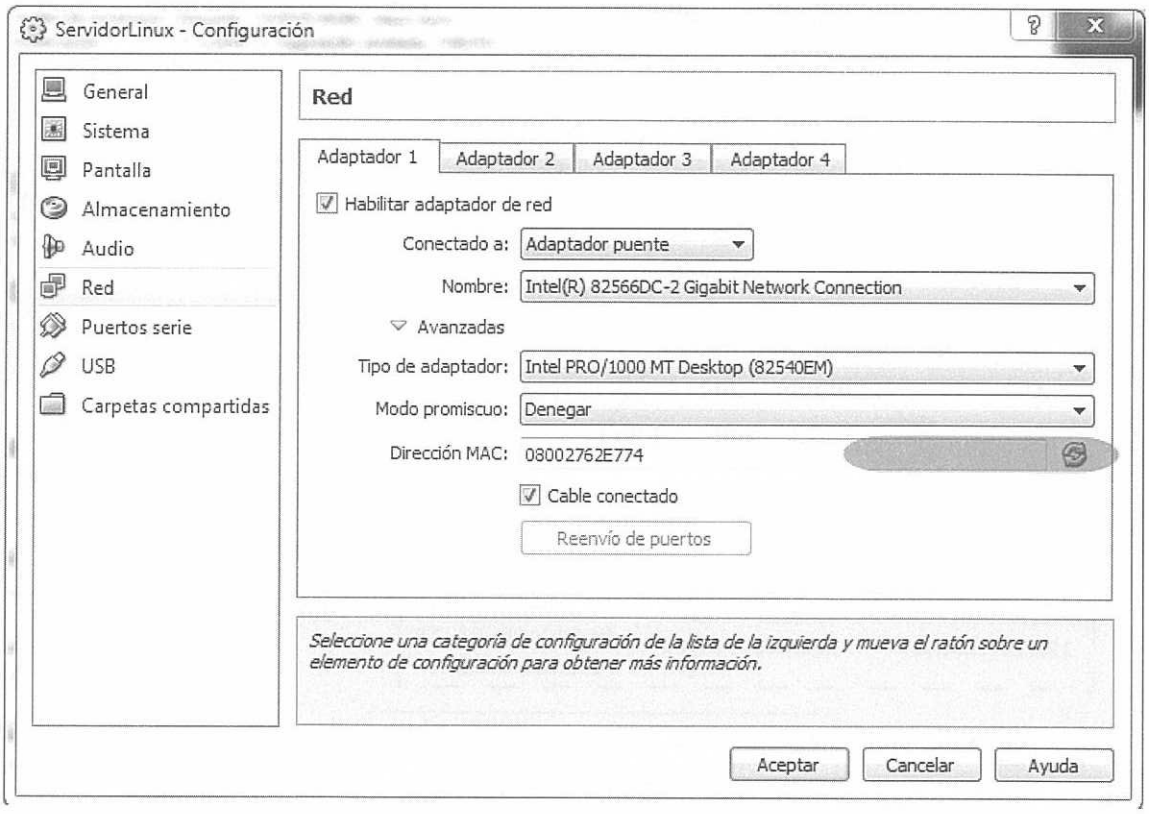


Figura 2.2: Cambiar la dirección MAC

2.2. Máquinas virtuales en la red IP del aula

Configura las máquinas virtuales para que formen parte de la red IP del aula y cambia el nombre de cada una de ellas para diferenciarlas de las máquinas de los compañeros, Figura 2.3.

Tendrás que adaptar las direcciones IP y la máscara de red a la red IP del aula donde realices las prácticas. Como servidor DNS puedes utilizar el del aula o alguno público de Internet como por ejemplo 8.8.8.8.

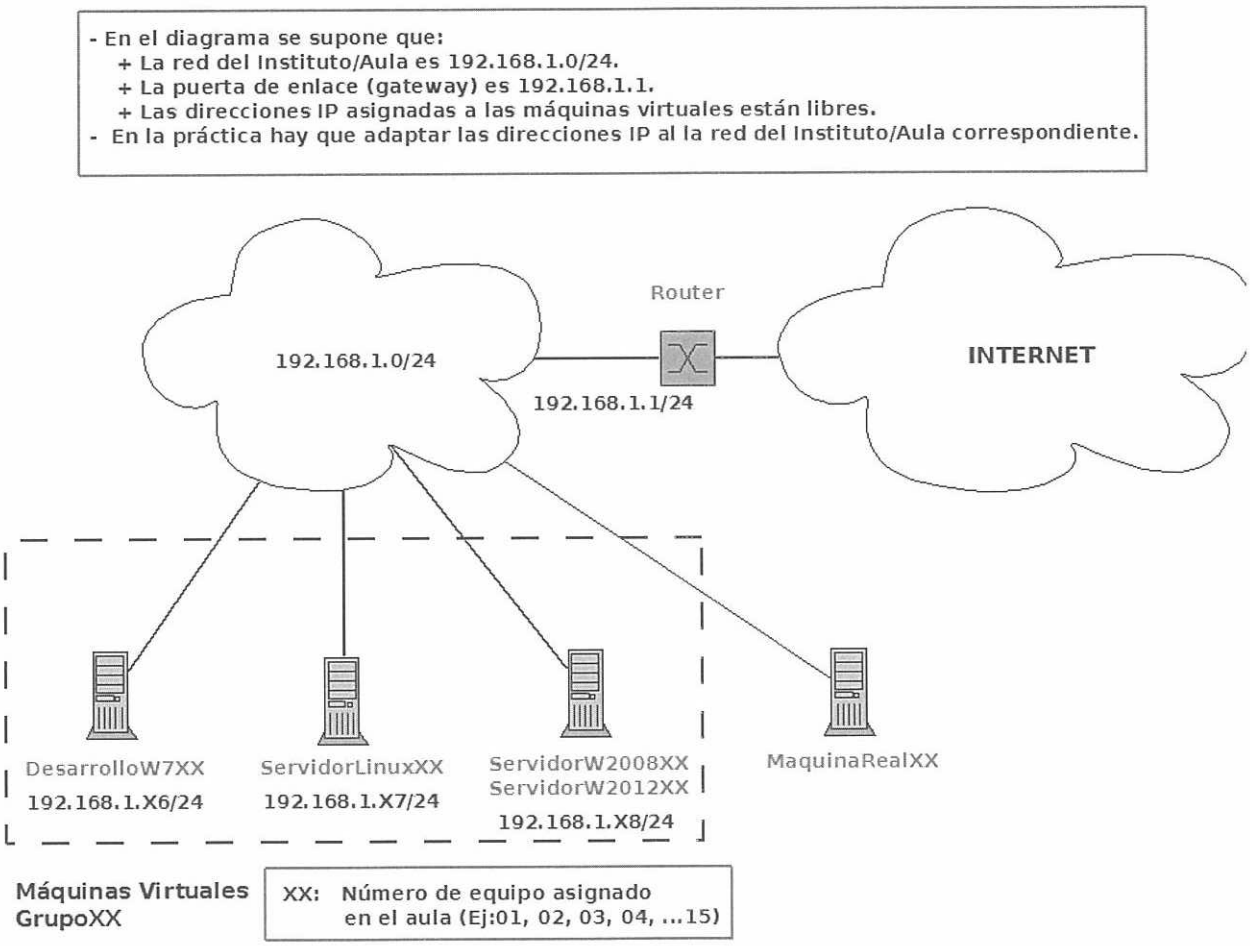


Figura 2.3: Máquinas virtuales conectadas a la red IP del aula

1. **Configuración de la máquina DesarrolloW7XX**

   1.1. Inicia sesión en **DesarrolloW7XX** con un usuario con privilegios de administrador.

   1.2. Accede a las propiedades del protocolo de Internet versión 4, **TCP/IPv4** (**Menú Inicio, Panel de control, Redes e Internet, Centro de redes y recursos compartidos, Conexión de área local, Propiedades, Protocolo de Internet versión** 4) y añade la configuración elegida en **Dirección IP**, **Máscara de subred**, **Puerta de enlace** y **Servidor DNS preferido**, véase Figura 2.4.

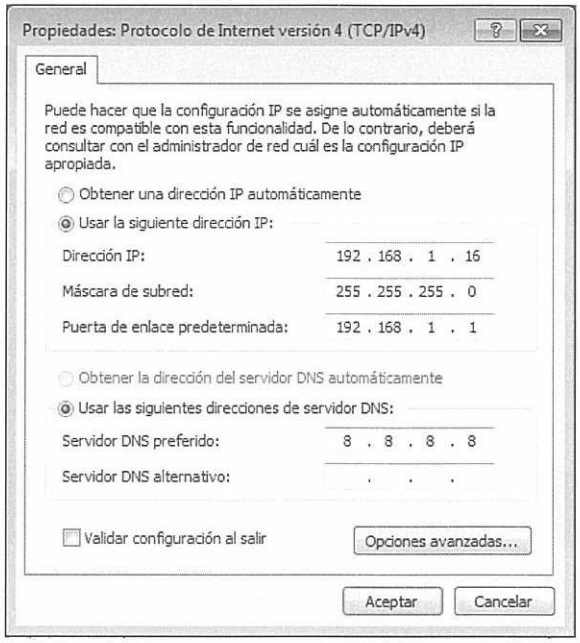


Figura 2.4: Configuración TCP/IP de la máquina DesarrolloW7XX

   1.3. Abre un terminal (**Inicio, Todos los Programas, Accesorios, Símbolo de Sistema**) y ejecuta el comando `ipconfig` para verificar la configuración.

   1.4. Configura el nombre del equipo accediendo a la ventana "**Cambios en el dominio o nombre del equipo**" (**Menú Inicio, Panel de control, Sistema y Seguridad, Sistema, Cambiar configuración, Cambiar...**). Asigna como nombre **DesarrolloW7XX** y como grupo de trabajo DESPLIEGUEXX.

   1.5. Reinicia el sistema para activar la nueva configuración.

2. **Configuración de la máquina ServidorLinuxXX**

   2.1. Inicia sesión en **ServidorLinuxXX** con un usuario con privilegios de administrador.

   2.2. Averigua el nombre que el sistema ha asignado a las interfaces de red ejecutando el comando `ifconfig -a`. El resultado obtenido deberá mostrar la interfaz **lo** de bucle local y otra interfaz con el formato **ethx** (eth0, eth1,... para una red cableada).

   2.3. Edita el fichero de configuración **/etc/network/interfaces**. Modifica el fichero tal como aparece en la Figura 2.5, sustituyendo **eth0** por el nombre de tu interfaz de red cableada.

```
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto eth0
iface eth0 inet static
        address 192.168.1.17
        netmask 255.255.255.0
        gateway 192.168.1.1
        dns-nameservers 8.8.8.8
```

Figura 2.5: **/etc/network/interfaces**

2.4. Activa y desactiva la configuración sobre eth0 para aplicar los cambios.

```
sudo ifdown eth0
sudo ifup eth0
```

2.5. Ejecuta el comando `ifconfig` para verificar la configuración.

2.6. Consulta el fichero de configuración **/etc/resolv.conf** y observa que se utiliza el servidor DNS que has definido previamente, Figura 2.6.

```
# Dynamic resolv.conf(5) file for glibc resolver(3) generated by resolvconf(8)
#     DO NOT EDIT THIS FILE BY HAND -- YOUR CHANGES WILL BE OVERWRITTEN
nameserver 8.8.8.8
```

Figura 2.6: **/etc/resolv.conf**

2.7. Edita el fichero de configuración **/etc/hostname** y escribe en el fichero únicamente el nombre que deseamos que tenga la máquina (**ServidorLinuxXX**), véase Figura 2.7.

```
ServidorLinux01
```

Figura 2.7: **/etc/hostname**

2.8. Edita el fichero **/etc/hosts** y asocia el nombre **ServidorLinuxXX** con la dirección IP de bucle interno (127.0.1.1) como se indica en la Figura 2.8.

2.9. Reinicia la máquina con el comando `sudo reboot`.

3. **Configuración de la máquina ServidorW2008XX**

3.1. Inicia sesión en **ServidorW2008XX** con un usuario con privilegios de administrador.

3.2. Accede a las propiedades del protocolo de Internet versión 4, **TCP/IPv4** (**Menú Inicio, Panel de control, Redes e Internet, Centro de redes y recursos compartidos, Ver Estado (Conexión de área local), Propiedades, Protocolo de Internet versión 4**) y añade la configuración elegida en **Dirección IP**, **Máscara de subred**, **Puerta de enlace** y **Servidor DNS preferido**.

```
127.0.0.1       localhost
127.0.1.1       ServidorLinux01_

# 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
```

Figura 2.8: **/etc/hosts**

3.3. Abre un terminal (**Inicio, Todos los Programas, Accesorios, Símbolo de Sistema**) y ejecuta el comando `ipconfig` para verificar la configuración.

3.4. Configura el nombre del equipo accediendo a la ventana "**Cambios en el dominio o nombre del equipo**" (**Menú Inicio, Panel de control, Sistema y Mantenimiento, Sistema, Cambiar configuración, Cambiar...**). Asigna como nombre **ServidorW2008XX** y como grupo de trabajo DESPLIEGUEXX.

3.5. Reinicia el sistema para activar la nueva configuración.

4. <u>**Configuración de la máquina ServidorW2012XX**</u>

4.1. Inicia sesión en **ServidorW2012XX** con un usuario con privilegios de administrador.

4.2. Accede a las propiedades del protocolo de Internet versión 4, **TCP/IPv4** (**Configuración, Panel de control, Redes e Internet, Centro de redes y recursos compartidos, Ver Estado (Conexión de área local), Propiedades, Protocolo de Internet versión 4**) y añade la configuración elegida en **Dirección IP**, **Máscara de subred**, **Puerta de enlace** y **Servidor DNS preferido**.

4.3. Abre un terminal (**Buscar, Símbolo de sistema**) y ejecuta el comando `ipconfig` para verificar la configuración.

4.4. Configura el nombre del equipo accediendo a la ventana "**Cambios en el dominio o nombre del equipo**" (**Configuración, Panel de control, Sistema y Mantenimiento, Sistema, Cambiar configuración, Cambiar...**). Asigna como nombre **ServidorW2008XX** y como grupo de trabajo DESPLIEGUEXX.

4.5. Reinicia el sistema para activar la nueva configuración.

5. <u>**Comprobar la configuración**</u>

5.1. Inicia sesión en **DesarrolloW7XX**.

5.2. Utiliza el comando `ping` para comprobar que existe comunicación con las otras máquinas, con la puerta de enlace y con Internet.

- `ping 192.168.1.X7`
- `ping 192.168.1.X8` ¿Qué ocurre?
- `ping 192.168.1.254`
- `ping 8.8.8.8`

5.3. Inicia sesión en **ServidorLinuxXX**.

5.4. Utiliza el comando `ping` para comprobar que existe comunicación con las otras máquinas, con la puerta de enlace y con Internet.

- `ping 192.168.1.X6` ¿Qué ocurre?
- `ping 192.168.1.X8` ¿Qué ocurre?
- `ping 192.168.1.254`
- `ping 8.8.8.8`

5.5. Inicia sesión en **ServidorW2008XX** o en **ServidorW2008XX**.

5.6. Utiliza el comando `ping` para comprobar que existe comunicación con las otras máquinas, con la puerta de enlace y con Internet.

- `ping 192.168.1.X6` ¿Qué ocurre?
- `ping 192.168.1.X7`
- `ping 192.168.1.254`
- `ping 8.8.8.8`

6. **Habilitar respuesta a *ping* en el *Firewall* de *Windows***

Los sistemas operativos *Linux* tienen habilitada por defecto la respuesta a *pings*, pero en *Windows 7* y *Windows 2008* el *firewall* iniciado por defecto no lo permite.

6.1. Inicia sesión en **DesarrolloW7XX** con un usuario con privilegios de administrador.

6.2. Accede a la ventana de *Firewall* de *Windows* con seguridad avanzada (**Menú Inicio, Panel de control, Sistema y seguridad, Firewall de Windows, Configuración avanzada**). Habilita, dentro de la sección **Reglas de entrada**, la regla **Archivos e impresoras compartidos (petición eco: ICMPv4 de entrada)**.

6.3. Inicia sesión en **ServidorW2008XX** o **ServidorW2012XX**con un usuario con privilegios de administrador.

6.4. Accede a la ventana de *Firewall* de *Windows* con seguridad avanzada (**Menú Inicio, Herramientas administrativas, Firewall de Windows con seguridad avanzada**). Habilita, dentro de la sección **Reglas de entrada**, la regla **Archivos e impresoras compartidos (petición eco: ICMPv4 de entrada)**.

6.5. Ahora sí, deberían responder con normalidad.

2.3. Puertos y conexiones

Averigua los puertos TCP y UDP a la escucha de cada una de las máquinas virtuales y comprueba las conexiones TCP que se establecen al conectarse a un servidor web de Internet.

1. Inicia sesión en **DesarrolloW7XX** con un usuario con privilegios de administrador.

1.1. Averigua los puertos TCP a la escucha con el comando `netstat -a -p TCP -n`.

1.2. Averigua los puertos UDP a la escucha con el comando `netstat -a -p UDP -n`.

1.3. Abre el navegador y accede a una web de Internet.

a. Muestra la conexiones TCP establecidas con el comando `netstat -p TCP -n`

b. ¿Qué puerto/s ha asignado el sistema operativo al navegador web para establecer la conexión/es TCP?

c. ¿Qué puerto/s utiliza/n el servidor/es con los que se establecen las conexiones?

2. Inicia sesión en **ServidorLinuxXX** con un usuario con privilegios de administrador.

   2.1. Averigua los puertos TCP a la escucha con el comando `netstat -ltn`.

   2.2. Averigua los puertos UDP a la escucha con el comando `netstat -lun`.

3. Inicia sesión en **ServidorW2008XX** o en **ServidorW2012XX** con un usuario con privilegios de administrador.

   3.1. Averigua los puertos TCP a la escucha con el comando `netstat -a -p TCP -n`.

   3.2. Averigua los puertos UDP a la escucha con el comando `netstat -a -p UDP -n`.

Capítulo 3

Servicio de nombres de dominio (DNS)

3.1. Funcionamiento del servicio DNS

En esta práctica se probará el funcionamiento del servicio DNS. Para ello utilizaremos las herramientas `nslookup` y `dig` que permiten realizar consultas personalizadas a servidores DNS. Estas herramientas se utilizan para comprobar el funcionamiento del servicio, obtener información y verificar el funcionamiento de los servidores.

1. Inicia sesión en **DesarrolloWindowsXX**.

   1.1. Utiliza el comando nslookup para obtener la/s direcciones IP asociadas al nombre DNS `www.madrid.org`.

   ```
   nslookup www.madrid.org
   ```

   - Observa que el servidor DNS que responde es el que está configurado en las propiedades TCP/IP del equipo.
   - Observa que existen varias IPs asociadas al nombre de dominio.
   - Observa los nombres de dominio que son equivalentes (alias).
   - Observa que la respuesta no es autorizada.

   1.2. Utiliza el comando `nslookup` para obtener el/los nombres de dominio asociados a la dirección IP `130.206.13.20`.

   ```
   nslookup 130.206.13.20
   ```

   1.3. Utiliza el comando `nslookup` para obtener la/s direcciones IP asociadas al nombre DNS `www.madrid.org` preguntando al servidor DNS `8.8.4.4`

   ```
   nslookup www.madrid.org 8.8.4.4
   ```

   - Observa que la respuesta no es autorizada.

   1.4. Utiliza el comando `nslookup` para obtener la/s direcciones IP asociadas al nombre DNS `www.madrid.org` preguntando al servidor DNS `olimpia.madrid.org`.

   ```
   nslookup www.madrid.org olimpia.madrid.org
   ```

   - Observa que la respuesta sí es autorizada.

2. Inicia sesión en **ServidorLinuxXX**.

   2.1. Utiliza el comando `nslookup` para obtener la/s direcciones IP asociadas al nombre DNS `www.google.es`.

   ```
   nslookup www.google.es
   ```

   2.2. Utiliza el comando `dig` para obtener la/s direcciones IP asociadas al nombre DNS `www.google.es`.

   ```
   dig www.google.es
   ```

   2.3. Utiliza el comando `dig` para obtener el/los nombres de dominio asociados a la dirección IP `130.206.13.20`.

   ```
   dig -x  130.206.13.20
   ```

   2.4. Utiliza el comando `dig` para obtener la/s direcciones IP asociadas al nombre DNS `www.madrid.org` preguntando al servidor DNS `8.8.4.4`

```
dig @8.8.4.4  www.google.es
```

2.5. Utiliza el comando `dog` para obtener la/s direcciones IP asociadas al nombre DNS `www.google.es` preguntando al servidor DNS `ns1.google.com`

```
dig @ns1.google.com www.google.es
```

3.2. Servidor DNS en *Microsoft Windows 2008 Server*. Instalación y configuración del servidor DNS como solo *cache*

Instala y configura un servidor DNS en **ServidorW2008XX** para que actúe como solo *cache* y responda a consultas recursivas.

1. <u>**Instalación**</u>

   1.1. Inicia una sesión en **ServidorW2008XX** con el usuario **administrador**.

   1.2. En el menú de **Inicio** selecciona **Administrador del Servidor**.

   1.3. En el árbol de la izquierda selecciona **Funciones**. En la ventana de la derecha haz clic en **Agregar Funciones**.

   1.4. Lee la información que muestra el Asistente de agregar funciones y haz clic en **Siguiente**.

   1.5. Selecciona la función **Servidor DNS** y haz clic en **Siguiente**.

   1.6. Lee la información que se ofrece sobre el servidor y sobre aspectos que hay que tener en cuenta y haz clic en **Siguiente**.

   1.7. Confirma las selecciones de instalación haciendo clic en **Instalar**. Comenzará el proceso de instalación.

   1.8. Cuando finalice el proceso de instalación cierra el asistente.

   1.9. Observa que en el resumen de funciones aparece instalado el servidor DNS y que en el árbol de la izquierda, en **funciones**, aparece un enlace al **Servidor DNS**. Si pinchas en él se muestra información sobre eventos, servicios y recursos y soporte técnico del servidor. Observa que el servidor DNS está iniciado, véase Figura 3.1.

   1.10. Abre un terminal, ejecuta el comando `netstat -a -n` y comprueba que el servidor está a la escucha en los puertos 53 TCP y UDP.

   1.11. En el menú de **Inicio, Herramientas Administrativas** se ha creado la entrada **DNS** que permite acceder a la consola de administración del servidor. Accede a ella, Figura 3.2.

   1.12. Puedes comprobar accediendo al *Firewall* de *Windows* que se ha creado una excepción para el servidor DNS.

2. <u>**Configuración del servidor como solo *cache***</u>

   Por defecto el servidor está configurado como solo *cache* (no es autorizado ninguna zona) que responde a consultas recursivas.

   2.1. Comprueba que el servidor resuelve nombres de dominio de Internet configurando el cliente DNS para que utilice el servidor DNS instalado en la máquina local (127.0.0.1), véase Figura 3.3.

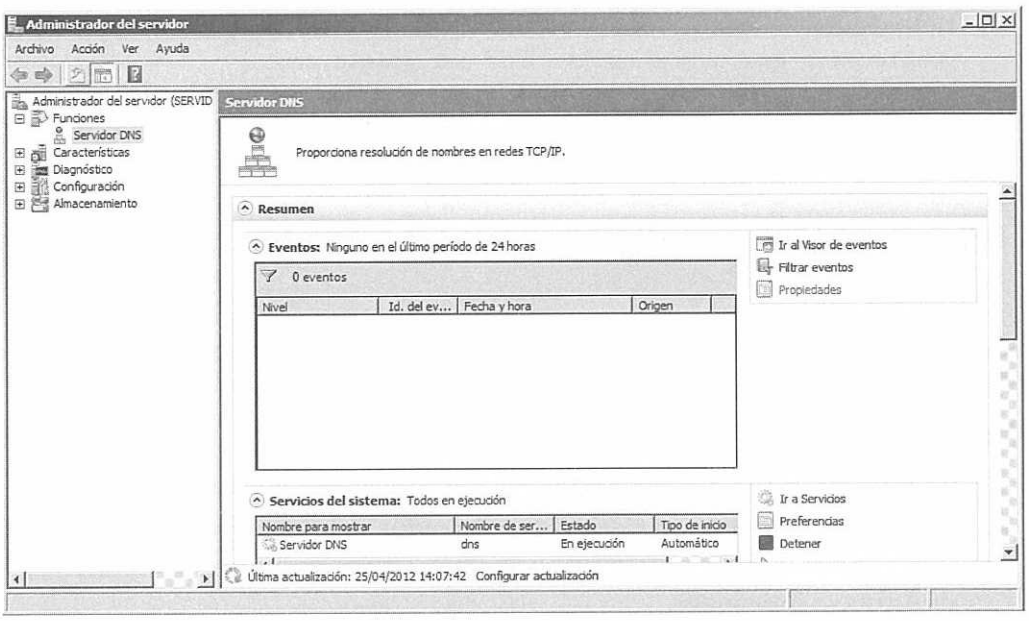


Figura 3.1: Servidor DNS instalado en iniciado

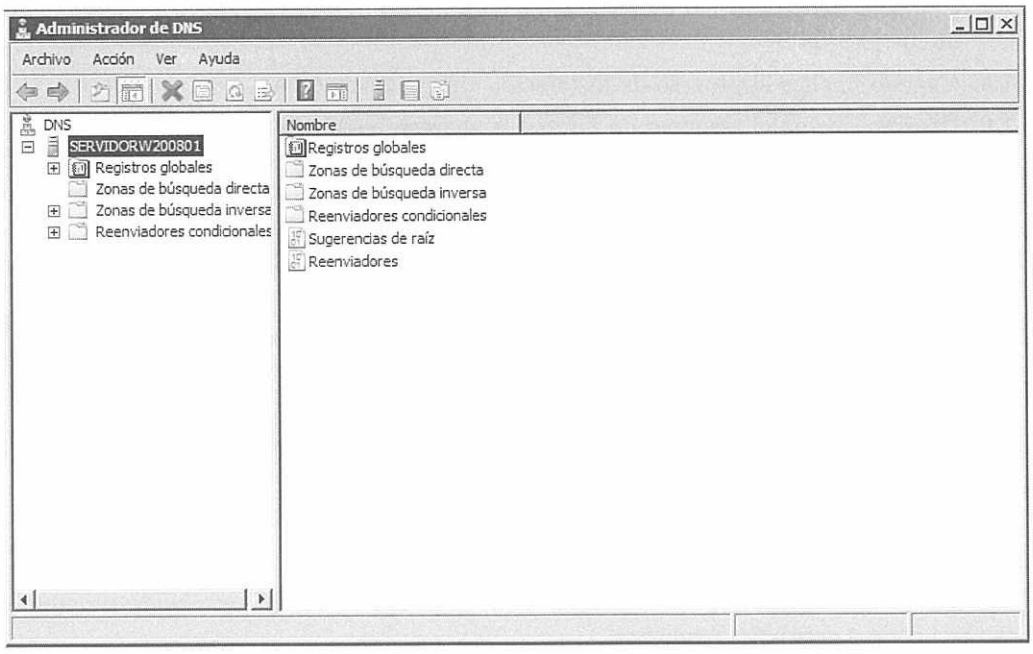


Figura 3.2: Consola de administración del servidor DNS

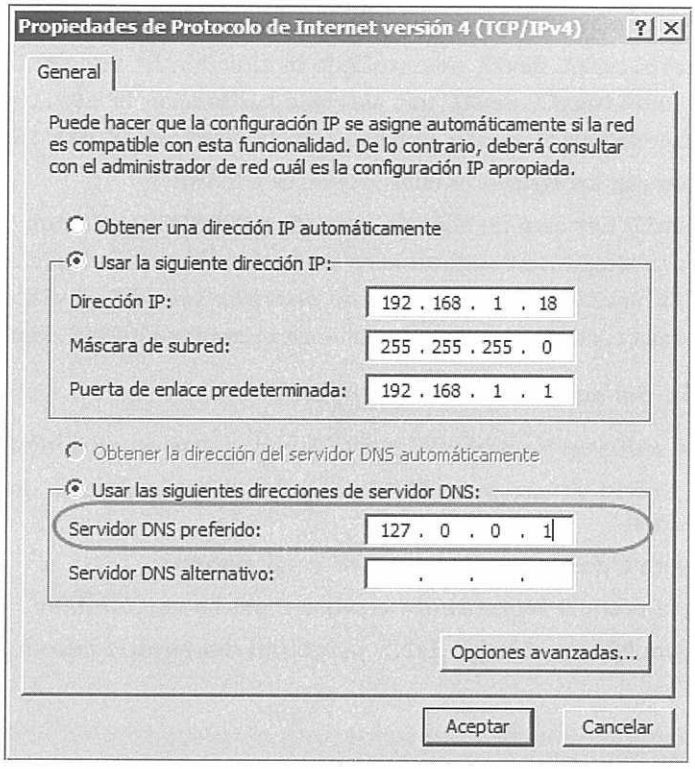


Figura 3.3: Configuración TCP/IP de la máquina ServidorW2008XX

2.2. Usa el comando `nslookup` para resolver un nombre, por ejemplo `www.madrid.org`.

2.3. Accede a la consola de administración del servidor DNS.

2.4. Activa la vista avanzada en el menú **Ver, Avanzada**.

2.5. En la parte izquierda de la ventana aparece una entrada que permite consultar la *cache* del servidor. Busca el nombre de dominio por el que has preguntado y observa que, como consecuencia de la resolución de consultas recursivas, hay más información.

2.6. Desactiva la vista avanzada.

3.3. Servidor DNS en *Microsoft Windows 2008 Server*. Configuración del servidor como primario (maestro) para una zona de resolución directa

Configura el servidor DNS del equipo **ServidorW2008XX**:

- El servidor solo servirá a la red local (no sirve a equipos de Internet).
- Actuará como maestro y tendrá autoridad sobre el dominio `dawXX.net`.
  - No se permitirán actualizaciones dinámicas.
  - El servidor DNS maestro del dominio será `servidorw2008XX.dawXX.net` (es decir, el equipo donde está instalado en servidor DNS) (registro NS).

- Se configurarán los siguientes nombres nombres de dominio (registros A).
  - `desarrollw7XX.dawXX.net` asociado la dirección IP `192.168.1.X6`
  - `servidorlinuxXX.dawXX.net` asociado la dirección IP `192.168.1.X7`
  - `servidorw2008XX.dawXX.net` asociado la dirección IP `192.168.1.X8`
- Se configurarán los siguientes alias (registros CNAME).
  - `ns.dawXX.net` será un alias de `servidorw2008XX.dawXX.net`
  - `asterix.dawXX.net` será un alias de `desarrollow7XX.dawXX.net`
  - `obelix.dawXX.net` será un alias de `servidorlinuxXX.dawXX.net`
  - `panoramix.dawXX.net` será un alias de `servidorw2008XX.dawXX.net`

1. **Configuración del sufijo DNS del equipo**

   1.1. Inicia una sesión en **ServidorW2008XX** con el usuario **administrador**.

   1.2. Accede a **Inicio** sitúate sobre **Equipo**, pulsa el botón derecho del ratón y selecciona **Propiedades**.

   1.3. Pincha sobre **Configuración avanzada del sistema**.

   1.4. Accede a la pestaña **Nombre de equipo** y pincha en **Cambiar**.

   1.5. Pincha sobre **Más** y en **sufijo DNS principal del equipo** introduce `dawXX.net`, véase Figura 3.4.

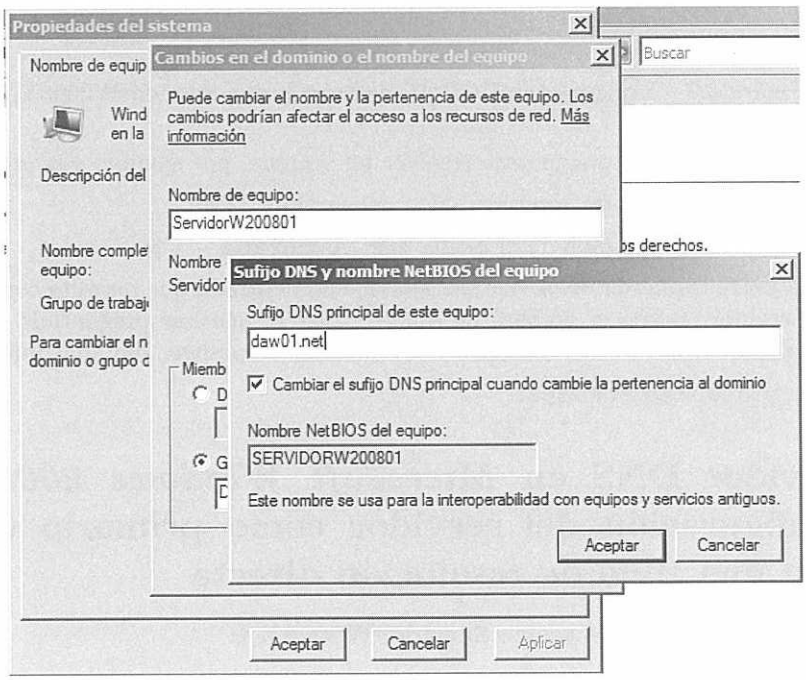


Figura 3.4: Sufijo DNS de la máquina **ServidorWindows2008XX**

   1.6. Acepta los cambios y reinicia el equipo.

2. **Configuración de la zona de resolución directa**

   2.1. Inicia una sesión en **ServidorW2008XX** con el usuario **administrador**.

   2.2. Accede a la consola de administración del servidor DNS (**Inicio, Herramientas Administrativas, DNS**).

   2.3. Sobre **Zonas de búsqueda directa** haz clic con el botón derecho del ratón y selecciona **Zona nueva**.

   2.4. Lee la información que ofrece el asistente y haz clic en **Siguiente** para continuar.

   2.5. Selecciona **Zona principal** y haz clic en **Siguiente**.

   2.6. Introduce `dawXX.net` como nombre de la zona y haz clic en **Siguiente**.

   2.7. Deja seleccionada la opción **Crear un archivo nuevo con este nombre** y mantén el nombre que sugiere el asistente. Pincha en **Siguiente**.

   2.8. Selecciona la opción **No admitir actualizaciones dinámicas** y pincha en **Siguiente**.

   2.9. Lee el resumen y pincha **Finalizar**.

   2.10. Observa que se ha creado una entrada en "**Zonas de búsqueda directa**" con el nombre de la zona (`dawXX.net`).

   2.11. Pincha sobre el nombre de la zona y observa en la zona central de la consola los registros de recursos que se han creado automáticamente. Se han añadido a la zona el registro **SOA** y un registro **NS** indicando que el servidor DNS para el dominio es este equipo (**servidorW2008XX**).

   2.12. Haz doble clic sobre el registro SOA y observa sus propiedades.

   2.13. Haz doble clic sobre el registro NS y observa sus propiedades. Fíjate que en el campo dirección IP del nombre de dominio del equipo aparece desconocido. Esto es porque todavía no hemos creado un registro A para el nombre `servidorw2008XX.dawXX.net`.

   2.14. Crea los registros A para los nombres de los equipos de la red virtual.

      a. Sobre la zona `dawXX.net` haz clic con el botón derecho del ratón y selecciona **Host nuevo (A o AAAA)**.

      b. Introduce el nombre y la IP asociada.

      c. Si estuviese creada una zona de resolución inversa podrías marcar la opción **Crear registro del puntero (PTR) asociado**.

   2.15. Crea los registros CNAME para los alias.

      a. Sobre la zona `dawXX.net` haz clic con el botón derecho del ratón y selecciona **Alias nuevo (CNAME)**.

      b. Introduce el nombre del alias y el nombre del dominio asociado. Puedes pinchar en **Examinar** para seleccionar los registros A que creaste previamente.

      Véase la Figura 3.5 con la zona configurada.

3. **Comprobar la configuración**

   3.1. Usa el comando `nslookup` para comprobar que el servidor DNS resuelve consultas directas sobre los nombres de la zona `dawXX.net` (recuerda que las prácticas anteriores configuraste la máquina **ServidorW2008XX** para que usase como servidor DNS el 127.0.0.1), véase Figura 3.6.

   3.2. Comprueba que si usas un nombre DNS sin puntos, se completa con el sufijo DNS configurado en el equipo, véase Figura 3.7.

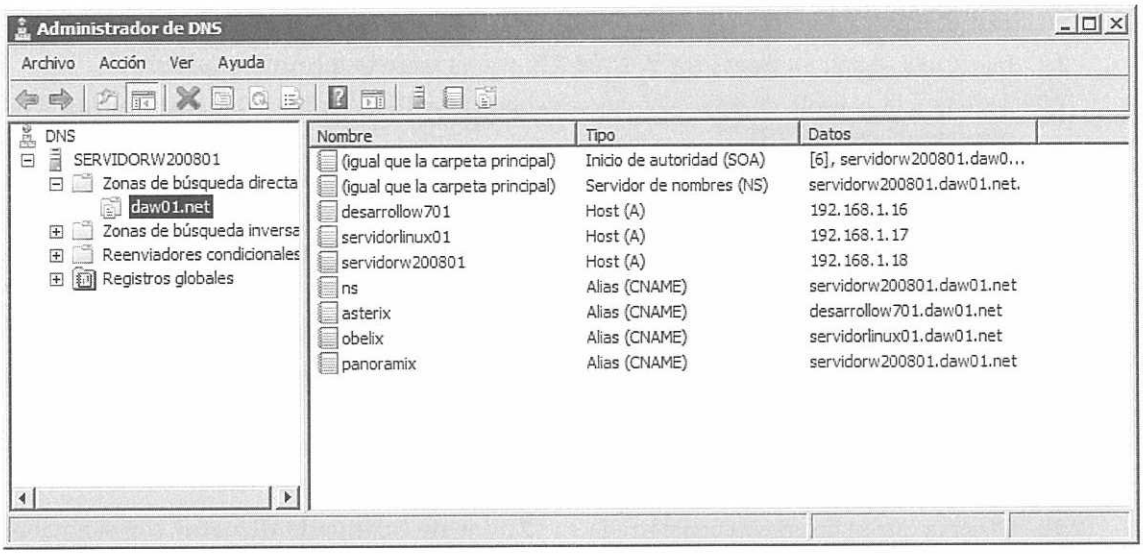


Figura 3.5: Zona primaria de resolución directa

Figura 3.6: Resolución directa

Figura 3.7: Resolución directa, se completa con el súfijo DNS del equipo

3.3. Usa *nslookup* para enviar una consulta de resolución inversa al servidor sobre algunas de las direcciones IP usadas en los registro A de la zona de resolución inversa. Observa que no resuelve los nombres.

3.4. Servidor DNS en *Microsoft Windows 2008 Server*. Configuración del servidor como primario (maestro) para una zona de resolución inversa

Configura el servidor DNS del equipo **ServidorW2008XX**:

- El servidor solo servirá a la red local (no sirve a equipos de Internet).
- Actuará como maestro y tendrá autoridad sobre la zona de resolución inversa de la red 192.168.1.0/24.
  - No se permitirán actualizaciones dinámicas.
  - El servidor DNS maestro del dominio será `servidorw2008XX.dawXX.net` (es decir, el equipo donde está instalado en servidor DNS) (registro NS).
  - Las direcciones IP de los equipos se corresponderán las representadas en el esquema de la red virtual(registros PTR).

1. **Configuración de la zona de resolución inversa**

   1.1. Accede a la consola de administración del servidor DNS.

   1.2. Sobre **Zonas de búsqueda inversa** haz clic con el botón derecho del ratón y selecciona **Zona nueva**.

   1.3. Lee la información que ofrece el asistente y haz clic en **Siguiente** para continuar.

   1.4. Selecciona **Zona principal** y haz clic en **Siguiente**.

   1.5. Selecciona **Zona de búsqueda inversa para IPv4** y haz clic en **Siguiente**.

   1.6. Introduce 192.168.1 como identificador de red de la zona (observa cuál es el nombre real de la zona) y haz clic en **Siguiente**.

   1.7. Deja seleccionada la opción **Crear un archivo nuevo con este nombre** y mantén el nombre que sugiere el asistente. Pincha en **Siguiente**.

   1.8. Selecciona la opción **no admitir actualizaciones dinámicas** y pincha en **Siguiente**.

   1.9. Lee el resumen y pincha **Finalizar**.

   1.10. Observa que se ha creado una entrada en Zonas de búsqueda inversa con el nombre de la zona (`1.168.192.in-addr.arpa`).

   1.11. Pincha sobre el nombre de la zona y observa en la zona central de la consola los registros de recursos que se han creado automáticamente. Se han añadido a la zona el registro **SOA** y un registro **NS** indicado que el servidor DNS para el dominio es este equipo (**servidorw2008XX**).

   1.12. Haz doble clic sobre el registro SOA y observa sus propiedades (consulta el apartado del capítulo dedicado a este registro).

1.13. Haz doble clic sobre el registro NS y observa sus propiedades. Fíjate que en el campo dirección IP del nombre de dominio del equipo si aparece su dirección IP porque ya se creó un registro A para el nombre `servidorw2008XX.dawXX.net`.

1.14. Crea los registros PTR para los nombres de los equipos de la red virtual.

a. Sobre la zona `1.168.192.in-addr.arpa` haz clic con el botón derecho del ratón y selecciona **Nuevo puntero (PTR)**.

b. Introduce la dirección IP y el nombre asociado. Puedes pinchar en **Examinar** para seleccionar los registros A que creaste previamente.

Véase la Figura 3.8 con la zona configurada.

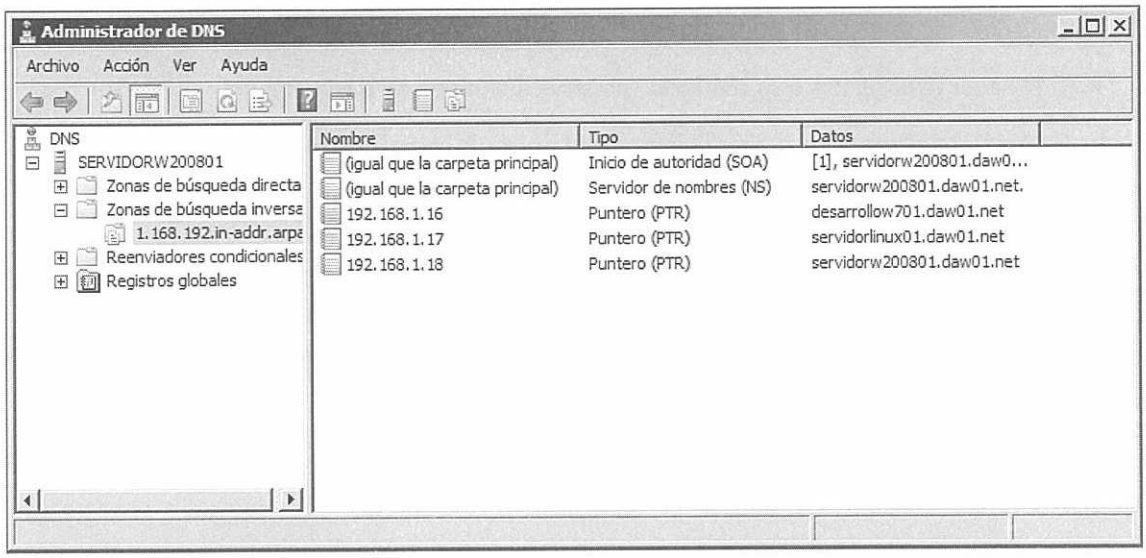


Figura 3.8: Zona de resolución inversa

2. **Comprobar la configuración**

2.1. Usa el comando `nslookup` para comprobar que el servidor DNS resuelve consultas inversas sobre direcciones, Figura 3.9.

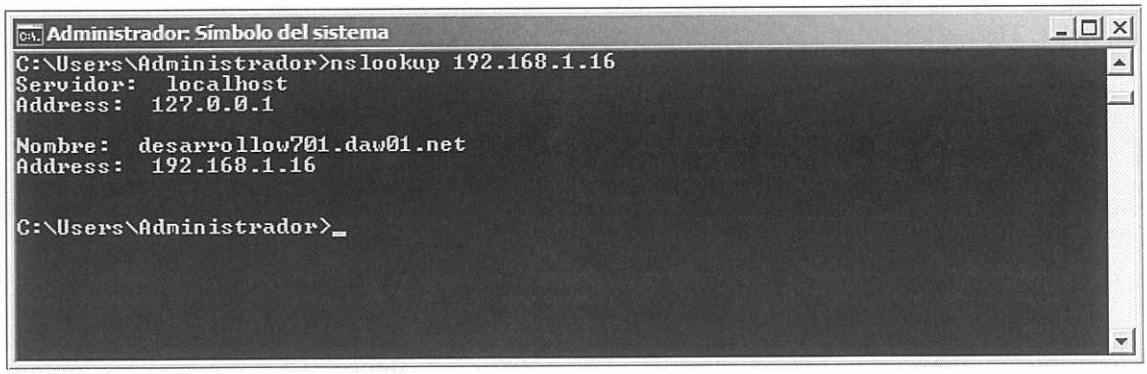


Figura 3.9: Resolución inversa

3.5. Servidor DNS en *Microsoft Windows 2012 Server*. Instalación y configuración del servidor DNS como solo *cache*

Instala y configura un servidor DNS en **ServidorW2012XX** para que actúe como solo *cache* y responda a consultas recursivas.

1. **Instalación**

   1.1. Inicia una sesión en **ServidorW2012XX** con el usuario **administrador**.

   1.2. Accede a **Administrador del Servidor**.

   1.3. Selecciona **Agregar Roles y Características**.

   1.4. Lee la información que muestra el asistente y haz clic en **Siguiente**.

   1.5. Selecciona **Instalación basada en características y roles** y haz clic en **Siguiente**.

   1.6. Selecciona tu servidor y haz clic en **Siguiente**.

   1.7. Selecciona **Servidor DNS** y haz clic en **Siguiente**.

   1.8. Haz clic en **Agregar características** y haz clic en **Siguiente**.

   1.9. No selecciones ninguna característica mas. Haz clic en **Siguiente**.

   1.10. Lee la información que se ofrece sobre el servidor y sobre aspectos que hay que tener en cuenta y haz clic en **Siguiente**.

   1.11. Confirma las selecciones de instalación haciendo clic en **Instalar**. Comenzará el proceso de instalación.

   1.12. Cuando finalice el proceso de instalación cierra el asistente.

   1.13. Observa que en el Administrador del servidor, en la columna de la izquierda, aparece un enlace al **DNS**. Si pinchas en él se muestra información sobre eventos, servicios y recursos y soporte técnico del servidor. Observa que el servidor DNS está iniciado, véase Figura 3.10.

   1.14. Abre un terminal, ejecuta el comando `netstat -a -n` y comprueba que el servidor está a la escucha en los puertos 53 TCP y UDP.

   1.15. En el Administrador de servidor, en la parte superior selecciona **Herramientas, DNS** para acceder a la consola de administración del servidor. Accede a ella, Figura 3.11.

   1.16. Puedes comprobar accediendo al *Firewall* de *Windows* que se ha creado una excepción para el servidor DNS.

2. **Configuración del servidor como solo *cache***

   Por defecto el servidor está configurado como solo *cache* (no es autorizado ninguna zona) que responde a consultas recursivas.

   2.1. Comprueba que el servidor resuelve nombres de dominio de Internet configurando el cliente DNS para que utilice el servidor DNS instalado en la máquina local (127.0.0.1), véase Figura 3.12.

Figura 3.10: Servidor DNS instalado en iniciado

Figura 3.11: Consola de administración del servidor DNS

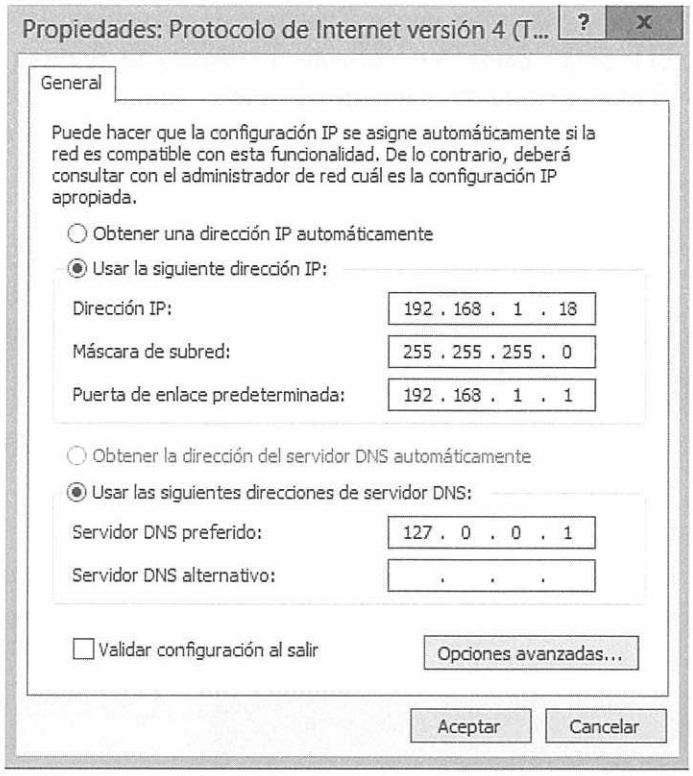


Figura 3.12: Configuración TCP/IP de la máquina ServidorW2008XX

2.2. Usa el comando `nslookup` para resolver un nombre, por ejemplo `www.madrid.org`.

2.3. Accede a la consola de administración del servidor DNS.

2.4. Activa la vista avanzada en el menú **Ver, Avanzada**.

2.5. En la parte izquierda de la ventana aparece una entrada que permite consultar la *cache* del servidor. Busca el nombre de dominio por el que has preguntado y observa que, como consecuencia de la resolución de consultas recursivas, hay más información.

2.6. Desactiva la vista avanzada.

3.6. Servidor DNS en *Microsoft Windows 2012 Server*. Configuración del servidor como primario (maestro) para una zona de resolución directa

Configura el servidor DNS del equipo **ServidorW2012XX**:

- El servidor solo servirá a la red local (no sirve a equipos de Internet).
- Actuará como maestro y tendrá autoridad sobre el dominio `dawXX.net`.
  - No se permitirán actualizaciones dinámicas.
  - El servidor DNS maestro del dominio será `servidorw2012XX.dawXX.net` (es decir, el equipo donde está instalado en servidor DNS) (registro NS).

- Se configurarán los siguientes nombres nombres de dominio (registros A).
  - `desarrollow7XX.dawXX.net` asociado la dirección IP `192.168.1.X6`
  - `servidorlinuxXX.dawXX.net` asociado la dirección IP `192.168.1.X7`
  - `servidorw2012XX.dawXX.net` asociado la dirección IP `192.168.1.X8`
- Se configurarán los siguientes alias (registros CNAME).
  - `ns.dawXX.net` será un alias de `servidorw2012XX.dawXX.net`
  - `asterix.dawXX.net` será un alias de `desarrollow7XX.dawXX.net`
  - `obelix.dawXX.net` será un alias de `servidorlinuxXX.dawXX.net`
  - `panoramix.dawXX.net` será un alias de `servidorw2012XX.dawXX.net`

1. **Configuración del sufijo DNS del equipo**

   1.1. Inicia una sesión en **ServidorW2012XX** con el usuario **administrador**.

   1.2. Accede a **Panel de Control, Sistema y Seguridad, Sistema**.

   1.3. Pincha sobre **Cambiar configuración**.

   1.4. Accede a la pestaña **Nombre de equipo** y pincha en **Cambiar**.

   1.5. Pincha sobre **Más** y en **sufijo DNS principal del equipo** introduce `dawXX.net`, véase Figura 3.13.

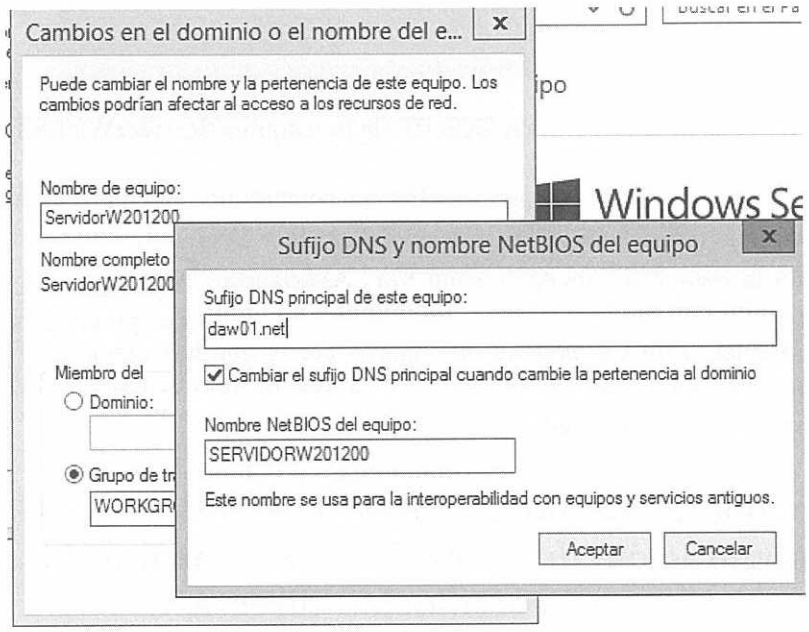


Figura 3.13: Súfijo DNS de la máquina **ServidorWindows2012XX**

   1.6. Acepta los cambios y reinicia el equipo.

2. **Configuración de la zona de resolución directa**

   2.1. Inicia una sesión en **ServidorW2012XX** con el usuario **administrador**.

2.2. Accede a la consola de administración del servidor DNS (**Administrador del servidor, Herramientas, DNS**).

2.3. Sobre **Zonas de búsqueda directa** haz clic con el botón derecho del ratón y selecciona **Zona nueva**.

2.4. Lee la información que ofrece el asistente y haz clic en **Siguiente** para continuar.

2.5. Selecciona **Zona principal** y haz clic en **Siguiente**.

2.6. Introduce `dawXX.net` como nombre de la zona y haz clic en **Siguiente**.

2.7. Deja seleccionada la opción **Crear un archivo nuevo con este nombre** y mantén el nombre que sugiere el asistente. Pincha en **Siguiente**.

2.8. Selecciona la opción **No admitir actualizaciones dinámicas** y pincha en **Siguiente**.

2.9. Lee el resumen y pincha **Finalizar**.

2.10. Observa que se ha creado una entrada en "**Zonas de búsqueda directa**" con el nombre de la zona (`dawXX.net`).

2.11. Pincha sobre el nombre de la zona y observa en la zona central de la consola los registros de recursos que se han creado automáticamente. Se han añadido a la zona el registro **SOA** y un registro **NS** indicando que el servidor DNS para el dominio es este equipo (**ServidorW2012XX**).

2.12. Haz doble clic sobre el registro SOA y observa sus propiedades.

2.13. Haz doble clic sobre el registro NS y observa sus propiedades. Fíjate que en el campo dirección IP del nombre de dominio del equipo aparece la IP del equipo local. E

2.14. Observa que se ha creado un registro A para el nombre `ServidorW2012XX.dawXX.net`.

2.15. Crea los registros A para los nombres de los equipos de la red virtual.

   a. Sobre la zona `dawXX.net` haz clic con el botón derecho del ratón y selecciona **Host nuevo (A o AAAA)**.
   b. Introduce el nombre y la IP asociada.
   c. Si estuviese creada una zona de resolución inversa podrías marcar la opción **Crear registro del puntero (PTR) asociado**.

2.16. Crea los registros CNAME para los alias.

   a. Sobre la zona `dawXX.net` haz clic con el botón derecho del ratón y selecciona **Alias nuevo (CNAME)**.
   b. Introduce el nombre del alias y el nombre del dominio asociado. Puedes pinchar en **Examinar** para seleccionar los registros A que creaste previamente.

   Véase la Figura 3.14 con la zona configurada.

3. <u>Comprobar la configuración</u>

3.1. Usa el comando `nslookup` para comprobar que el servidor DNS resuelve consultas directas sobre los nombres de la zona `dawXX.net` (recuerda que las prácticas anteriores configuraste la máquina **ServidorW2012XX** para que usase como servidor DNS el 127.0.0.1), véase Figura 3.15.

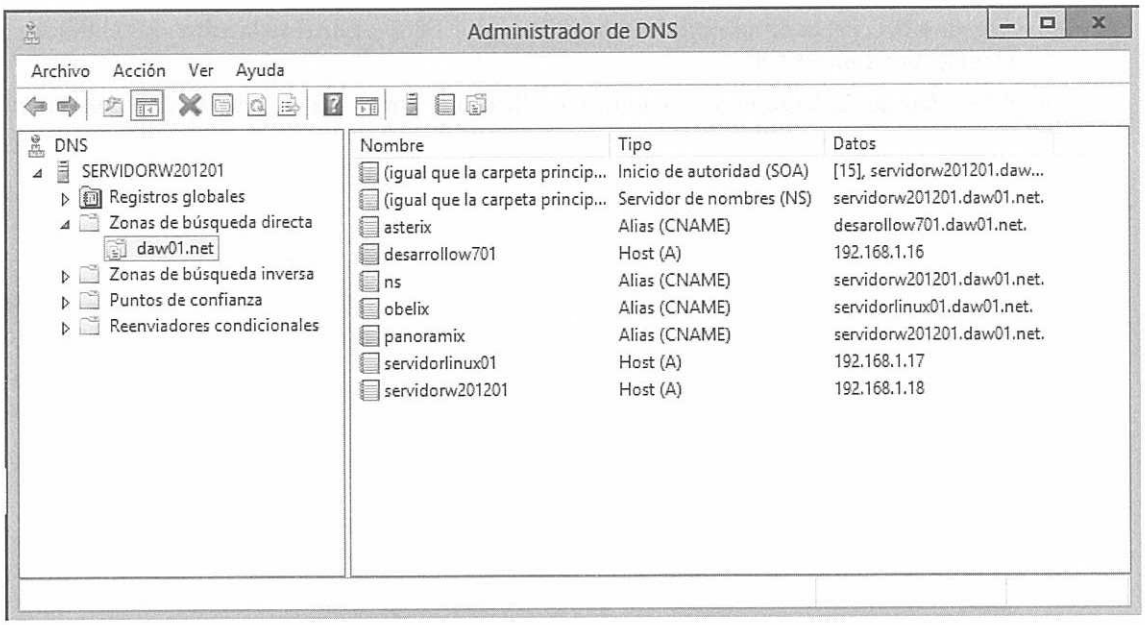


Figura 3.14: Zona primaria de resolución directa

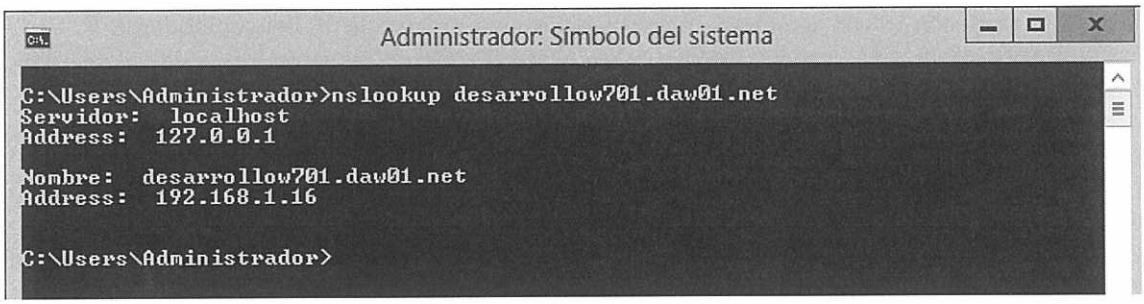


Figura 3.15: Resolución directa

3.2. Comprueba que si usas un nombre DNS sin puntos, se completa con el sufijo DNS configurado en el equipo, véase Figura 3.16.

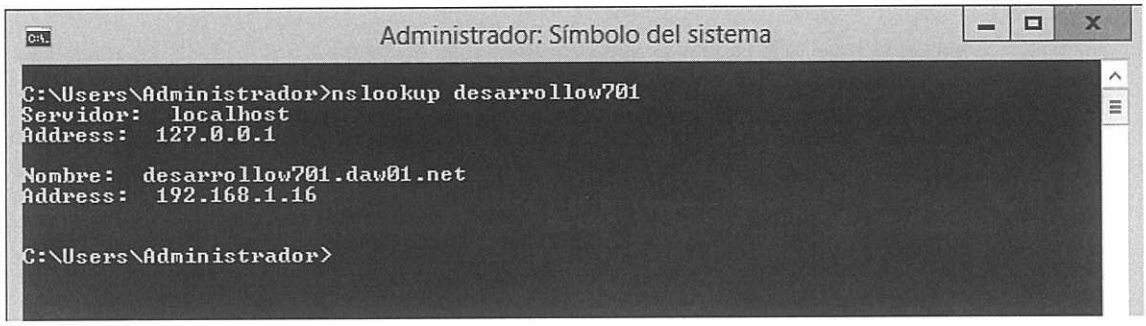


Figura 3.16: Resolución directa, se completa con el súfijo DNS del equipo

3.3. Usa *nslookup* para enviar una consulta de resolución inversa al servidor sobre algunas de las direcciones IP usadas en los registro A de la zona de resolución inversa. Observa que no resuelve los nombres.

3.7. Servidor DNS en *Microsoft Windows 2012 Server*. Configuración del servidor como primario (maestro) para una zona de resolución inversa

Configura el servidor DNS del equipo **ServidorW2012XX**:

- El servidor solo servirá a la red local (no sirve a equipos de Internet).
- Actuará como maestro y tendrá autoridad sobre la zona de resolución inversa de la red 192.168.1.0/24.
  - No se permitirán actualizaciones dinámicas.
  - El servidor DNS maestro del dominio será `servidorw2012XX.dawXX.net` (es decir, el equipo donde está instalado en servidor DNS) (registro NS).
  - Las direcciones IP de los equipos se corresponderán las representadas en el esquema de la red virtual (registros PTR).

1. <u>**Configuración de la zona de resolución inversa**</u>

   1.1. Accede a la consola de administración del servidor DNS.

   1.2. Sobre **Zonas de búsqueda inversa** haz clic con el botón derecho del ratón y selecciona **Zona nueva**.

   1.3. Lee la información que ofrece el asistente y haz clic en **Siguiente** para continuar.

   1.4. Selecciona **Zona principal** y haz clic en **Siguiente**.

   1.5. Selecciona **Zona de búsqueda inversa para IPv4** y haz clic en **Siguiente**.

   1.6. Introduce 192.168.1 como identificador de red de la zona (observa cuál es el nombre real de la zona) y haz clic en **Siguiente**.

   1.7. Deja seleccionada la opción **Crear un archivo nuevo con este nombre** y mantén el nombre que sugiere el asistente. Pincha en **Siguiente**.

   1.8. Selecciona la opción **no admitir actualizaciones dinámicas** y pincha en **Siguiente**.

   1.9. Lee el resumen y pincha **Finalizar**.

   1.10. Observa que se ha creado una entrada en Zonas de búsqueda inversa con el nombre de la zona (`1.168.192.in-addr.arpa`).

   1.11. Pincha sobre el nombre de la zona y observa en la zona central de la consola los registros de recursos que se han creado automáticamente. Se han añadido a la zona el registro **SOA** y un registro **NS** indicado que el servidor DNS para el dominio es este equipo (**servidorw2012XX**).

   1.12. Haz doble clic sobre el registro SOA y observa sus propiedades (consulta el apartado del capítulo dedicado a este registro).

1.13. Haz doble clic sobre el registro NS y observa sus propiedades. Fíjate que en el campo dirección IP del nombre de dominio del equipo si aparece su dirección IP porque ya se creó un registro A para el nombre `servidorw2012XX.dawXX.net`.

1.14. Crea los registros PTR para los nombres de los equipos de la red virtual.

a. Sobre la zona `1.168.192.in-addr.arpa` haz clic con el botón derecho del ratón y selecciona **Nuevo puntero (PTR)**.

b. Introduce la dirección IP y el nombre asociado. Puedes pinchar en **Examinar** para seleccionar los registros A que creaste previamente.

Véase la Figura 3.17 con la zona configurada.

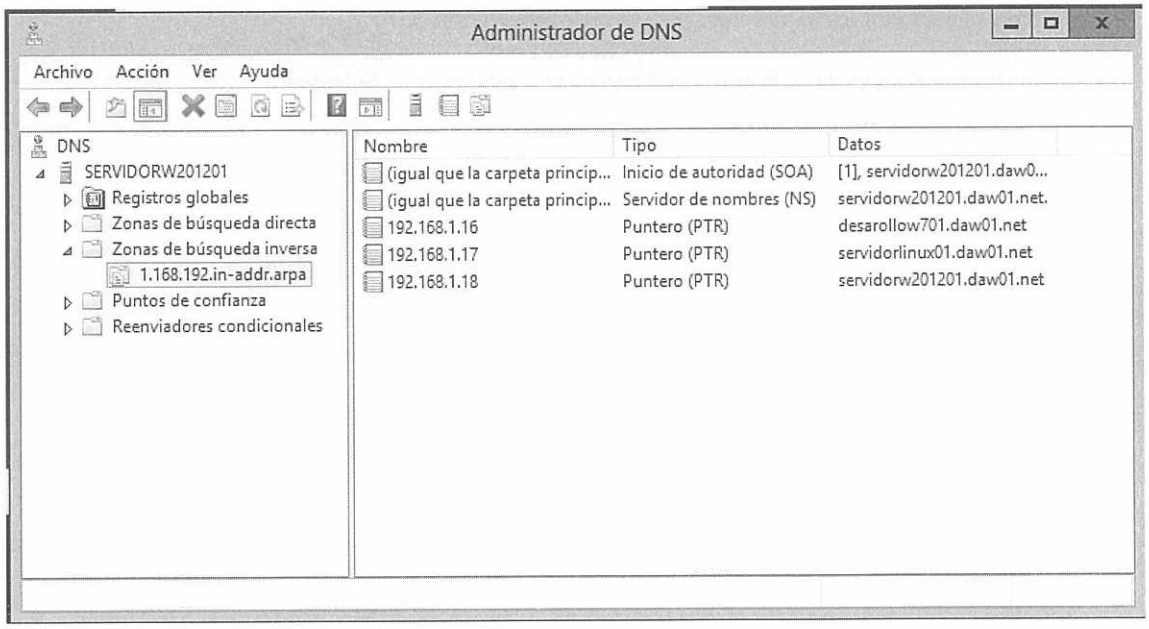


Figura 3.17: Zona de resolución inversa

2. **Comprobar la configuración**

2.1. Usa el comando `nslookup` para comprobar que el servidor DNS resuelve consultas inversas sobre direcciones, Figura 3.18.

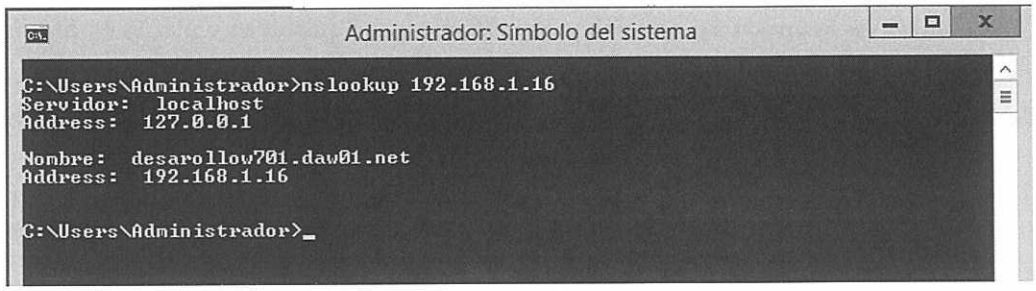


Figura 3.18: Resolución inversa

3.8. Cliente DNS en las otras máquinas

Configura el cliente DNS de las máquinas **DesarrolloW701XX** y **ServidorLinuxXX** para que utilicen el servidor DNS instalado en **ServidorW2008XX** .

1. **Configuración de la máquina DesarrolloW7XX.**

   1.1. Inicia sesión en **DesarrolloW7XX** con un usuario con privilegios de administrador.

   1.2. Accede a las propiedades del protocolo de Internet versión 4, **TCP/IPv4** (**Menú Inicio, Panel de control, Redes e Internet, Centro de redes y recursos compartidos, Conexión de área local, Propiedades, Protocolo de Internet versión 4**) y configura **ServidorW2008XX** como **Servidor DNS preferido**, véase Figura 3.19.

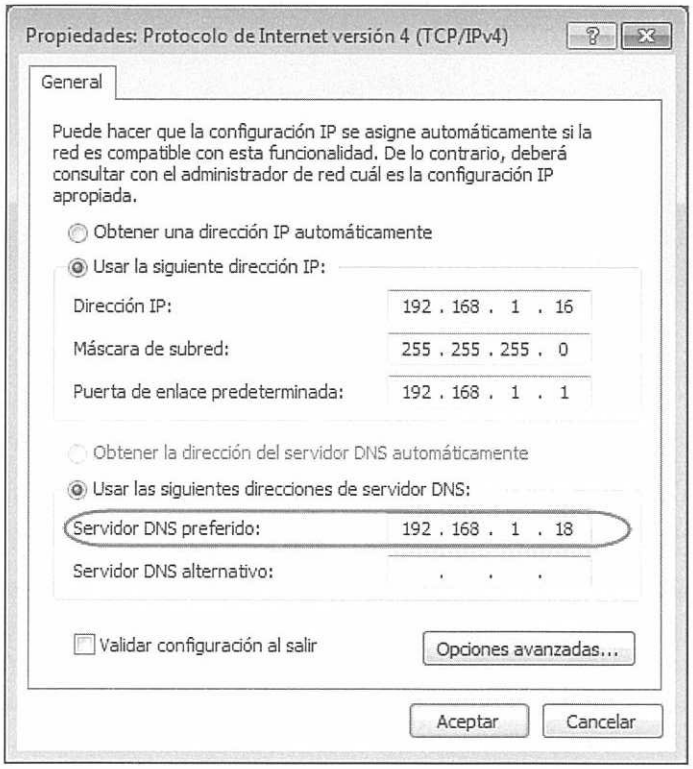


Figura 3.19: Configuración TCP/IP de la máquina DesarrolloW7XX

   1.3. Accede a **Inicio** sitúate sobre **Equipo**, pulsa el botón derecho del ratón y selecciona **Propiedades**.

   1.4. Pincha sobre **Configuración avanzada del sistema**.

   1.5. Accede a la pestaña **Nombre de equipo** y pincha en **Cambiar**.

   1.6. Pincha sobre **Más** y en **sufijo DNS principal del equipo** introduce `dawXX.net`.

   1.7. Acepta los cambios y reinicia el equipo.

   1.8. Usa el comando `nslookup` para comprobar que el servidor DNS resuelve consultas directas sobre los nombres de la zona `dawXX.net`, véanse Figuras 3.20 y 3.21.

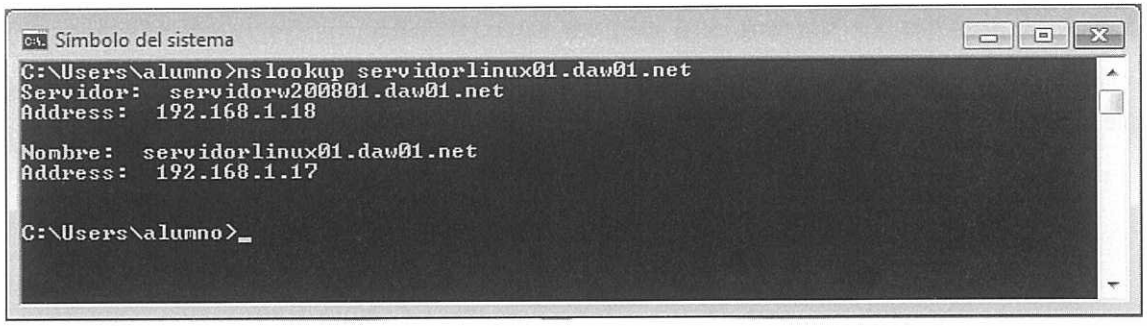


Figura 3.20: Resolución directa

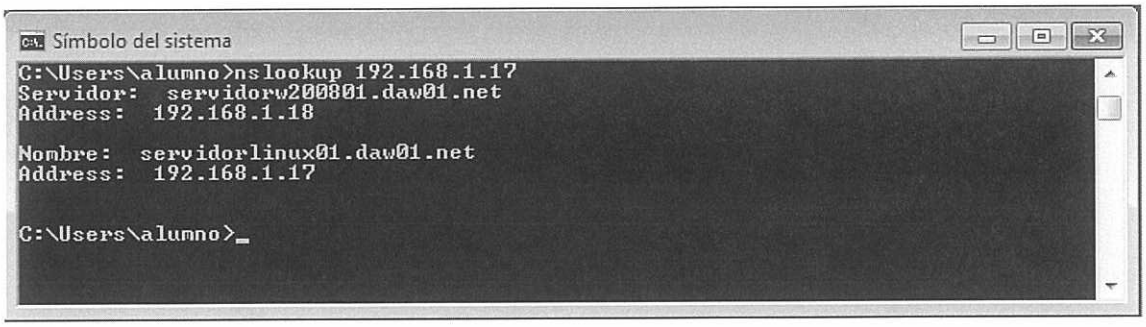


Figura 3.21: Resolución inversa

1.9. Comprueba que si usas un nombre DNS sin puntos, se completa con el sufijo DNS configurado en el equipo.

2. **Configuración de la máquina ServidorLinuxXX.**

2.1. Inicia sesión en **ServidorLinuxXX** con un usuario con privilegios de administrador.

2.2. Edita el fichero de configuración **/etc/network/interfaces**. Modifica el fichero y cambia el servidor DNS y añade `dawXX.net` como sufijo DNS del equipo, véase Figura 3.22.

2.3. Reinicia el servicio de red con el comando `sudo /etc/init.d/networking restart` para aplicar los cambios.

2.4. Ejejecuta el comando `ifconfig` para verificar la configuración.

2.5. Consulta el fichero de configuración **/etc/resolv.conf** y observa que se utilizan el servidor DNS y el sufijo que has definido previamente, Figura 3.23.

2.6. Edita el fichero de configuración **/etc/hostname** y escribe en el fichero el nombre que deseamos que tenga la máquina (**ServidorLinuxXX.dawXX.net**), véase Figura 3.24.

2.7. Edita el fichero **/etc/hosts** y asocia el nombre **ServidorLinuxXX.dawXX.net** con la dirección IP de bucle interno (127.0.1.1) como se indica en la Figura 3.25.

2.8. Reinicia la máquina con el comando `sudo reboot`.

2.9. Usa el comando `dig` para comprobar que el servidor DNS resuelve consultas directas sobre los nombres de la zona `dawXX.net`, véanse Figuras 3.26 y 3.27.

```
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto eth0
iface eth0 inet static
        address 192.168.1.17
        netmask 255.255.255.0
        gateway 192.168.1.1
        dns-nameservers 192.168.1.18
        dns-search daw01.net
```

Figura 3.22: /etc/network/interfaces

```
# Dynamic resolv.conf(5) file for glibc resolver(3) generated by resolvconf(8)
#     DO NOT EDIT THIS FILE BY HAND -- YOUR CHANGES WILL BE OVERWRITTEN
nameserver 192.168.1.18
search daw01.net
```

Figura 3.23: /etc/resolv.conf

```
ServidorLinux01.daw01.net
```

Figura 3.24: /etc/hostname

```
127.0.0.1       localhost
127.0.1.1       ServidorLinux01.daw01.net
```

Figura 3.25: /etc/hosts

```
alumno@ServidorLinux01:~$ dig asterix.daw01.net

; <<>> DiG 9.8.1-P1 <<>> asterix.daw01.net
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 2956
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;asterix.daw01.net.             IN      A

;; ANSWER SECTION:
asterix.daw01.net.      3600    IN      CNAME   desarrollow701.daw01.net.
desarrollow701.daw01.net. 3600  IN      A       192.168.1.16

;; Query time: 16 msec
;; SERVER: 192.168.1.18#53(192.168.1.18)
;; WHEN: Wed Apr 25 15:22:50 2012
;; MSG SIZE  rcvd: 80
```

Figura 3.26: Resolución directa

```
alumno@ServidorLinux01:~$ dig -x 192.168.1.16

; <<>> DiG 9.8.1-P1 <<>> -x 192.168.1.16
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 9618
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;16.1.168.192.in-addr.arpa.      IN      PTR

;; ANSWER SECTION:
16.1.168.192.in-addr.arpa. 3600 IN      PTR     desarrollow701.daw01.net.

;; Query time: 13 msec
;; SERVER: 192.168.1.18#53(192.168.1.18)
;; WHEN: Wed Apr 25 15:23:21 2012
;; MSG SIZE  rcvd: 81
```

Figura 3.27: Resolución inversa

Capítulo 4

WWW y protocolo HTTP

4.1. Protocolo HTTP

En esta práctica se analiza la información de los mensajes de petición y respuesta del protocolo HTTP.

1. Inicia sesión con un usuario con privilegios de administrador en **DesarrolloW7XX**.
2. Abre el navegador *Google Chrome*.
3. Inicia una captura con *Wireshark* (**Inicio, Todos los programas, *Wireshark, Capture, Interfaces, Start***).
4. Desde el navegador *Google Chrome* establece una conexión a `http://http://www.apache.org/`.
5. Accede a *Wireshark* y para la captura (***Capture, Stop***).
6. Buscar una trama HTTP en donde la petición sea **GET / HTTP/1.1**, véase Figura 4.1.

63 4.976046	192.168.1.16	140.211.11.131	HTTP	424 GET / HTTP/1.1

Figura 4.1: *Wireshark*

7. Haz clic con el botón derecho del ratón y selecciona ***Follow TCP Stream***, véase Figura 4.2.
8. Responde a las siguientes preguntas:
   8.1. ¿Cuál es la IP de la máquina donde se ejecuta el servidor Web? `140.211.11.131`.
   8.2. ¿Qué versión de HTTP se utiliza? *HTTP 1.1*.
   8.3. ¿Qué método de petición se utiliza? *GET*.
   8.4. ¿Qué recurso se solicita al servidor? *El directorio raíz*.
   8.5. ¿Qué valor tiene la cabecera ***Host***? `www.apache.org`
   8.6. ¿Se envían *cookies* en la petición HTTP? *No*.
   8.7. ¿Que lenguaje utiliza el navegador? *es-ES*.
   8.8. ¿Qué código de estado tiene la respuesta HTTP? *200*.
   8.9. ¿Qué servidor Web y versión se utiliza? *Apache/2.4.1 (Unix)*.
   8.10. ¿De qué tipo MIME es el recurso recibido? *text/html*.
   8.11. ¿Se han utilizado conexiones persistentes, es decir, en la misma conexión TCP hay varias peticiones y respuestas HTTP? *Sí*.

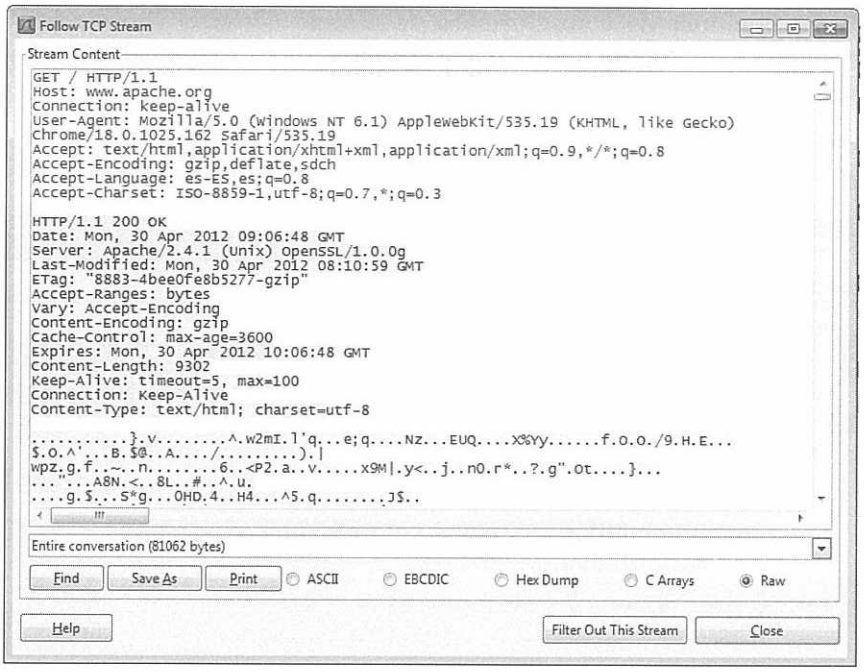


Figura 4.2: Tráfico HTTP

8.12. ¿Existen peticiones y respuestas de imágenes? *Sí*, véase Figura 4.3.

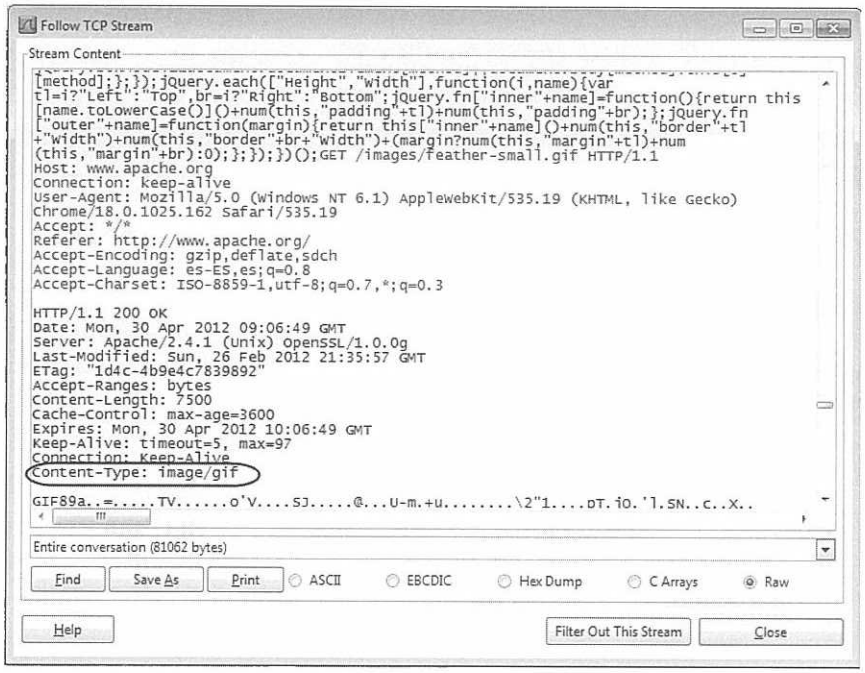


Figura 4.3: Envío de una imagen

9. Accede a las **opciones de configuración de** *Google Chrome* **(botón con una herramienta en la parte superior derecha), Herramientas, Herramientas para desarrolladores.**

10. Accede a `http://tomcat.apache.org/` y analiza la peticiones y respuestas HTTP, qu,e métodos usan, los códigos de respuesta, los recursos que envía el servidor, ..., Figura 4.4.

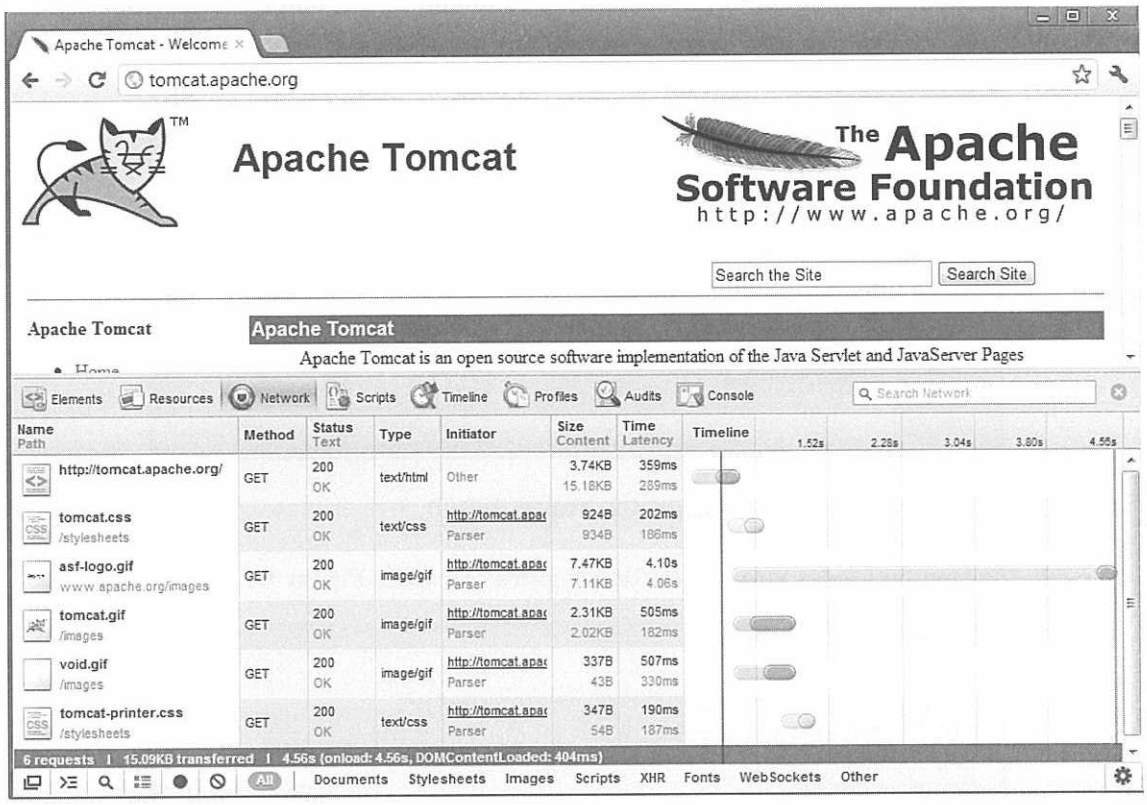


Figura 4.4: Herramienta para desarrolladores de *Google Chrome*

11. Accede a las **opciones de configuración de** *Google Chrome* **(botón con una herramienta en la parte superior derecha), Configuración, Avanzada, Configuración de contenido, Todas las cookies y los datos de sitios** y observa las *cookies* que tiene almacenadas el navegador.

Capítulo 5

Administración de Apache

5.1. Instalación del servidor web *Apache 2.4* en *Linux*

Instala el servidor web *Apache 2.4* en la máquina **ServidorLinuxXX** desde los repositorios de software de *Ubuntu*. Prueba el funcionamiento del servidor estableciendo conexiones desde **DesarrolloW7XX** empleando tanto direcciones IP como nombres de dominio.

1. **Instalación**

   1.1. Inicia sesión con un usuario con privilegios de administrador en **ServidorLinuxXX**.

   1.2. Instala el servidor *Apache 2.4* desde los repositorios oficiales de *Ubuntu*.

   ```
   sudo apt-get update
   sudo apt-get install apache2
   ```

   - Se crean los archivos de configuración.
   - Se crean el usuario **www-data** que se añade al grupo **www-data**. Son el usuario y el grupo con el que se ejecutan los procesos de *Apache* que se encargan de atender peticiones.
   - Se crea el directorio **/var/www/html**.
     - Su propietario es **root** y su grupo es **root**.
     - Es el directorio raíz del servidor virtual por defecto.

   1.3. Comprueba que el servidor está iniciado y escuchando en el puerto 80/TCP, Figura 5.1.

   ```
   ps -ef | grep apache
   netstat -ltn
   ```

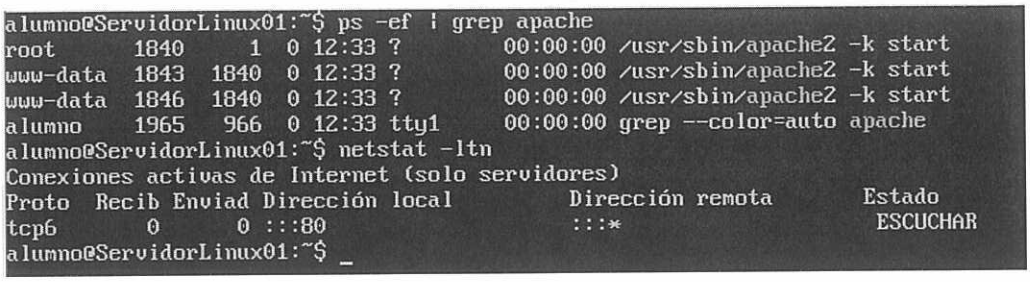


Figura 5.1: *Apache* iniciado y escuchando en el puerto 80/TCP

   1.4. Comprueba que se ha creado el directorio **/var/www/html** y que su propietario es el usuario **root**.

   ```
   ls -l /var
   ```

   1.5. Ejecuta el siguiente comando para ver las opciones con las que se ha compilado la versión de *Apache* instala.

   ```
   apache2 -v
   ```

   1.6. Consulta el contenido del directorio **/var/www/html**, Figura 5.2.

   1.7. Consulta el contenido del fichero **/var/www/html/index.html**, Figura 5.3.

```
alumno@ServidorLinux01:/var/www/html$ ls
index.html
alumno@ServidorLinux01:/var/www/html$
```

Figura 5.2: Directorio **/var/www/html**

```
<div class="content_section_text">
  <p>
        Please use the <tt>ubuntu-bug</tt> tool to report bugs in the
        Apache2 package with Ubuntu. However, check <a
        href="https://bugs.launchpad.net/ubuntu/+source/apache2">existi
g
        bug reports</a> before reporting a new bug.
  </p>
  <p>
        Please report bugs specific to modules (such as PHP and others)
        to respective packages, not to the web server itself.
```

Figura 5.3: Fichero **/var/www/html/index.html**

2. <u>**Prueba de conexión al servidor**</u>

   2.1. Inicia sesión **DesarrolloW7XX**.

   2.2. Abre un navegador y accede a `http://192.168.1.X7` (como no se especifica el puerto se utiliza por defecto el puerto 80), Figura 5.4.

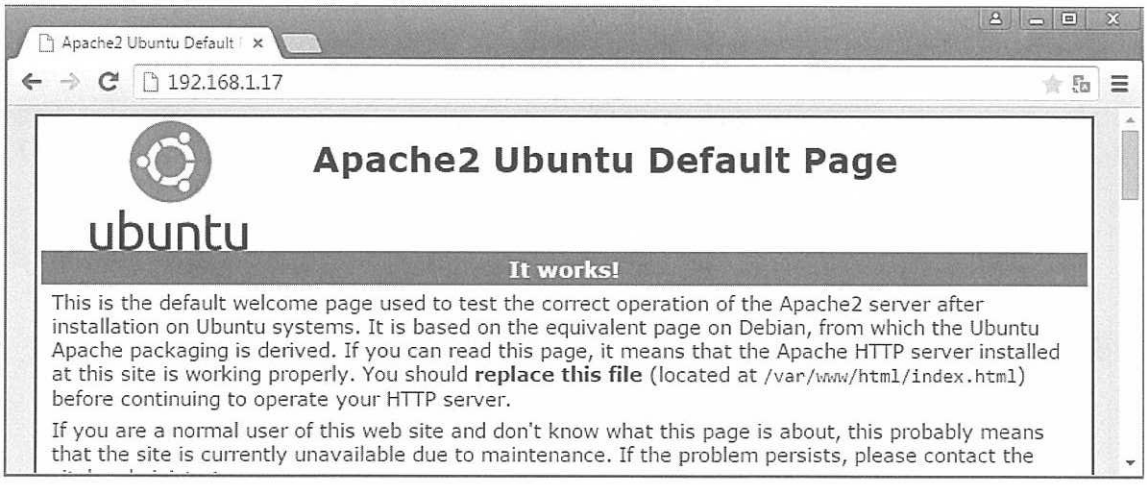


Figura 5.4: Conexión a *Apache*

   2.3. Observa que *Apache* sirve por defecto el fichero **index.html** (que está en el directorio **/var/www/html**).

   2.4. También puedes usar los nombres DNS `servidorlinuxXX.dawXX.net` y `obelix.dawXX.net`, véanse Figuras 5.5 y 5.6.

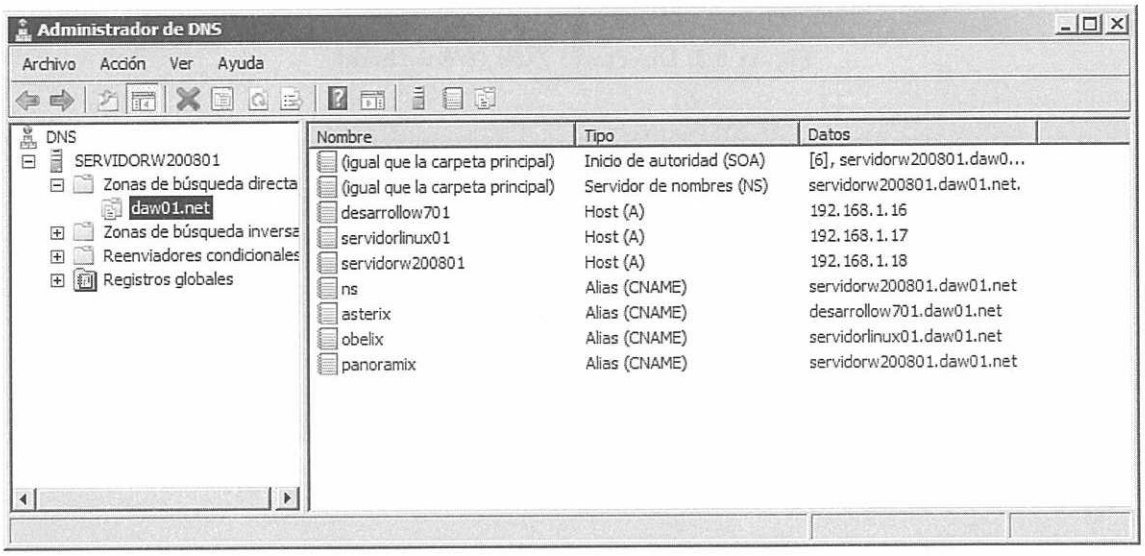


Figura 5.5: Zona **dawXX.net**

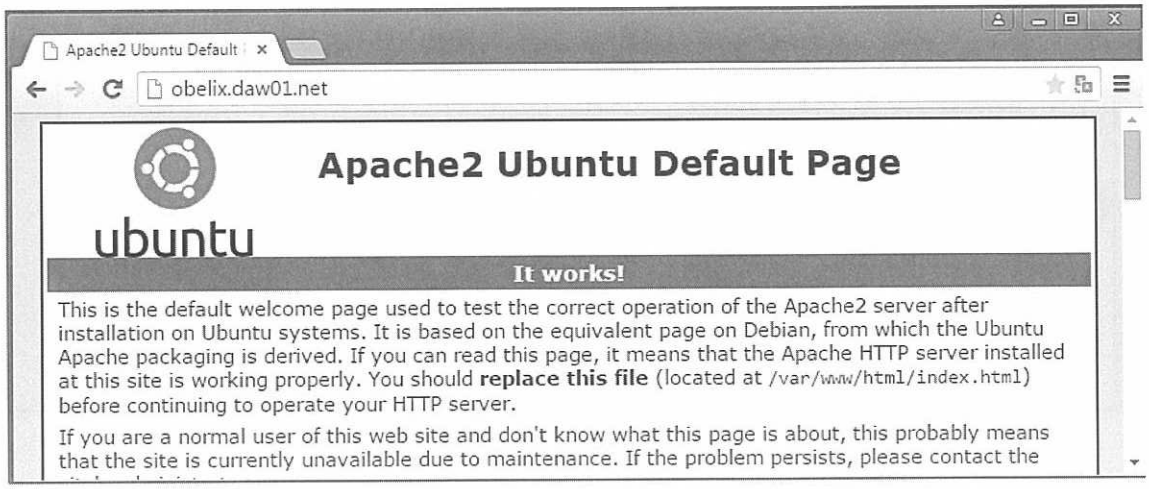


Figura 5.6: Conexión a *Apache*

5.2. Instalación del servidor web *Apache 2.2* en *Windows*

Instala el servidor web *Apache 2.2* en la máquina **ServidorW2008XX** o **ServidorW2012XX** usando los paquetes binarios. Prueba el funcionamiento del servidor estableciendo conexiones desde **DesarrolloW7XX** empleando tanto direcciones IP como nombres de dominio.

1. <u>**Instalación**</u>

   1.1. Inicia sesión con un usuario con privilegios de administrador en **ServidorW2008XX**.

   1.2. Inicia el navegador *Google Chrome* y accede a `http://httpd.apache.org/download.cgi`, Figura 5.7.

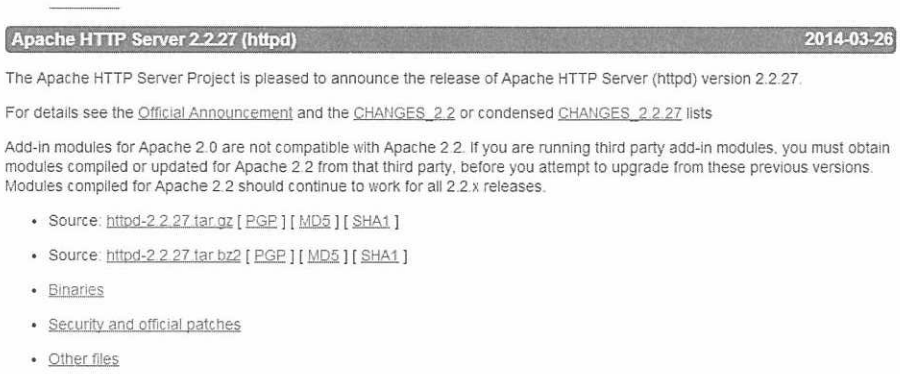


Figura 5.7: Descarga de *Apache* para *Windows*

   1.3. Accede a ***Other Files***.

   1.4. Accede a ***binaries*** y luego a ***win32***.

   1.5. Descarga el archivo para win32 con OpenSSL (***httpd-2.2.25-win32-x86-openssl-0.9.8y*** en el momento de probar la práctica).

   1.6. Ejecuta el instalador.

   1.7. Pincha en ***Next*** para iniciar el asistente de instalación.

   1.8. Acepta los términos de la licencia y pincha en ***Next***.

   1.9. Lee la información y pincha en ***Next***.

   1.10. Configura las opciones tal y como se muestran en la Figura 5.8 (adáptalo a tu grupo XX). Observa que se configurará el servidor para que escuche peticiones en el puerto 80.

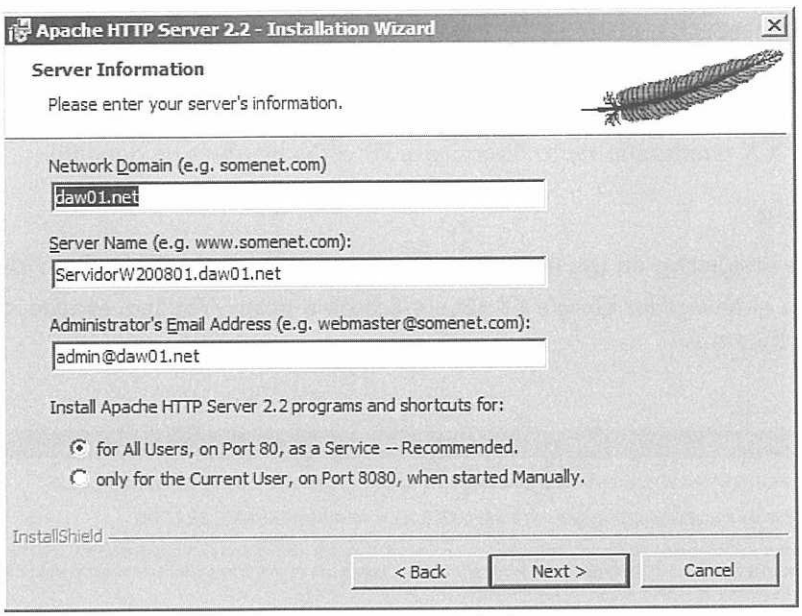


Figura 5.8: Instalación de *Apache (Server Information)*

1.11. Selecciona el tipo de instalación ***Custom*** y pincha en ***Next***.

1.12. Observa qué componentes se instalarán, Figura 5.9, y pincha en ***Next***.

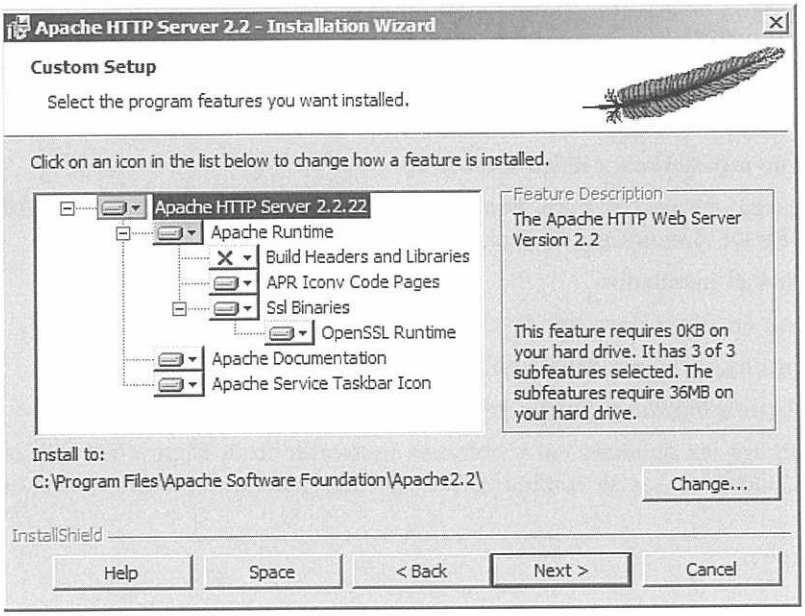


Figura 5.9: Instalación de *Apache (Custom Setup)*

1.13. Pincha en ***Install*** para continuar.

1.14. Una vez terminada la instalación pincha en ***Finish***.

1.15. Comprueba que el servidor está iniciado, en la barra de tareas abajo a la izquierda hay un icono que muestra una herramienta para monitorizar el servidor. Figura 5.10.

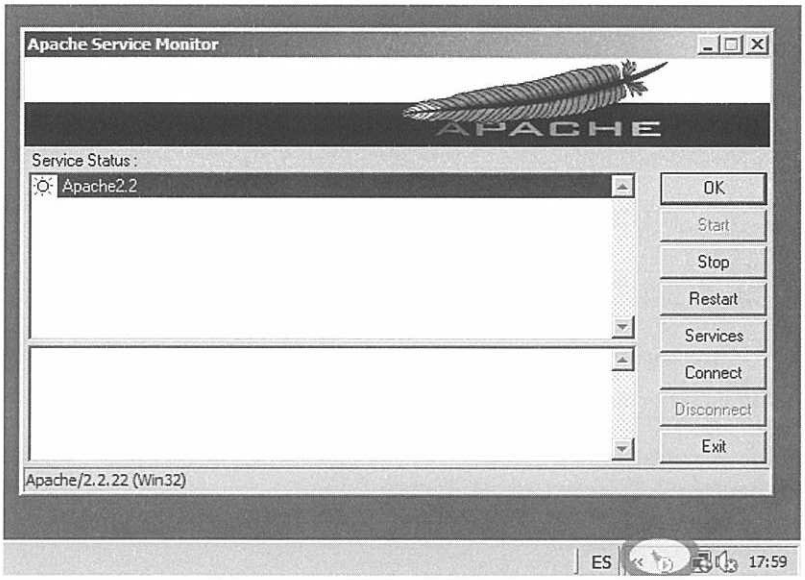


Figura 5.10: Monitorización del servidor

1.16. Comprueba que el servidor está escuchando en el puerto 80/TCP, Figura 5.11

```
netstat -a -p TCP -n
```

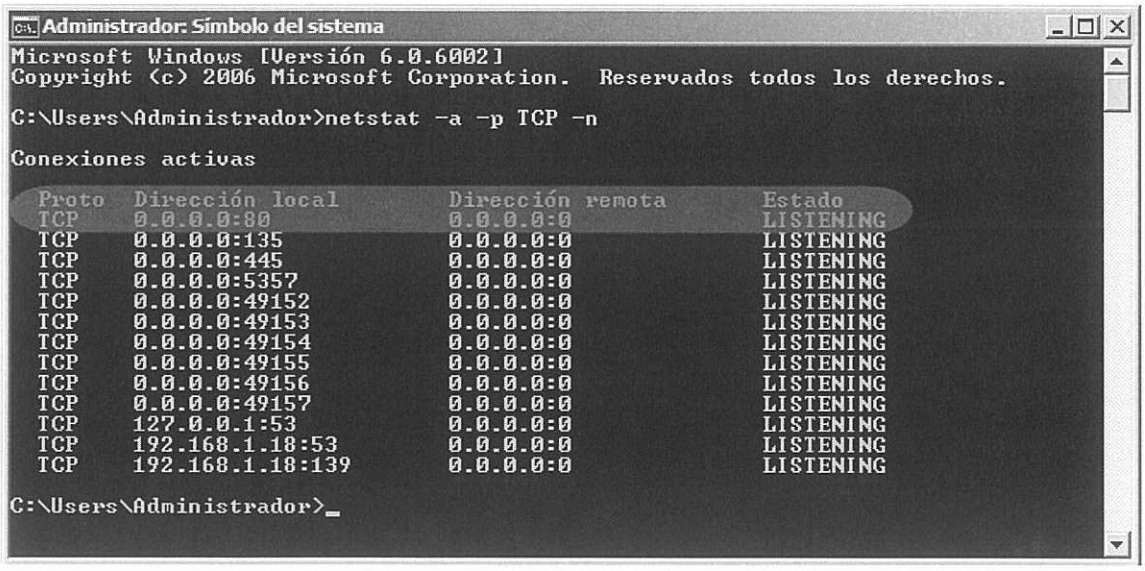


Figura 5.11: *Apache* iniciado y escuchando en el puerto 80/TCP

1.17. Consulta el contenido del directorio **C:\Program Files\Apache Software Foundation\ Apache2.2\htdocs**, Figura 5.12.

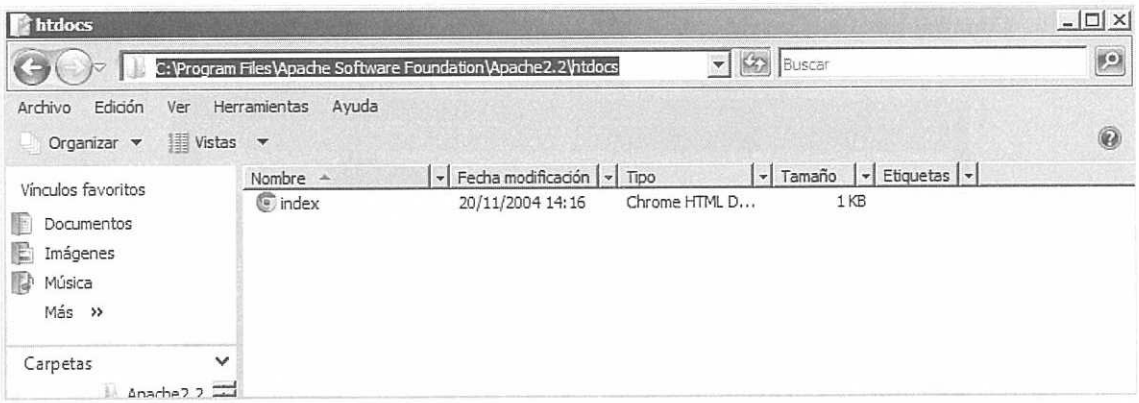


Figura 5.12: Directorio **C:\Program Files\Apache Software Foundation\Apache2.2\htdocs**

1.18. Consulta el contenido del fichero **C:\Program Files\Apache Software Foundation\ Apache2.2\index.html**, Figura 5.13.

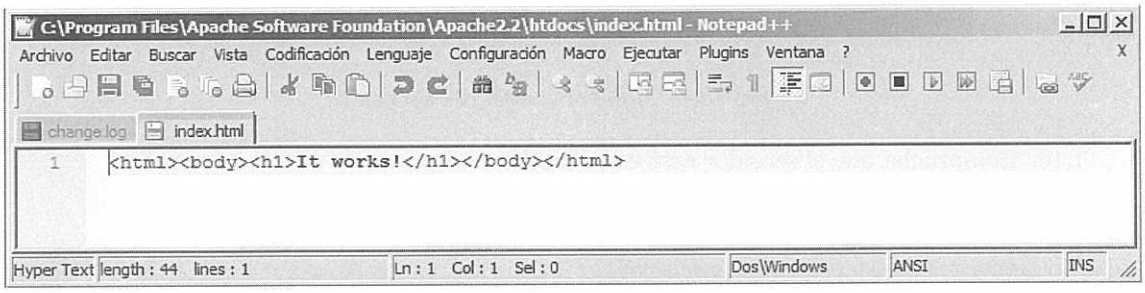


Figura 5.13: Fichero **index.html**

1.19. Habilita una regla en el *Firewall* para que se permitan conexiones a *Apache*.

   a. Accede a **Inicio, Panel de control, Seguridad, Dejar pasar un programa a través de *firewall* de *Windows***.
   b. Pincha en **Agregar Programa**.
   c. Pincha en **Examina** y selecciona **C:\Program Files\Apache Software Foundation\ Apache2.2\bin\httpd.exe**.
   d. Acepta los cambios.

1.20. Si queremos utilizar las aplicaciones de *Apache* desde la cualquier localización en la línea de comandos podemos añadir la ruta de los binarios de *Apache* en la variable de entorno **PATH**.

   a. Accede a **Inicio, Panel de control, Sistema y mantenimiento, Sistema, Configuración avanzada del sistema, Variables de Entorno**.
   b. En Variables de sistema selecciona la variable **Path** y pincha en **Editar**.
   c. Añade la ruta **C:\Program Files\Apache Software Foundation\Apache2.2\bin**, véase Figura 5.14.
   d. Acepta los cambios.

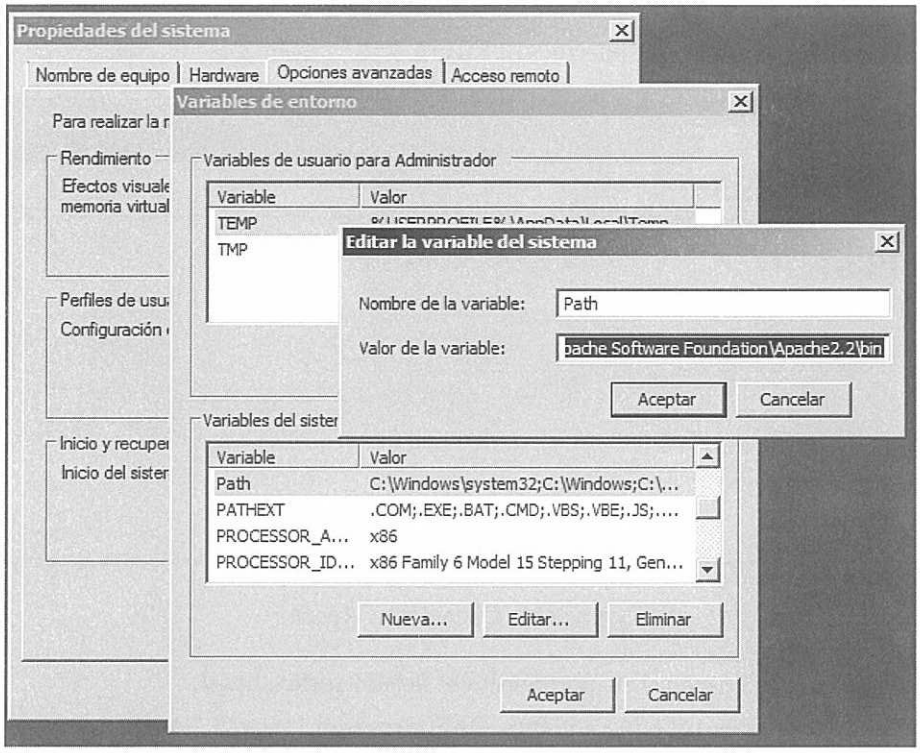


Figura 5.14: Variable de entorno Path

2. <u>Prueba de conexión al servidor</u>

   2.1. Abre un navegador y accede a `http://127.0.0.1` o a `http://localhost` (como no se especifica el puerto se utiliza por defecto el puerto 80), Figura 5.15.

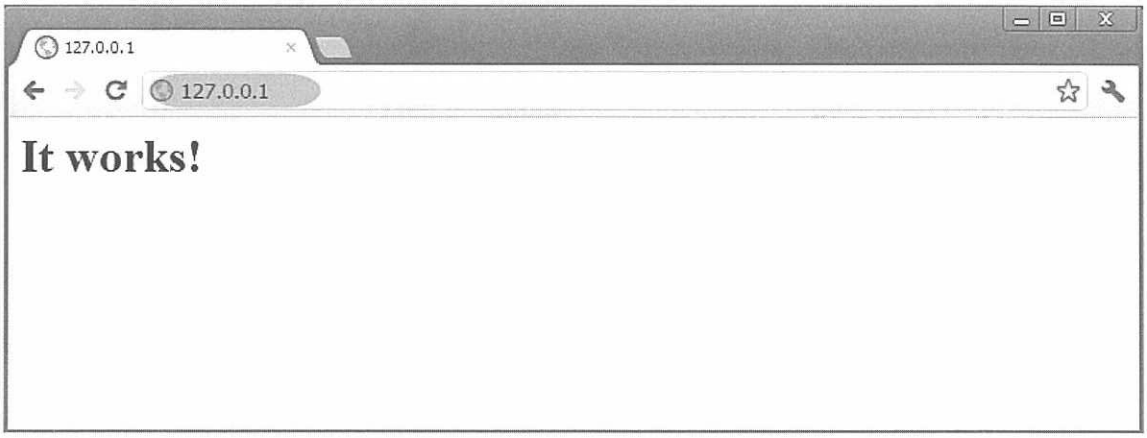


Figura 5.15: Conexión a *Apache*

   2.2. Inicia sesión **DesarrolloW7XX**.

2.3. Abre un navegador y accede a `http://192.168.1.X8` (como no se especifica el puerto se utiliza por defecto el puerto 80), Figura 5.16.

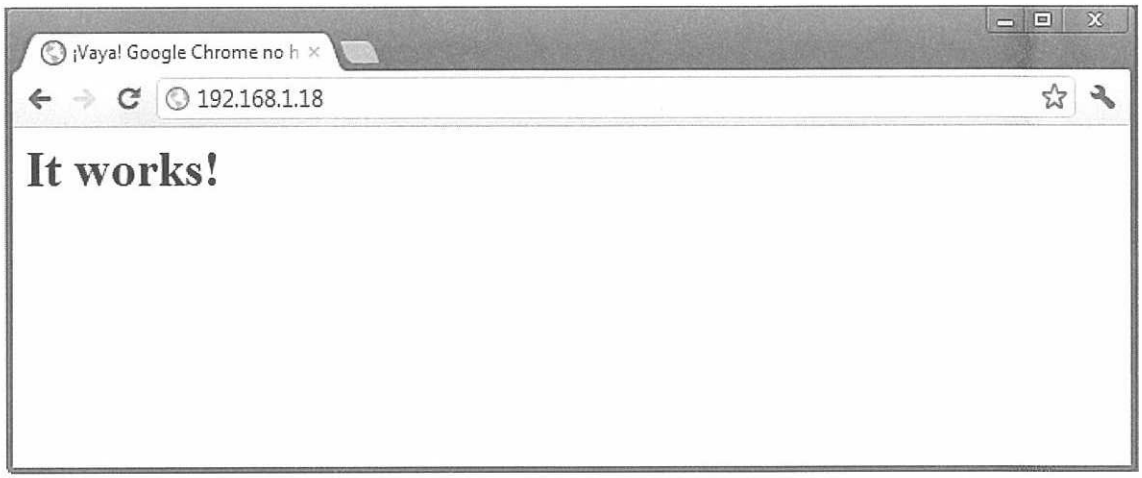


Figura 5.16: Conexión a *Apache*

2.4. Observa que *Apache* sirve por defecto el fichero **index.html**.

2.5. También puedes usar los nombres DNS `servidorlinuxXX.dawXX.net` y `panoramix.dawXX.net`, Figura 5.17.

Figura 5.17: Conexión a *Apache*

5.3. Ficheros de configuración y directivas en *Linux*

En esta práctica analizaremos los principales ficheros de configuración y el valor de algunas directivas del servidor web *Apache2.4* instalado en la máquina **ServidorLinuxXX**.

1. **Ficheros de configuración**

   1.1. Inicia sesión con un usuario con privilegios de administrador en **ServidorLinuxXX**.

   1.2. Abre un terminal y accede al directorio **/etc/apache2**.

   1.3. Haz un listado del directorio y observa los ficheros de configuración.

   1.4. Abre el fichero **/etc/apache/apache2.conf** y analiza su configuración. Observa que incluye con la directiva `include` a otros ficheros y directorios, Figura 5.18.

```
LogFormat "%h %l %u %t \"%r\" %>s %O \"%{Referer}i\" \"%{User-Agent}i\"" combi
LogFormat "%h %l %u %t \"%r\" %>s %O" common
LogFormat "%{Referer}i -> %U" referer
LogFormat "%{User-agent}i" agent

# Include of directories ignores editors' and dpkg's backup files,
# see README.Debian for details.

# Include generic snippets of statements
IncludeOptional conf-enabled/*.conf

# Include the virtual host configurations:
IncludeOptional sites-enabled/*.conf

# vim: syntax=apache ts=4 sw=4 sts=4 sr noet
```

Figura 5.18: Fichero **/etc/apache/apache2.conf**

2. **Servidor virtual por defecto**

   2.1. Accede al directorio **/etc/apache2/sites-available** y comprueba que está creado el archivo **default** que contiene la configuración del servidor virtual por defecto.

   2.2. Accede a **/etc/apache2/sites-enabled** y comprueba que existe el fichero **000-default.conf** que es un enlace simbólico a **default**, Figura 5.19 (recuerda que los ficheros incluidos en **/etc/apache2/sites-enabled** se incluyen en **/etc/apache2/apache2.conf** en orden alfabético).

```
alumno@ServidorLinux01:/etc/apache2/sites-enabled$ ls -l
total 0
lrwxrwxrwx 1 root root 35 jun  8 12:33 000-default.conf -> ../sites-available/00
0-default.conf
alumno@ServidorLinux01:/etc/apache2/sites-enabled$
```

Figura 5.19: Fichero **/etc/apache2/sites-enabled/000-default.conf**

3. **Directivas**

   3.1. Consulta el fichero **/etc/apache2/apache2.conf** y comprueba cuál es el valor de las siguientes directivas.

   - `ServerRoot`
   - `User` y `Group`
   - `TimeOut`

3.2. Consulta la documentación de *Apache* y responde a las siguientes preguntas:

a ¿Se permiten conexiones persistentes (que todas las conexiones de un usuario se atienden en la misma conexión TCP)? *Si* ¿Qué directiva define este comportamiento? `KeepAlive`.

b ¿Cuál es el fichero de erroores? **/var/log/apache2/error.log** ¿Qué directiva lo define? `ErrorLog`.

3.3. Consulta el fichero **/etc/apache2/ports.conf**, Figura 5.20 y comprueba cuál es el puerto en el que escucha peticiones *Apache* (*puerto 80*) ¿En qué puerto escuchará también si se habilita el módulo modssl? *443*.

```
# If you just change the port or add more ports here, you will likely also
# have to change the VirtualHost statement in
# /etc/apache2/sites-enabled/000-default.conf

Listen 80

<IfModule ssl_module>
        Listen 443
</IfModule>

<IfModule mod_gnutls.c>
        Listen 443
</IfModule>

# vim: syntax=apache ts=4 sw=4 sts=4 sr noet
```

Figura 5.20: Contenido del fichero **/etc/apache2/ports.conf**

3.4. Consulta el fichero **/etc/apache2/sites-available/default** y observa, Figura 5.21.

- Dentro de la directiva `<VirtualHost> ... </VirtualHost>` se define el comportamiento del servidor virtual por defecto.
- El valor de la directiva `DocumentRoot` es **/var/www/html**.
- El valor de la directiva `ErrorLog`.

3.5. Consulta el fichero **/etc/apache2/apache.conf** y observa, Figura 5.22.

- La directiva contenedora `<Directory> ...</Directory>` que se utiliza para determinar cómo *Apache* sirve el contenido del directorio **/var/www**.

```
<VirtualHost *:80>
        # The ServerName directive sets the request scheme, hostname and port t
        # the server uses to identify itself. This is used when creating
        # redirection URLs. In the context of virtual hosts, the ServerName
        # specifies what hostname must appear in the request's Host: header to
        # match this virtual host. For the default virtual host (this file) thi
        # value is not decisive as it is used as a last resort host regardless.
        # However, you must set it for any further virtual host explicitly.
        #ServerName www.example.com

        ServerAdmin webmaster@localhost
        DocumentRoot /var/www/html

        # Available loglevels: trace8, ..., trace1, debug, info, notice, warn,
        # error, crit, alert, emerg.
        # It is also possible to configure the loglevel for particular
        # modules, e.g.
        #LogLevel info ssl:warn

        ErrorLog ${APACHE_LOG_DIR}/error.log
        CustomLog ${APACHE_LOG_DIR}/access.log combined

        # For most configuration files from conf-available/, which are
        # enabled or disabled at a global level, it is possible to
        # include a line for only one particular virtual host. For example the
```

Figura 5.21: Contenido del fichero **/etc/apache2/sites-avalilabe/default**

```
# access here, or in any related virtual host.
<Directory />
        Options FollowSymLinks
        AllowOverride None
        Require all denied
</Directory>

<Directory /usr/share>
        AllowOverride None
        Require all granted
</Directory>

<Directory /var/www/>
        Options Indexes FollowSymLinks
        AllowOverride None
        Require all granted
</Directory>

#<Directory /srv/>
#       Options Indexes FollowSymLinks
#       AllowOverride None
#       Require all granted
#</Directory>
```

Figura 5.22: Contenido del fichero **/etc/apache2/apache2.conf**

5.4. Ficheros de configuración y directivas en *Windows*

En esta práctica analizaremos los principales ficheros de configuración y el valor de algunas directivas del servidor web *Apache2.2* instalado en la máquina **ServidorW2008XX** o **ServidorW2012XX**.

1. <u>**Ficheros de configuración**</u>

   1.1. Inicia sesión con un usuario con privilegios de administrador en **ServidorW2008XX/2012XX**.

   1.2. Accede al directorio **C:\Program Files\Apache Software Foundation\Apache2.2\conf**.

   1.3. Observa los ficheros de configuración.

   1.4. Abre el fichero **C:\Program Files\Apache Software Foundation\Apache2.2\conf\ httpd.conf** y analiza su configuración.

2. <u>**Directivas**</u>

   2.1. Consulta el fichero **C:\Program Files\Apache Software Foundation\Apache2.2\conf\ httpd.conf**

   2.2. Comprueba cuál es el valor de la directiva `ServerRoot`.

   2.3. Comprueba cuál es el puerto en el que escucha peticiones *Apache* (*puerto 80*), Figura 5.23.

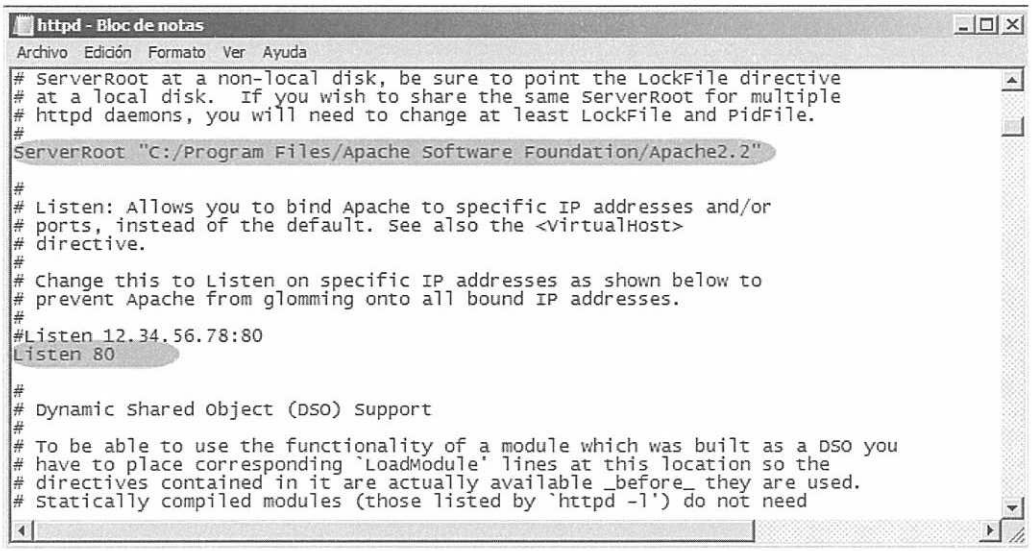


Figura 5.23: *ServerRoot* y *Listen*

2.4. El valor de la directiva `DocumentRoot` es **C:\Program Files\Apache Software Foundation\Apache2.2\htdocs**, Figura 5.24.

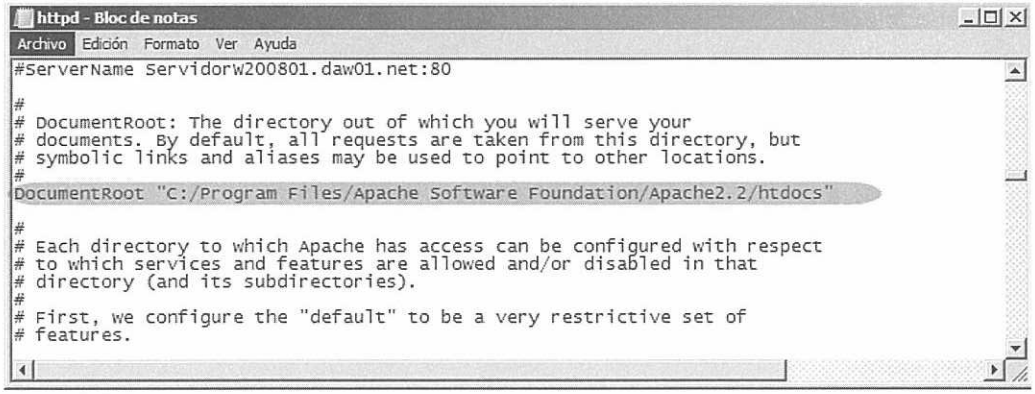


Figura 5.24: *DocumentRoot*

2.5. La directiva contenedora `<Directory> ...</Directory>` que se utiliza para determinar cómo *Apache* sirve el contenido del directorio **C:\Program Files\Apache Software Foundation\Apache2.2\htdocs**, Figura 5.25.

```
# This should be changed to whatever you set DocumentRoot to.
#
<Directory "C:/Program Files/Apache Software Foundation/Apache2.2/htdocs">
    #
    # Possible values for the Options directive are "None", "All",
    # or any combination of:
    #   Indexes Includes FollowSymLinks SymLinksifOwnerMatch ExecCGI MultiViews
    #
    # Note that "MultiViews" must be named *explicitly* --- "Options All"
    # doesn't give it to you.
    #
    # The Options directive is both complicated and important.  Please see
    # http://httpd.apache.org/docs/2.2/mod/core.html#options
    # for more information.
    #
    Options Indexes FollowSymLinks

    #
    # AllowOverride controls what directives may be placed in .htaccess files.
    # It can be "All", "None", or any combination of the keywords:
    #   Options FileInfo AuthConfig Limit
    #
    AllowOverride None

    #
    # Controls who can get stuff from this server.
    #
    Order allow,deny
    Allow from all

</Directory>

#
# DirectoryIndex: sets the file that Apache will serve if a directory
# is requested.
#
<IfModule dir_module>
    DirectoryIndex index.html
</IfModule>
```

Figura 5.25: *Directory*

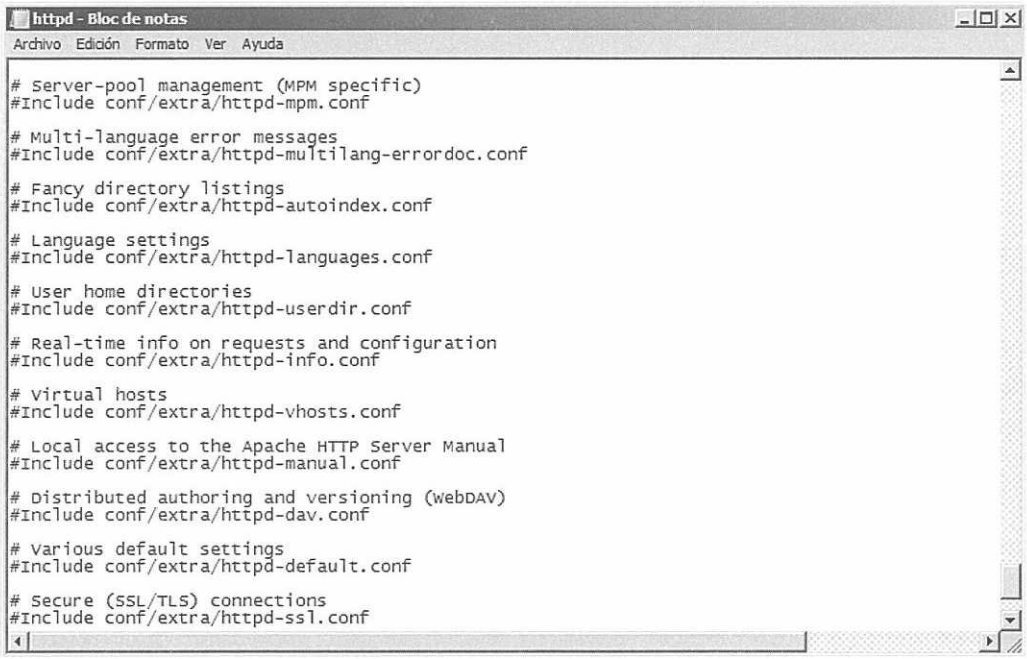
2.6. Observa el valor de la directiva `ErrorLog`, Figura 5.26.

```
# ErrorLog: The location of the error log file.
# If you do not specify an ErrorLog directive within a <VirtualHost>
# container, error messages relating to that virtual host will be
# logged here.  If you *do* define an error logfile for a <VirtualHost>
# container, that host's errors will be logged there and not here.
#
ErrorLog "logs/error.log"

#
# LogLevel: Control the number of messages logged to the error_log.
# Possible values include: debug, info, notice, warn, error, crit,
# alert, emerg.
#
```

Figura 5.26: *ErrorLog*

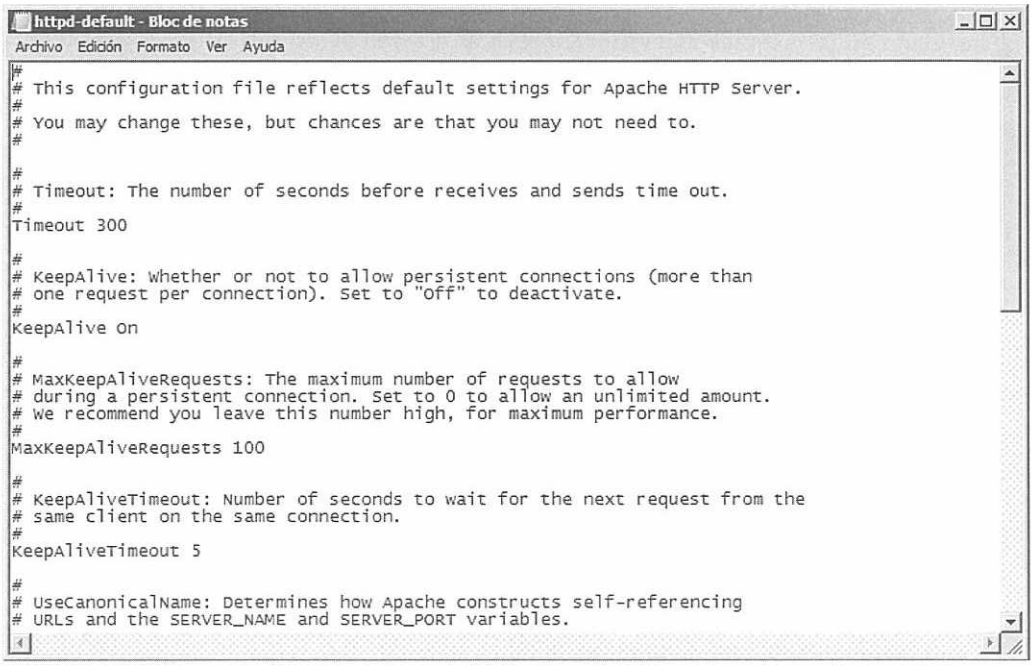
2.7. Observa que existen varias directivas `include` comentadas para incluir nuevas funcionalidades, Figura 5.27.

2.8. Consulta los ficheros **C:\Program Files\Apache Software Foundation\Apache2.2\conf\extra\httpd-default** y **C:\Program Files\Apache Software Foundation\Apache2.2\conf\extra\httpd-vhosts** para ver cómo podrías modificar directivas por defecto y añadir servidores virtuales, Figuras 5.28 y 5.29.



```
# Server-pool management (MPM specific)
#Include conf/extra/httpd-mpm.conf

# Multi-language error messages
#Include conf/extra/httpd-multilang-errordoc.conf

# Fancy directory listings
#Include conf/extra/httpd-autoindex.conf

# Language settings
#Include conf/extra/httpd-languages.conf

# User home directories
#Include conf/extra/httpd-userdir.conf

# Real-time info on requests and configuration
#Include conf/extra/httpd-info.conf

# Virtual hosts
#Include conf/extra/httpd-vhosts.conf

# Local access to the Apache HTTP Server Manual
#Include conf/extra/httpd-manual.conf

# Distributed authoring and versioning (WebDAV)
#Include conf/extra/httpd-dav.conf

# Various default settings
#Include conf/extra/httpd-default.conf

# Secure (SSL/TLS) connections
#Include conf/extra/httpd-ssl.conf
```

Figura 5.27: *Include*



```
#
# This configuration file reflects default settings for Apache HTTP Server.
#
# You may change these, but chances are that you may not need to.
#

#
# Timeout: The number of seconds before receives and sends time out.
#
Timeout 300

#
# KeepAlive: Whether or not to allow persistent connections (more than
# one request per connection). Set to "Off" to deactivate.
#
KeepAlive On

#
# MaxKeepAliveRequests: The maximum number of requests to allow
# during a persistent connection. Set to 0 to allow an unlimited amount.
# We recommend you leave this number high, for maximum performance.
#
MaxKeepAliveRequests 100

#
# KeepAliveTimeout: Number of seconds to wait for the next request from the
# same client on the same connection.
#
KeepAliveTimeout 5

#
# UseCanonicalName: Determines how Apache constructs self-referencing
# URLs and the SERVER_NAME and SERVER_PORT variables.
```

Figura 5.28: Fichero **httpd-default**

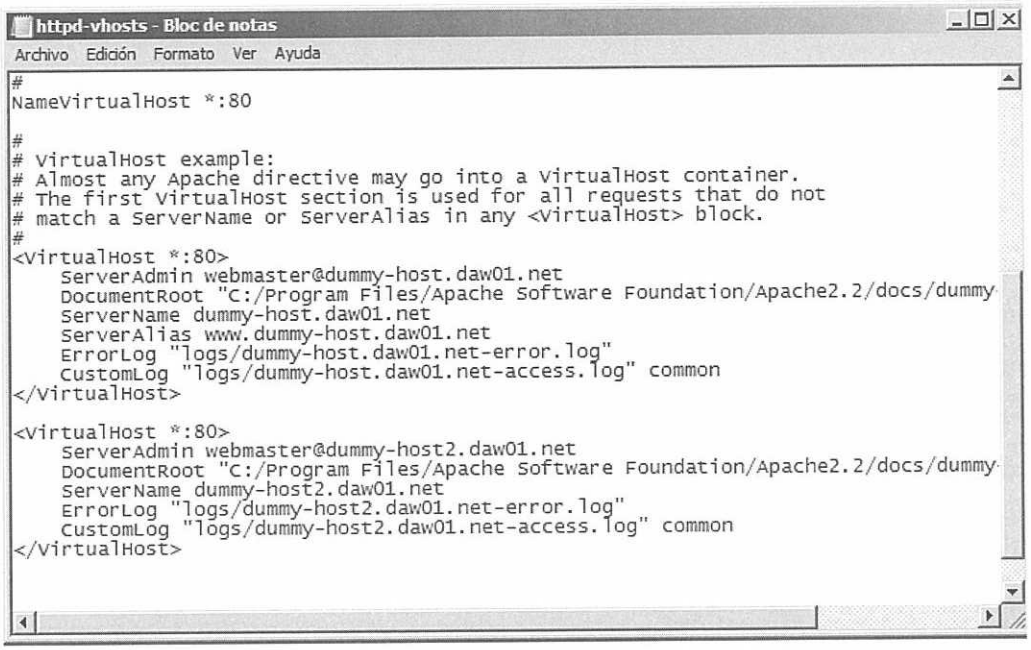


Figura 5.29: Fichero **httpd-vhosts**

5.5. Configuración básica en *Linux*

En esta práctica realizaremos varias pruebas de configuración sobre el servidor web *Apache 2.4* instalado en la máquina **ServidorLinuxXX**. Usaremos las directivas: `DirectoryIndex`, `<Directory>` ... `</Directory>`, `Options Indexes`, `ErrorDocument`, `Alias` y `Redirect`.

1. **Ficheros y directorios de prueba**

   1.1. Inicia sesión con un usuario con privilegios de administrador en **ServidorLinuxXX**.

   1.2. Accede al directorio **/var/www/html** y crea los siguientes ficheros y directorios con el contenido que quieras, Figura 5.30.

   - **/var/www/html/despligue.html**
   - **/var/www/html/fp.html**
   - **/var/www/html/ciclos/listado.html**
   - **/var/www/html/ciclos/asir.html**
   - **/var/www/html/ciclos/daw.html**
   - **/var/www/html/ciclos/dam.html**

   1.3. Desde **DesarrolloW7XX** abre un navegador y establece las siguientes conexiones:

   - `http://192.168.1.X7`
   - `http://192.168.1.X7/despliegue.html`
   - `http://192.168.1.X7/ciclos`
   - `http://192.168.1.X7/ciclos/listado.html`

```
alumno@ServidorLinux01:~$ cd /var/www/html/
alumno@ServidorLinux01:/var/www/html$ ls -l
total 24
drwxr-xr-x 2 root root  4096 jun  8 13:23 ciclos
-rw-r--r-- 1 root root    26 jun  8 13:21 despliegue.html
-rw-r--r-- 1 root root    12 jun  8 13:22 fp.html
-rw-r--r-- 1 root root 11510 jun  8 12:33 index.html
alumno@ServidorLinux01:/var/www/html$ cd ciclos/
alumno@ServidorLinux01:/var/www/html/ciclos$ ls -l
total 16
-rw-r--r-- 1 root root  5 jun  8 13:22 asir.html
-rw-r--r-- 1 root root  4 jun  8 13:22 dam.html
-rw-r--r-- 1 root root  4 jun  8 13:22 daw.html
-rw-r--r-- 1 root root 13 jun  8 13:22 listado.html
alumno@ServidorLinux01:/var/www/html/ciclos$ _
```

Figura 5.30: Ficheros y directorios de prueba

2. **Ficheros a servir por defecto (*Directory Index*)**

   2.1. Desde **DesarrolloW7XX** abre un navegador y establece una conexión a `http://192.168.1.X7`.

   2.2. En la URL no se ha pedido ningún recurso en concreto. El servidor ha enviado por defecto **index.html** (valor de la directiva *DirectoryIndex* por defecto).

   2.3. Renombra el fichero **/var/www/html/index.html** a **/var/www/html/indice.html**.

   2.4. Desde **DesarrolloW7XX** abre un navegador y establece una conexión a `http://192.168.1.X7` (pulsa F5 para refrescar). Como no se ha encontrado **index.html** en el directorio raíz **/var/www/html** se muestra el contenido del directorio, Figura 5.31.

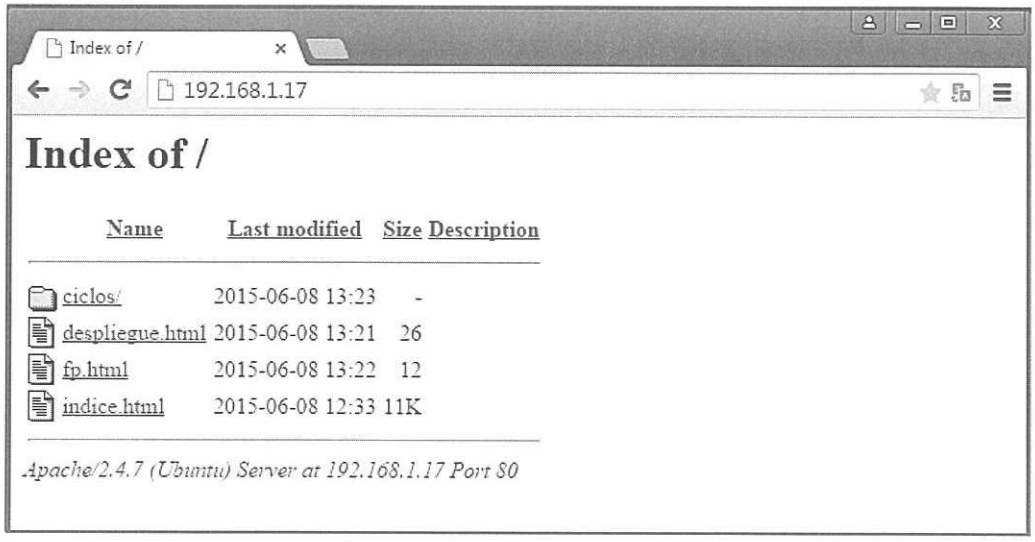


Figura 5.31: Listado del directorio raíz del servidor

2.5. Edita el archivo **/etc/apache2/sites-available/000-default.conf** y añade las siguientes directiva. Sección
`<Directory /var/www/html>` ... `</Directory>` que incluye la directiva `DirectoryIndex` con el valor **despliegue.html** tal y como se muestra en la Figura 5.32.

```
# For most configuration files from conf-available/, which are
# enabled or disabled at a global level, it is possible to
# include a line for only one particular virtual host. For example the
# following line enables the CGI configuration for this host only
# after it has been globally disabled with "a2disconf".
#Include conf-available/serve-cgi-bin.conf

<Directory /var/www/html>
        DirectoryIndex despliegue.html
        Options Indexes FollowSymLinks
        AllowOverride None
        Require all granted
</Directory>

</VirtualHost>
```

Figura 5.32: *DirectoryIndex*

2.6. Reinicia el servidor para que los cambios tengan efecto.

```
sudo service apache2 stop
sudo service apache2 start
```

2.7. Desde **DesarrolloW7XX** abre un navegador y establece una conexión a `http://192.168.1.X7`. Ahora se sirve el fichero **despliegue.html**, Figura 5.33.

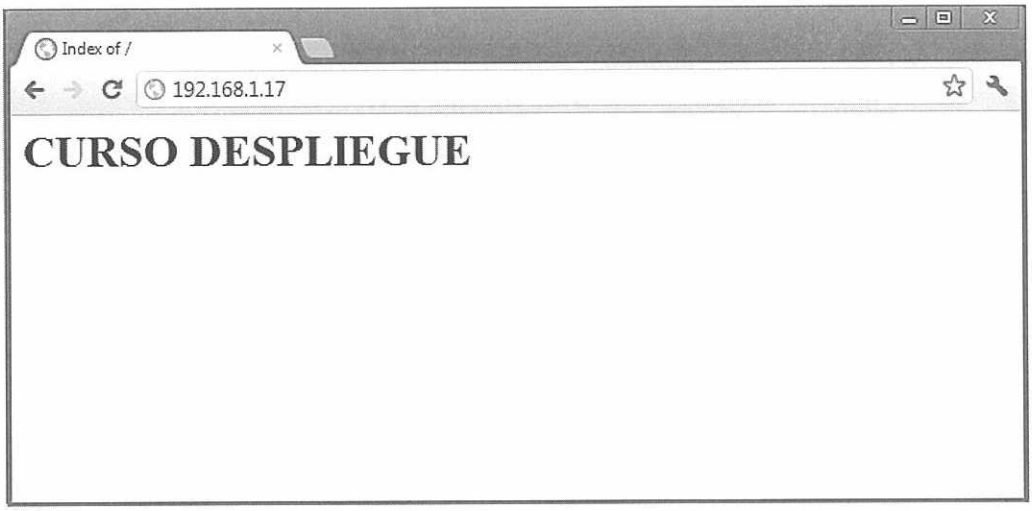


Figura 5.33: Fichero **despliegue.html**

3. **Opciones sobre directorios(*Directory* y *Options Indexes*)**

3.1. Edita el archivo **/etc/apache2/sites-available/000-default.conf** y observa que la directiva `<Directory>` ... `</Directory>` contiene las directivas que determinan cómo

Apache sirve el contenido de ese directorio. Todos los directorios que estén dentro de **/var/www/html** heredan su configuración, y **/var/www** hereda y sobrescribe la configuración del directorio raíz (/) (la configuración del raíz está definida en el fichero apache2.conf). Figura 5.34 .

```
        <Directory /var/www/html>
                DirectoryIndex despliegue.html
                Options Indexes FollowSymLinks
                AllowOverride None
                Require all granted
        </Directory>

</VirtualHost>
```

Figura 5.34: *Directory*

3.2. Desde **DesarrolloW7XX** abre un navegador y establece una conexión a `http://192.168.1.X7/ciclos`. Dentro de **/var/www/html/ciclos** (ha heredado la configuración de **/var/www/html**) no existe el fichero **despliegue.html** (`DirectoryIndex`) y por eso se muestra su contenido.

3.3. Crea una nueva directiva `<Directory> ... </Directory>` para **/var/www/ciclos** tal y como se muestra en la Figura 5.35.

```
<Directory /var/www/html>
        DirectoryIndex despliegue.html
        Options Indexes FollowSymLinks
        AllowOverride None
        Require all granted
</Directory>

<Directory /var/www/html/ciclos>
        Options FollowSymLinks
        AllowOverride None
        Require all granted
</Directory>
```

Figura 5.35: *Directory* y opción *Indexes*

3.4. Observa que para **/var/www/html/ciclos** no se ha definido la opción `Indexes`. Cuando aparece esta opción, el servidor lista el contenido del directorio si no encuentra los ficheros definidos en `DirectoryIndex`, y si no aparece no se muestra el contenido del directorio indicando un mensaje de prohibición.

3.5. Reinicia el servidor para que los cambios tengan efecto.

```
sudo service apache2 stop
sudo service apache2 start
```

3.6. Desde **DesarrolloW7XX** abre un navegador y establece una conexión a `http://192.168.1.X7/`. Se muestra **despliegue.html**.

3.7. Desde **DesarrolloW7XX** abre un navegador y establece una conexión a `http://192.168.1.X7/ciclos`. Como en **/var/www/html/ciclos** no existe **despliegue.html** y no se permite el listado del directorio (`Options Indexes`) el servidor retorna el código **403 Forbbiden**, Figura 5.36.

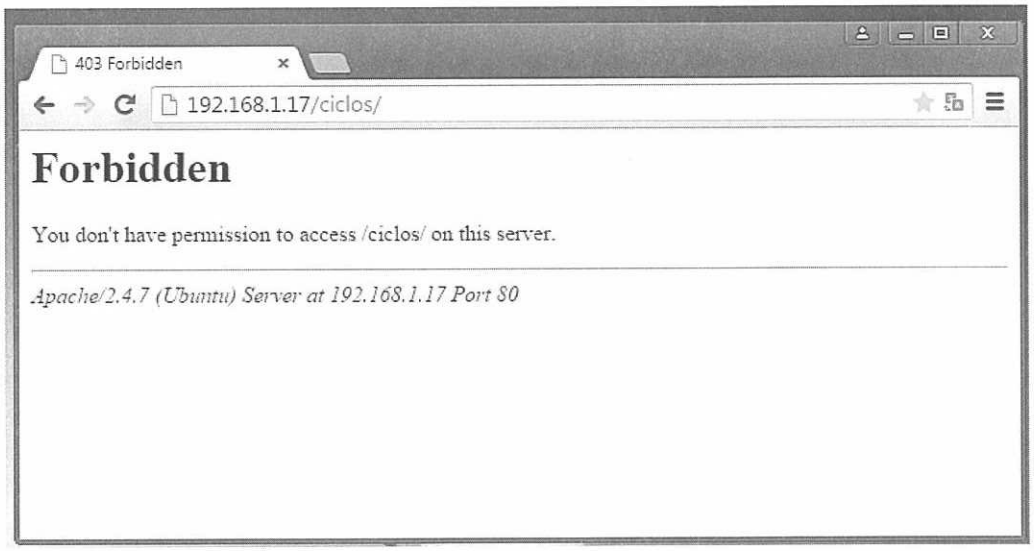


Figura 5.36: *DirectoryIndex* no encontrado en **/var/www/html/ciclos**

3.8. Desde **DesarrolloW7XX** abre un navegador y establece una conexión a `http://192.168.1.X7/ciclos/listado.html` y verifica que es posible acceder, Figura 5.37.

Figura 5.37: Acceso a un fichero dentro del directorio **/var/www/html/ciclos**

4. **Códigos del error (*ErrorDocument*)**

4.1. Configura el servidor virtual por defecto para que cuando retorne el código de error 404 (página no encontrada) envíe el texto "Página no encontrada en el servidor de la red

`dawXX.net`". Edita el archivo **/etc/apache2/sites-available/000-default.conf** tal y como se muestra en la Figura 5.38.

```
        <Directory /var/www/html/ciclos>
                Options FollowSymLinks
                AllowOverride None
                Require all granted
        </Directory>

        ErrorDocument 404 "Página no encontrada en la red daw01.net"

</VirtualHost>
```

Figura 5.38: *ErrorDocument*

4.2. Reinicia el servidor para que los cambios tengan efecto.

4.3. Desde **DesarrolloW7XX** abre un navegador y establece una conexión a `http://192.168.1.X7/noexiste.html`, Figura 5.39.

Figura 5.39: Error 404

4.4. Configura el servidor virtual por defecto para que cuando retorne el código de error 404 (página no encontrada) envíe la página **404.html** almacenada en el directorio raíz del servidor, Figura 5.40. Crea el fichero **/var/www/html/404.html** con el contenido que quieras.

4.5. Reinicia el servidor para que los cambios tengan efecto.

4.6. Desde **DesarrolloW7XX** abre un navegador y establece una conexión a `http://192.168.1.X7/noexiste.html`.

5. **Directorios Virtuales (*Alias*)**

5.1. Crea el directorio **/home/alumno/apuntes**. Dentro del directorio crea un fichero denominado **apuntes.html** con el contenido que qieras.

```
ErrorLog ${APACHE_LOG_DIR}/error.log

# Possible values include: debug, info, notice, warn, error, crit,
# alert, emerg.
LogLevel warn

CustomLog ${APACHE_LOG_DIR}/access.log combined

ErrorDocument 404 /404.html
```

Figura 5.40: Error 404

5.2. Edita el archivo **/etc/apache2/sites-available/000-default.conf**. Utiliza la directiva *Alias* para crear un directorio virtual denominado **/apuntes** que referencie a **/home/alumno/apuntes**. Usa la directiva `Directory` para definir las opciones de configuración del directorio **/home/alumno/apuntes**, Figura 5.41.

```
        <Directory /var/www/html/ciclos>
                Options FollowSymLinks
                AllowOverride None
                Require all granted
        </Directory>

        Alias /apuntes /home/alumno/apuntes
        <Directory /home/alumno/apuntes>
                Options Indexes FollowSymLinks
                AllowOverride None
                Require all granted
        </Directory>

        ErrorDocument 404 /404.html

</VirtualHost>
```

Figura 5.41: Directorios virtuales (*Alias*)

5.3. Reinicia el servidor para que los cambios tengan efecto.

5.4. Desde **DesarrolloW7XX** abre un navegador y establece una conexión a `http://192.168.1.X7/apuntes`, Figura 5.42.

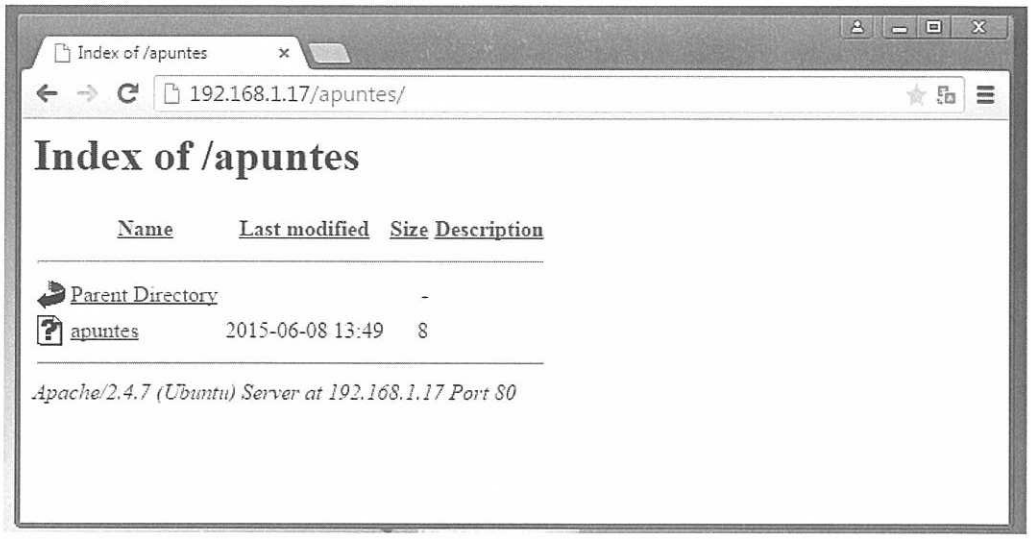


Figura 5.42: Acceso al directorio virtual **apuntes**

6. **Redirecciones (*Redirect*)**

6.1. Edita el archivo **/etc/apache2/sites-available/default**. Utiliza la directiva *Redirect* para crear una redirección de **/fp** hacía a **http://www.todofp.es**, Figura 5.43.

```
Alias /apuntes /home/alumno/apuntes
<Directory /home/alumno/apuntes>
        Options Indexes FollowSymLinks
        AllowOverride None
        Require all granted
</Directory>

Redirect /fp http://www.todofp.es_

ErrorDocument 404 /404.html
```

Figura 5.43: Redirección (*Redirect*)

6.2. Reinicia el servidor para que los cambios tengan efecto.

6.3. Desde **DesarrolloW7XX** abre un navegador y establece una conexión a `http://192.168.1.X7/fp`, Figura 5.44.

Figura 5.44: Redirección de **http://192.168.1.x7/fp** a **http://www.todofp.es**

5.6. Configuración básica en *Windows*

Replica la configuración de *Apache* que has realizado en la práctica anterior sobre el servidor **Apache** de la máquina **ServidorW2008XX** o **ServidorW2012XX**.

1. **Ficheros y directorios de prueba**

   1.1. Inicia sesión con un usuario con privilegios de administrador en **ServidorW2008XX/2012XX**.

   1.2. Accede al directorio **C:\Program Files\Apache Software Foundation\Apache2.2\htdocs** y crea los siguientes ficheros y directorios con el contenido que quieras, Figura 5.45.

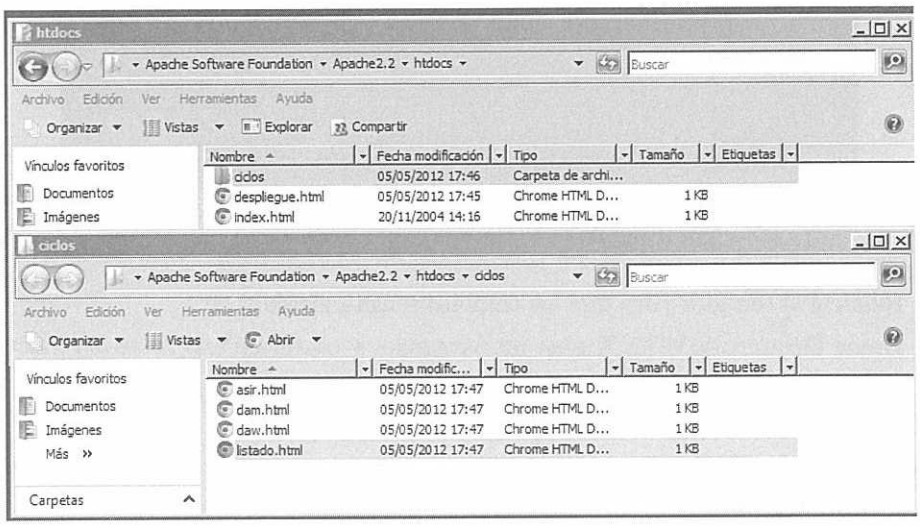


Figura 5.45: Ficheros y directorios de prueba

- **..\htdocs\despligue.html**
- **..\htdocs\fp.html**
- **..\htdocs\ciclos\listado.html**
- **..\htdocs\ciclos\asir.html**
- **..\htdocs\ciclos\daw.html**
- **..\htdocs\ciclos\dam.html**

1.3. Crear el fichero **C:\Usuarios\Administrador\apuntes.html**.

1.4. Desde **DesarrolloW7XX** abre un navegador y establece las siguientes conexiones:

- `http://192.168.1.X8`
- `http://192.168.1.X8/despliegue.html`
- `http://192.168.1.X8/ciclos`
- `http://192.168.1.X8/ciclos/listado.html`

2. Configuración

2.1. Edita el archivo **C:\Program Files\Apache Software Foundation\Apache2.2\conf\ httpd.conf** y crea las directivas necesarias tal y como se muestra en las Figuras 5.48, 5.46 y 5.47.

```
<Directory "C:/Program Files/Apache Software Foundation/Apache2.2/htdocs/ciclos">
    Options FollowSymLinks
    AllowOverride None
    Order allow,deny
    Allow from all
</Directory>
```

Figura 5.46: *Directory*

```
Alias /apuntes C:/Users/Administrador/apuntes
<Directory "C:/Users/Administrador/apuntes">
    Options Indexes FollowSymLinks
    AllowOverride None
    Order allow,deny
    Allow from all
</Directory>

Redirect /fp http://www.todofp.es
```

Figura 5.47: *Alias* y *Redirection*

2.2. Reinicia el servidor para que los cambios tengan efecto, Figura 5.49.

2.3. Desde **DesarrolloW7XX** abre un navegador y establece conexiones con `http://192.168.1.X8` para probar la configuración, Figuras 5.50, 5.51, 5.52 y 5.53.

```
<Directory "C:/Program Files/Apache Software Foundation/Apache2.2/htdocs">
    #
    # Possible values for the Options directive are "None", "All",
    # or any combination of:
    #   Indexes Includes FollowSymLinks SymLinksifOwnerMatch ExecCGI MultiViews
    #
    # Note that "MultiViews" must be named *explicitly* --- "Options All"
    # doesn't give it to you.
    #
    # The Options directive is both complicated and important.  Please see
    # http://httpd.apache.org/docs/2.2/mod/core.html#options
    # for more information.
    #
    DirectoryIndex despliegue.html
    Options Indexes FollowSymLinks

    #
    # AllowOverride controls what directives may be placed in .htaccess files.
    # It can be "All", "None", or any combination of the keywords:
    #   Options FileInfo AuthConfig Limit
    #
    AllowOverride None

    #
    # Controls who can get stuff from this server.
    #
    Order allow,deny
    Allow from all

</Directory>
```

Figura 5.48: *DirectoryIndex*

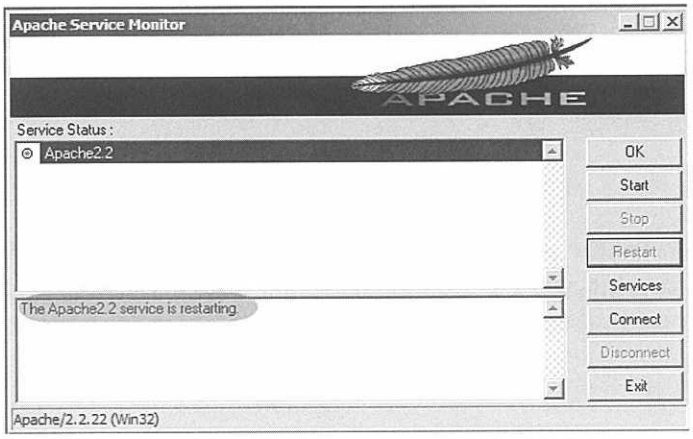


Figura 5.49: Reiniciar el servidor

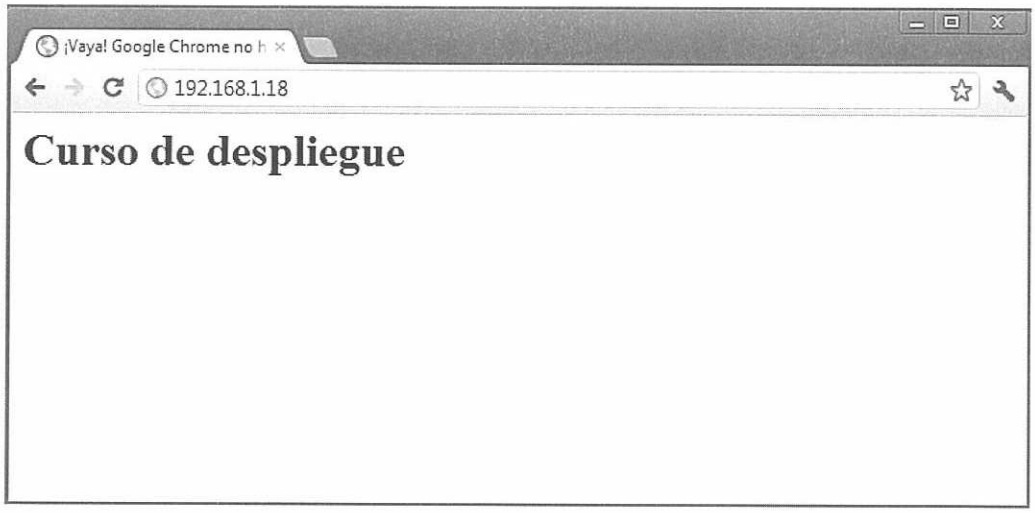


Figura 5.50: Conexión al servidor

Figura 5.51: Conexión al servidor

Figura 5.52: Conexión al servidor

Figura 5.53: Conexión al servidor

5.7. Módulos en *Linux*

Consulta los módulos estáticos y los módulos dinámicos cargados por defecto en la versión de *Apache* instalada en la máquina **ServidorLinuxXX**. Posteriormente, investiga la funcionalidad del módulo *userdir*, habilítalo y prueba su funcionalidad.

1. Módulos

   1.1. Inicia una sesión en **ServidorLinuxXX** con un usuario con privilegios de administrador.

   1.2. Comprueba los módulos estáticos que se han cargado al compilar el servidor ejecutando el siguiente comando, Figura 5.54:

```
sudo apache2ctl -l
```

```
alumno@ServidorLinux01:~$ sudo apachectl -l
Compiled in modules:
  core.c
  mod_so.c
  mod_watchdog.c
  http_core.c
  mod_log_config.c
  mod_logio.c
  mod_version.c
  mod_unixd.c
alumno@ServidorLinux01:~$ _
```

Figura 5.54: Módulos estáticos

   1.3. Comprueba los módulos que se han cargado dinámicamente al arrancar el servidor consultando el directorio **/etc/apache2/mods-enabled**. Observa que los ficheros que aparecen en este directorio son enlaces simbólicos a ficheros de **/etc/apache2/mods-available**.

   1.4. Edita uno de los ficheros **.load** (por ejemplo **dir.load**) y observa cómo se utiliza la directiva `LoadModule`, Figura 5.55 para cargar el módulo. Comprueba cuál es la ruta donde está el código del modulo (archivo **.so**).

```
LoadModule dir_module /usr/lib/apache2/modules/mod_dir.so
```

Figura 5.55: Fichero **dir.load**

   1.5. Edita uno de los ficheros **.conf** (por ejemplo **dir.conf**) y observa cómo se añaden directivas dentro de una declaración `<IfModule nombremodulo> ... </IfModule}>`, Figura 5.56 que se ejecutarán si se carga el módulo.

   1.6. Consulta el directorio **/usr/lib/apache2/modules/** y observa los módulos disponibles para cargar.

```
<IfModule mod_dir.c>
        DirectoryIndex index.html index.cgi index.pl index.php index.xhtml
</IfModule>
```

Figura 5.56: Fichero **dir.conf**

1.7. Ejecuta el siguiente comando para mostrar los paquetes disponibles en los repositorios de *Ubuntu* que permiten instalar módulos adicionales en *Apache*, Figura 5.57.

```
sudo apt-cache search libapache2-mod
```

```
libapache2-mod-apparmor - changehat AppArmor library as an Apache module
libapache2-mod-perl2-doc - Integration of perl with the Apache2 web server -
umentation
libapache2-mod-php5 - server-side, HTML-embedded scripting language (Apache 2
dule)
libapache2-mod-wsgi - Python WSGI adapter module for Apache
php5-cgi - server-side, HTML-embedded scripting language (CGI binary)
libapache2-mod-auth-kerb - modulo apache2 para la autentificación en Kerberos
libapache2-mod-auth-mysql - módulo apache 2 para autenticación MySQL
libapache2-mod-auth-pgsql - Módulo para Apache2 que proporciona autenticación
sql
libapache2-mod-auth-plain - Módulo para Apache2 que provee de autenticación e
exto plano
libapache2-mod-axis2c - Motor de servicios web Apache - módulo de apache
libapache2-mod-macro - Use macros en los archivos de configuración de apache2
libapache2-mod-perl2 - Integración de perl con el servidor web Apache 2
libapache2-mod-perl2-dev - Integración de perl con el servidor web Apache 2 -
chivos de desarrollo
libapache2-mod-python - módulo integrado Python para Apache 2
libapache2-mod-python-doc - Modulo de empotrado Python para Apache 2 - docume
ción
libapache2-mod-fastcgi - módulo FastCGI de Apache 2 para scripts CGI de gran
cucion
libapache2-mod-apreq2 - generic Apache request library - Apache module
libapache2-mod-auth-cas - CAS authentication module for Apache2
```

Figura 5.57: Módulos disponibles en los repositorios de *Ubuntu*

2. <u>Módulo *userdir*</u>

2.1. Comprueba, consultando el directorio **/etc/apache2/mods-enabled** que el módulo *userdir* no está habilitado.

2.2. Habilita el módulo ejecutando el siguiente comando:

```
sudo a2enmod userdir
```

2.3. Verifica que dentro del directorio **/etc/apache2/mods-enabled** se han creado enlaces simbólicos del módulo *userdir* (ficheros **.conf** y **.load**) hacia **/etc/apache2/mod_availables**, Figura 5.58.

2.4. Reinicia el servidor para que los cambios tengan efecto.

```
lrwxrwxrwx 1 root root 30 may  4 23:09 userdir.conf -> ../mods-available/userdir
.conf
lrwxrwxrwx 1 root root 30 may  4 23:09 userdir.load -> ../mods-available/userdir
.load
alumno@ServidorLinux01:/etc/apache2/mods-enabled$
```

Figura 5.58: Módulo **userdir** habilitado

2.5. Consulta el fichero **/etc/apache2/mod_enabled/userdir.conf**. Observa que está habilitado el uso de directorios personales para todos los usuarios excepto para el usuario **root** y que **public_html** es el nombre del subdirectorio que pueden crear los usuarios en su directorio **home** para poner sus páginas personales.

2.6. Crea en directorio **/home/alumno/public_html**. Dentro del directorio crea un fichero denominado **personal.html** con el contenido que quieras.

2.7. Desde **DesarrolloW7XX** accede a `http://192.168.1.X7/~alumno`, Figura 5.59.

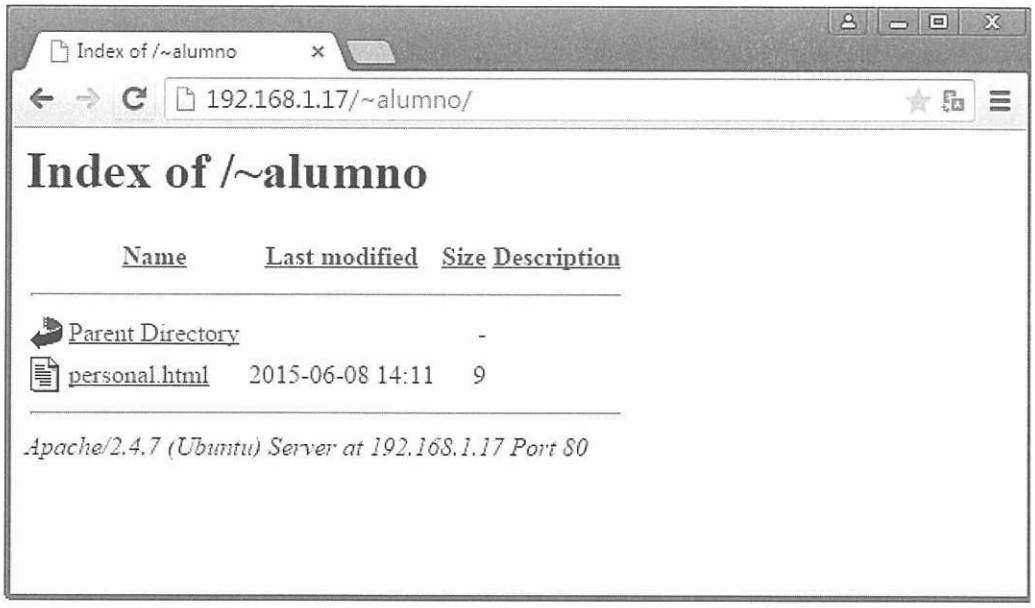


Figura 5.59: Acceso al directorio personal del usuario alumno

2.8. Deshabilita el módulo ejecutando el siguiente comando:

```
sudo a2dismod userdir
```

2.9. Reinicia el servidor para que los cambios tengan efecto.

5.8. Módulos en *Windows*

Consulta los módulos estáticos y los módulos dinámicos cargados por defecto en la versión de *Apache* instalada en la máquina **ServidorW2008XX** o **ServidorW2012XX**. Posteriormente, investiga la funcionalidad del módulo *userdir*, habilítalo y prueba su funcionalidad.

1. Módulos

   1.1. Inicia una sesión en **ServidorW2008XX/2012XX** con un usuario con privilegios de administrador.

   1.2. Comprueba los módulos estáticos que se han cargado al compilar el servidor ejecutando el siguiente comando, Figura 5.60:

   ```
   httpd -l
   ```

   ```
   C:\Users\Administrador>httpd -l
   Compiled in modules:
     core.c
     mod_win32.c
     mpm_winnt.c
     http_core.c
     mod_so.c

   C:\Users\Administrador>_
   ```

   Figura 5.60: Módulos estáticos

   1.3. Comprueba los módulos que se han cargado dinámicamente al arrancar el servidor consultando el fichero **C:\Program Files\Apache Software Foundation\Apache2.2\conf\httpd.conf**.

   1.4. Consulta el directorio **C:\Program Files\Apache Software Foundation\Apache2.2\conf\modules** y observa los módulos disponibles para cargar.

2. Módulo (*userdir*)

   2.1. Edita el fichero **C:\Program Files\Apache Software Foundation\Apache2.2\conf\httpd.conf** y habilita el módulo **userdir** eliminando el comentario de las directivas `LoadModule` e `Include`, Figuras 5.61 y 5.62.

   ```
   #LoadModule speling_module modules/mod_speling.so
   #LoadModule ssl_module modules/mod_ssl.so
   #LoadModule status_module modules/mod_status.so
   #LoadModule substitute_module modules/mod_substitute.so
   #LoadModule unique_id_module modules/mod_unique_id.so
   LoadModule userdir_module modules/mod_userdir.so
   #LoadModule usertrack_module modules/mod_usertrack.so
   #LoadModule version_module modules/mod_version.so
   #LoadModule vhost_alias_module modules/mod_vhost_alias.so
   ```

   Figura 5.61: Habilitar el módulo **userdir**

   ```
   # Fancy directory listings
   #Include conf/extra/httpd-autoindex.conf

   # Language settings
   #Include conf/extra/httpd-languages.conf

   # User home directories
   Include conf/extra/httpd-userdir.conf
   ```

   Figura 5.62: Habilitar directivas del módulo **userdir**

2.2. Edita el fichero **C:\Program Files\Apache Software Foundation\Apache2.2\conf\extras\httpd-userdir.conf**. Observa que está habilitado el uso de directorios personales para todos los usuarios y que **My Documents/My Website** es el nombre del subdirectorio que pueden crear los usuarios para poner sus páginas personales. Edita el fichero y cambia **My Documents/My Website** por **Documents/Website**, Figura 5.63.

```
# Settings for user home directories
#
# Required module: mod_userdir

#
# UserDir: The name of the directory that is appended onto a user's home
# directory if a ~user request is received.  Note that you must also set
# the default access control for these directories, as in the example below.
#
UserDir "Documents/Website"

#
# Control access to UserDir directories.  The following is an example
# for a site where these directories are restricted to read-only.
#
<Directory "C:/Users/*/Documents/Website">
    AllowOverride FileInfo AuthConfig Limit Indexes
    Options MultiViews Indexes SymLinksIfOwnerMatch IncludesNoExec
    <Limit GET POST OPTIONS>
        Order allow,deny
        Allow from all
    </Limit>
    <LimitExcept GET POST OPTIONS>
        Order deny,allow
        Deny from all
    </LimitExcept>
</Directory>
```

Figura 5.63: Fichero **httpd-userdir.conf**

2.3. Reinicia el servidor para que los cambios tengan efecto.

2.4. Crea en directorio **C:\Usuarios\Administrador\Documents\Website**. Dentro del directorio crea un fichero denominado **personal.html** con el contenido que quieras.

2.5. Desde **DesarrolloW7XX** accede a `http://192.168.1.X8/~administrador`, Figura 5.64.

Figura 5.64: Acceso al directorio personal del usuario administrador

2.6. Edita el fichero **C:\Program Files\Apache Software Foundation\Apache2.2\conf\httpd.conf** y deshabilita el módulo **userdir** poniendo el comentario de las directivas `LoadModule` e `Include`.

2.7. Reinicia el servidor para que los cambios tengan efecto.

5.9. Control de acceso por IP y nombre de dominio

En la máquina **ServidorLinuxXX** crea el directorio **/var/www/html/profesor** y configura *Apache* para que solo se pueda acceder desde el equipo local y desde **DesarrolloW7XX** (192.168.1.X6).

1. Inicia una sesión en **ServidorLinuxXX** con un usuario con privilegios de administrador.
2. Crea en directorio **/var/www/html/profesor**. Dentro del directorio crea un fichero denominado **profesor.html** con el contenido que quieras.
3. Edita el fichero de configuración **/etc/apache2/sites-available/000-default.conf** y utiliza la directiva `<Directory>` junto con la directivas `Require` para denegar el acceso al directorio a todos los equipos excepto al local y a **DesarrolloW7XX**, Figura 5.65.

```
<Directory /var/www/html/profesor>
        Options Indexes FollowSymLinks
        AllowOverride None
        Require ip 127.0.0.1
        Require ip 192.168.1.16
</Directory>
```

Figura 5.65: Control de acceso por IP y nombre de dominio

4. Reinicia el servidor para que los cambios tengan efecto.
5. Comprueba que se puede acceder a `http://192.168.1.X7/profesor/` desde **DesarrolloW7XX** pero no desde la máquina real, Figuras 5.66 y 5.67.

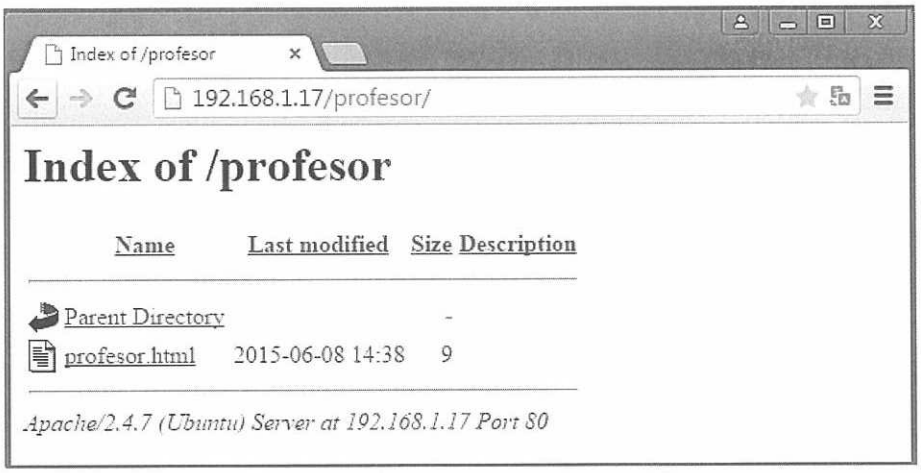


Figura 5.66: Conexión desde **DesarrolloW7XX**

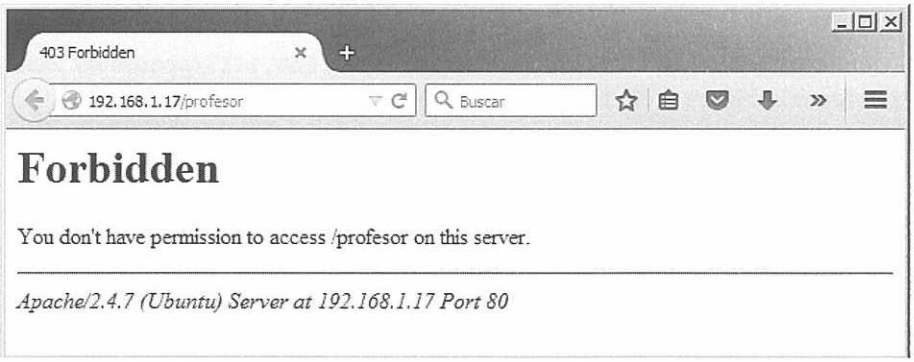


Figura 5.67: Conexión desde la máquina real

6. Prueba a permitir/denegar el acceso desde las máquinas de tus compañeros.

5.10. Autenticación y autorización *Basic* y *Digest*

En la máquina **ServidorLinuxXX** configura la autenticación HTTP *Basic* sobre el directorio **/var/www/html/profesor** para que solo puedan acceder los usuarios **profesor1** y **profesor2**. Configura la autenticación HTTP *Digest* sobre el directorio **/var/www/html/departamento** para que solo puedan acceder los usuarios **admin1** y **admin2**.

1. **Autenticación HTTP *Basic***

   1.1. Inicia una sesión en **ServidorLinuxXX** con un usuario con privilegios de administrador.

   1.2. Comprueba, consultando el directorio **/etc/apache2/mods-enabled**, que el módulo ***auth_basic*** está habilitado.

   1.3. Para usar la autenticación *basic* hay que crear un fichero accesible por *Apache* en el que se guardarán los usuarios y sus contraseñas. Para crear este fichero se utilizará el comando `httpaswd` (`http://httpd.apache.org/docs/2.4/programs/htpasswd.html`).

   a Instala el paquete **apache2-utils** que contiene **htpasswd**.

   ```
   sudo apt-get install apache2-utils
   ```

   b Crea el fichero y añade el usuario **profesor1** (la opción -c es para crear el fichero).

   ```
   sudo htpasswd -c /etc/apache2/passwd profesor1
   ```

   c Añade el usuario **profesor2** (no se usa la opción -c porque el fichero ya existe).

   ```
   sudo htpasswd /etc/apache2/passwd profesor2
   ```

   1.4. Edita el fichero de configuración **/etc/apache2/sites-available/000-default.conf** y permite el acceso a directorio **/var/www/html/profesor** a los usuarios **profesor1** y **profesor1**, es necesario utilizar las directivas `<RequireAll>` y `<RequireAny>` para controlar cuáles de las directivas `Require` queremos que se cumplan. Figura 5.68.

1.5. Reinicia el servidor para que los cambios tengan efecto.

1.6. Desde **DesarrolloW7XX** accede a `http://192.168.1.X7/profesor/` con el usuario **profesor1**, Figura 5.69. Intenta el acceso con otro usuario ¿Es posible?

```
<Directory /var/www/html/profesor>
        Options Indexes FollowSymLinks
        AllowOverride None
        AuthType Basic
        AuthName "Acceso restringido"
        AuthUserFile /etc/apache2/passwd
        <RequireALL>
                Require user profesor1 profesor2
                <RequireAny>
                        Require ip 127.0.0.1
                        Require ip 192.168.1.16
                </RequireAny>
        </RequireALL>
</Directory>
```

Figura 5.68: Autenticación *Basic*

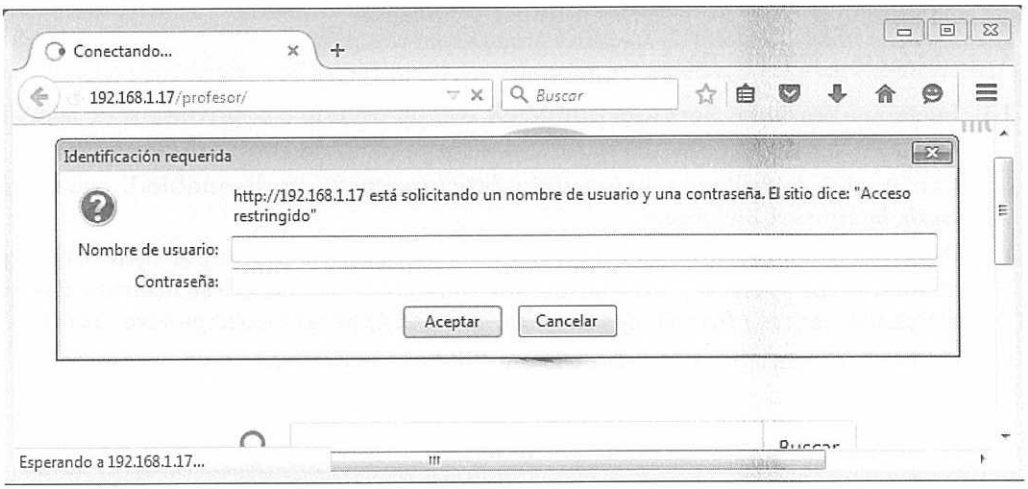


Figura 5.69: Conexión desde **DesarrrolloW7XX**

2. **Autenticación HTTP *Digest***

2.1. Crea en directorio **/var/www/html/departamento**. Dentro del directorio crea un fichero denominado **departamento.html** con el contenido que quieras.

2.2. Habilita el módulo ***auth_digest***.

```
sudo a2enmod auth_digest
```

2.3. Reinicia el servidor para que los cambios tengan efecto.

2.4. Para usar la autenticación *digest* hay que crear un fichero accesible por *Apache* en el que se guardarán los usuarios y sus contraseñas asociados a un dominio (*realm*). Para crear este fichero se utilizará el comando `htdigest` (`http://httpd.apache.org/docs/2.4/es/programs/htdigest.html`).

a Crea el fichero y añade el usuario **admin1** al dominio `informatica` (la opción -c es para crear el fichero).

```
sudo htdigest -c /etc/apache2/digest informatica admin1
```

b Añade el usuario **admin2** (no se usa la opción -c porque el fichero ya existe).

```
sudo htdigest /etc/apache2/digest informatica admin2
```

2.5. Edita el fichero de configuración **/etc/apache2/sites-available/000-default.conf** y permite el acceso a directorio **/var/www/html/departamento** a los usuarios **admin1** y **admin2**, Figura 5.70.

```
<Directory /var/www/html/departamento>
        Options Indexes FollowSymLinks
        AllowOverride None
        AuthType Digest
        AuthName "informatica"
        AuthDigestProvider file
        AuthUserFile /etc/apache2/digest
        Require user admin1 admin2
</Directory>
```

Figura 5.70: Autenticación *Digest*

2.6. Desde **DesarrolloW7XX** accede a `http://192.168.1.X7/departamento/` con el usuario **admin1**, Figura 5.71. Intenta el acceso con otro usuario ¿Es posible?

5.11. Ficheros .htaccess

En la máquina **ServidorLinuxXX**

- Habilita en *Apache* el uso de ficheros de configuración personalizada de directorios (**.htaccess**) en el directorio **/home/profesor/blog** para que sea el propio usuario **profesor** el que pueda controlar como sirve *Apache* los contenidos de ese directorio.
- Configura el alias **/blog** que permita acceder al directorio **/home/profesor/blog**.
- Como usuario **profesor** haz uso del fichero **.htaccess** y configura sobre el directorio **/home/profesor/blog**.
  - Que solo se pueda acceder desde **DesarrolloW7XX**.
  - Autenticación HTTP *Digest* para que solo pueda acceder el usuario **blog**.

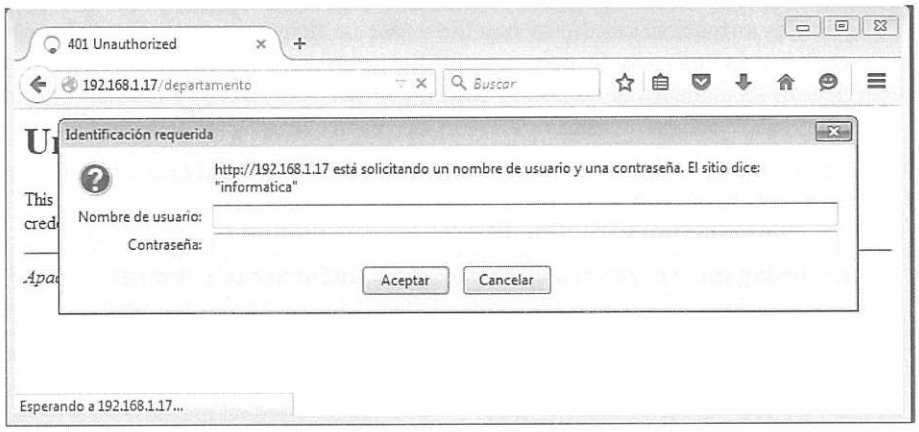


Figura 5.71: Conexión desde **DesarrrolloW7XX**

1. Inicia una sesión en **ServidorLinuxXX** con un usuario con privilegios de administrador.
2. Crea el usuario **profesor**.

   ```
   sudo adduser profesor
   ```

3. Edita el fichero de configuración **/etc/apache2/sites-available/000-default.conf**. Crea el alias y habilita el uso de ficheros **.htaccess** permitiendo sobrescribir todas las directivas en el directorio **/home/profesor/blog**. Elimina las directivas anteriores y añade la directiva `AllowOverride All`, Figura 5.72.

   ```
   Alias /blog /home/profesor/blog
   <Directory /home/profesor/blog>
           AllowOverride All
   </Directory>
   ```

Figura 5.72: Permitir el uso de ficheros **.htaccess**

4. Reinicia el servidor para que los cambios tengan efecto.
5. Inicia una sesión en **ServidorLinuxXX** como usuario **profesor**.
6. Crea el directorio **/home/profesor/blog/**. Crea dentro el fichero **blog.html** con el contenido que quieras.
7. Crea el fichero **/home/profesor/blog/.htdigest** y añade al usuario **blog**.

   ```
   htdigest -c /home/profesor/.htdigest informatica blog
   ```

8. Crea el fichero **/home/profesor/blog/.htaccess** y añade las directivas para realizar la configuración pedida (no es necesario incluir la directiva `<Directory>` porque el fichero ya está en el directorio en el que se aplicará su configuración), Figura 5.73.
9. Desde **DesarrolloW7XX** accede a `http://192.168.1.X7/blog` para probar la configuración, Figuras 5.74 y 5.75.

```
Options Indexes
AuthType Digest
AuthName "informatica"
AuthDigestProvider file
AuthUserFile /home/profesor/blog/.htdigest
Require user blog
```

Figura 5.73: Fichero **.htaccess**

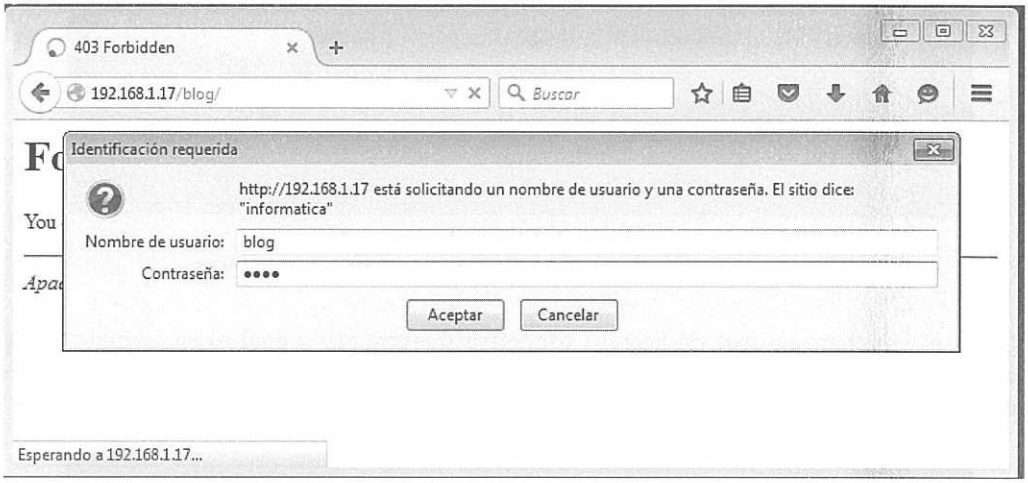


Figura 5.74: Conexión desde **DesarrolloW7XX**

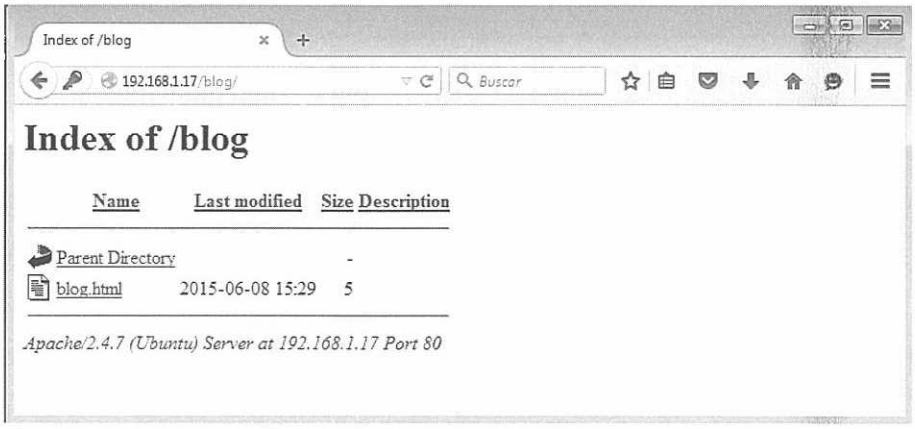


Figura 5.75: Conexión desde **DesarrolloW7XX**

5.12. Ficheros de registros (*logs*)

Sobre la máquina **ServidorLinuxXX** consulta los ficheros de configuración de *Apache* y observa las directivas utilizadas para definir la configuración de los ficheros de registros (logs). Posteriormente consulta estos ficheros.

1. Inicia una sesión en **ServidorLinuxXX** con un usuario con privilegios de administrador.
2. Consulta el archivo **/etc/apache2/sites-available/000-default.conf**.
3. ¿Cuál es el fichero de *logs* de errores (directiva `ErrorLog`) y cuál es su nivel de prioridad (directiva `LogLevel`)?, Figura 5.76.

```
# Available loglevels: trace8, ..., trace1, debug, info, notice, warn,
# error, crit, alert, emerg.
# It is also possible to configure the loglevel for particular
# modules, e.g.
#LogLevel info ssl:warn

ErrorLog ${APACHE_LOG_DIR}/error.log
CustomLog ${APACHE_LOG_DIR}/access.log combined
```

Figura 5.76: Configuración del fichero de *logs* de errores

4. ¿Cuál es el fichero de *logs* de accesos (directiva `CustomLog`) y cuál es su formato (como no especifica ningún formato con `LogFormat` se usa el definido para el servidor principal en el fichero **/etc/apache2/apache2.conf**)?, Figuras 5.77 y 5.78.

```
# Available loglevels: trace8, ..., trace1, debug, info, notice, warn,
# error, crit, alert, emerg.
# It is also possible to configure the loglevel for particular
# modules, e.g.
#LogLevel info ssl:warn

ErrorLog ${APACHE_LOG_DIR}/error.log
CustomLog ${APACHE_LOG_DIR}/access.log combined
```

Figura 5.77: Configuración del fichero de *logs* de accesos

5. Consulta el *log* de errores **/var/log/apache2/error.log**.
6. Consulta el *log* de accesos **/var/log/apache2/access.log**.

5.13. Módulos *mod_status* y *mod_info*

Sobre la máquina **ServidorLinuxXX** prueba la funcionalidad de los módulos *mod_status* y *mod_info*.

1. *mod_status*

   Módulo que permite monitorizar el rendimiento del servidor *Apache*. Genera un documento en HTML con información sobre el estado actual del servidor.

```
#
# The following directives define some format nicknames for use with
# a CustomLog directive.
#
# These deviate from the Common Log Format definitions in that they use %O
# (the actual bytes sent including headers) instead of %b (the size of the
# requested file), because the latter makes it impossible to detect partial
# requests.
#
# Note that the use of %{X-Forwarded-For}i instead of %h is not recommended.
# Use mod_remoteip instead.
#
LogFormat "%v:%p %h %l %u %t \"%r\" %>s %O \"%{Referer}i\" \"%{User-Agent}i\"" $
LogFormat "%h %l %u %t \"%r\" %>s %O \"%{Referer}i\" \"%{User-Agent}i\"" combin$
LogFormat "%h %l %u %t \"%r\" %>s %O" common
LogFormat "%{Referer}i -> %U" referer
LogFormat "%{User-agent}i" agent
```

Figura 5.78: Configuración del fichero de *logs* de accesos
(Fichero **/etc/apache2/apache2.conf**)

1.1. Inicia una sesión en **ServidorLinuxXX** con un usuario con privilegios de administración.

1.2. Habilita el módulo si no está habilitado.

```
sudo a2enmod status
```

1.3. Edita el fichero de configuración del módulo **/etc/apache2/mods-enabled/status.conf** y habilita el acceso a **/server-status** desde **DesarrolloW7XX**, Figura 5.79.

```
<Location /server-status>
        SetHandler server-status
        Require local
        Require ip 192.168.1.16
        #Require ip 192.0.2.0/24
</Location>
```

Figura 5.79: Fichero de configuración **status.conf**

1.4. Reinicia el servidor para aplicar los cambios.

1.5. Desde **DesarrolloW7XX** conéctate a `http://192.168.1.X7/server-status`, Figura 5.80.

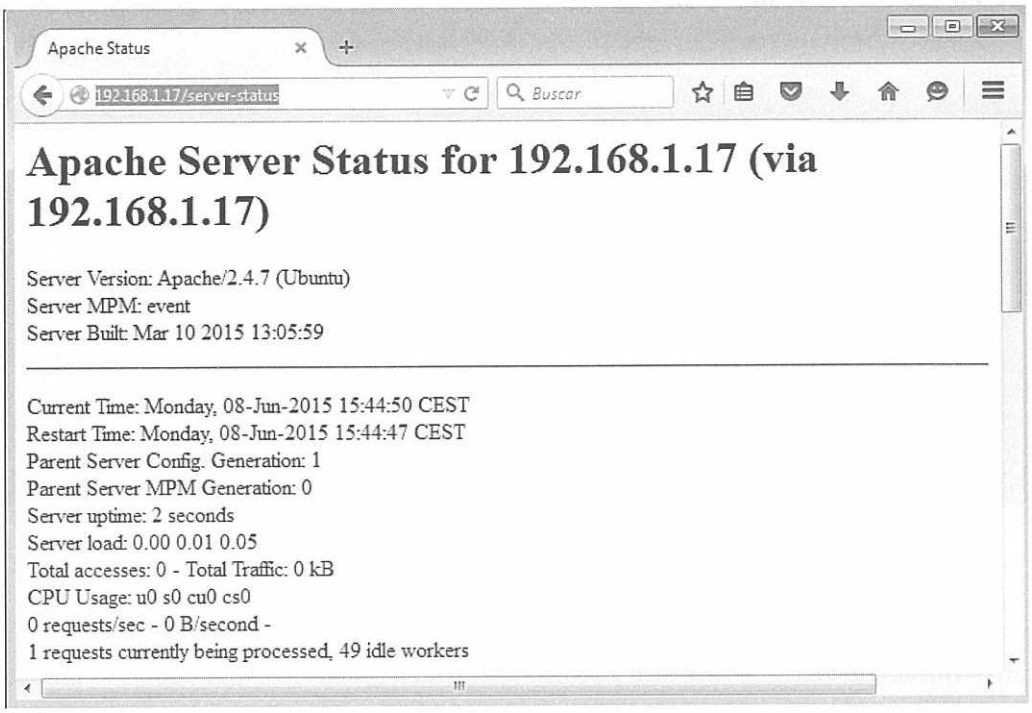


Figura 5.80: Estado del servidor

1.6. También es posible consultar la información desde un terminal ejecutando el siguiente comando.

```
sudo apache2ctl status
```

2. *mod_info*

Módulo que proporciona una vista resumida de la configuración del servidor.

2.1. Abre un terminal y habilita el módulo.

```
sudo a2enmod info
```

2.2. Edita el fichero de configuración del módulo **/etc/apache2/mods-enabled/info.conf** y habilita el acceso a **/server-info** desde **DesarrolloW7XX**, Figura 5.81.

```
<Location /server-info>
        SetHandler server-info
        Require local
        Require ip 192.168.1.16
        #Require ip 192.0.2.0/24
</Location>
```

Figura 5.81: Fichero de configuración **info.conf**

2.3. Reinicia el servidor para aplicar los cambios.

2.4. Desde **DesarrolloW7XX** conéctate a `http://192.168.1.X7/server-info`, Figura 5.82.

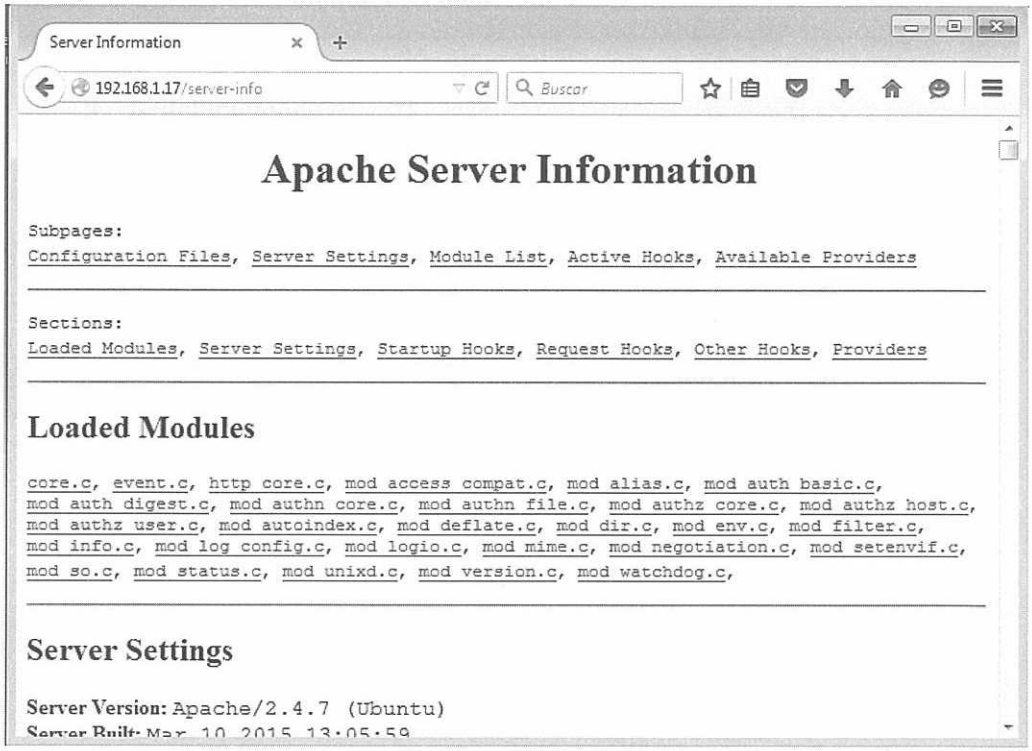


Figura 5.82: Información sobre la configuración del servidor

5.14. *Webalizer*

Sobre la máquina **ServidorLinuxXX** instala y prueba la funcionalidad de *Webalizer*.

1. Inicia una sesión en **ServidorLinuxXX** con un usuario con privilegios de administración.
2. Abre un terminal e instala el software.

```
sudo apt-get update
sudo apt-get install webalizer
```

3. Consulta el fichero de configuración **/etc/webalizer/webzalizer.conf** y observa que se analizará el fichero de *logs* de accesos del servidor virtual por defecto. Quita la extensión .1, e indica **/var/www/html/webalizer** como directorio de salida, Figura 5.83.
4. Crea el directorio **/var/www/html/webalizer**.
5. Lanza el programa para que lea el fichero de *log* y genere el documento html con las estadísticas, Figura 5.84.

```
sudo webalizer
```

6. Desde **DesarrolloW7XX** conéctate a `http://192.168.1.X7/webalizer/index.html`, Figura 5.85.

```
LogFile /var/log/apache2/access.log

# LogType defines the log type being processed.  Normally, the Webalizer
# expects a CLF or Combined web server log as input.  Using this option,
# you can process ftp logs (xferlog as produced by wu-ftp and others),
# Squid native logs or W3C extended format web logs. Values can be 'clf',
# 'ftp', 'squid' or 'w3c'.  The default is 'clf'.

#LogType        clf

# OutputDir is where you want to put the output files.  This should
# should be a full path name, however relative ones might work as well.
# If no output directory is specified, the current directory will be used.

OutputDir /var/www/html/webalizer
```

Figura 5.83: Fichero **/etc/webalizer/webzalizer.conf**

```
alumno@ServidorLinux01:/var/www/webalizer$ sudo webalizer
Webalizer V2.23-08 (Linux 3.16.0-30-generic x86_64) locale: UTF-8
Utilizando histórico /var/log/apache2/access.log (clf)
Creando informe en /var/www/html/webalizer
El nombre de máquina en el informe es 'ServidorLinux01.daw01.net'
Leyendo archivo... webalizer.hist
Generando informe de Junio 2015
Guardando información de archivo...
Generando informe resumido
312 registros en 1 segundos, 312/sec
alumno@ServidorLinux01:/var/www/webalizer$
```

Figura 5.84: Ejecución de *webalizer*

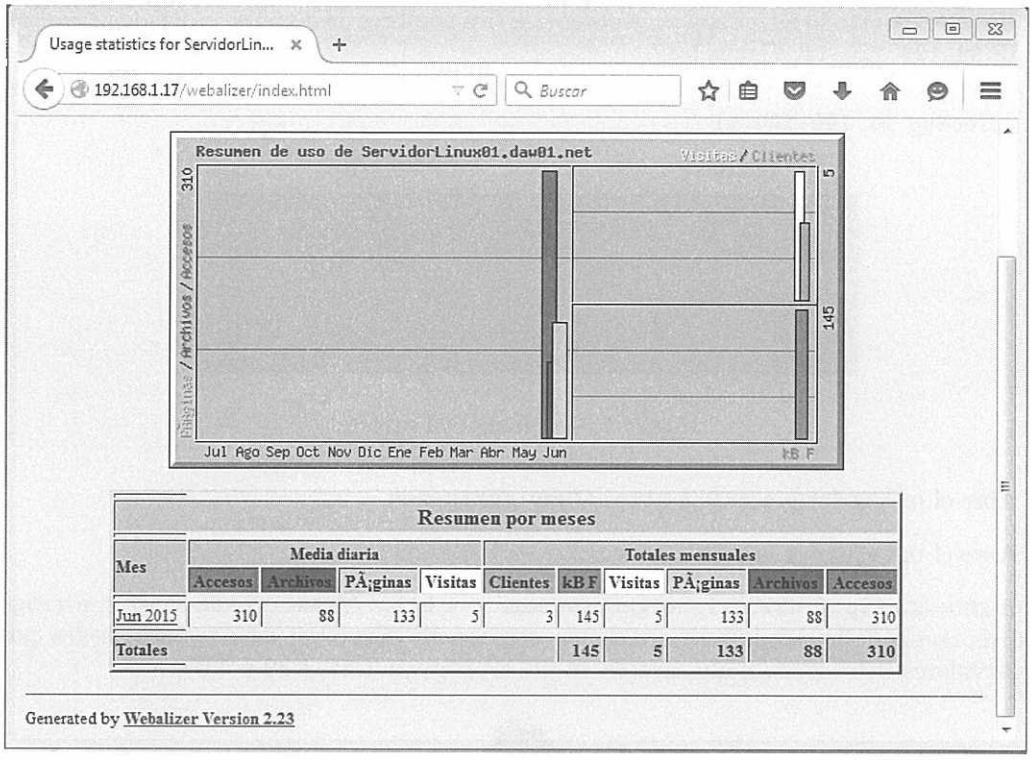


Figura 5.85: Estadísticas de *webalizer*

5.15. Alojamiento virtual de sitios web en Internet

Consulta las siguientes webs `http://www.myipneighbors.com/` y `http://www.robtex.com/` y averigua información sobre nombres de dominio alojados en la misma IP.

1. Inicia una sesión en **DesarrolloW7XX** con un usuario con privilegios de administración.

2. Utiliza el comando `nslookup` para obtener las direcciones IP asociadas al nombre de dominio `www.agsfp.net`, Figura 5.86.

```
nslookup www.agsfp.net
```

Figura 5.86: Resolución directa

3. Utiliza el comando `nslookup` para obtener el/los nombres de dominio asociados a la dirección IP `91.199.120.11`, Figura 5.87.

```
nslookup 91.199.120.11
```

```
C:\Users\alumno>nslookup 91.199.120.11
Servidor:  servidorw200801.daw01.net
Address:  192.168.1.18

Nombre:  ela.h3m.com
Address:  91.199.120.11
```

Figura 5.87: Resolución inversa

4. Abre el navegador y accede a `http://www.agsfp.net`.
5. Abre el navegador y accede a `http://91.199.120.11`.
6. Accede a `http://www.myipneighbors.com/` y a `http://www.robtex.com/` y averiguga qué más nombres de dominio se relacionan con esa IP. Esto nos da una idea de los posibles servidores web virtuales que existen en esa IP, Figuras 5.88, 5.89 y 5.90.

Figura 5.88: http://www.myipneighbors.com

www.agsfp.net has one IP number (91.199.120.11) , which is the same as for agsfp.net, via cname , but the reverse is ela.h3m.com. Miedoalaverdad.com, webdeportes.es, muguruza.es, igalpel.com, gatsrocafort.com and at least 200 other hosts point to the same IP. Disparatefarmaceutico.com, ibaplicaciones.com, hermandaddesantiago.org, treshormigas.es, tutecnico.com and at least ten other hosts use www.agsfp.net as a mail server under another name.

Figura 5.89: http://www.robtex.com/

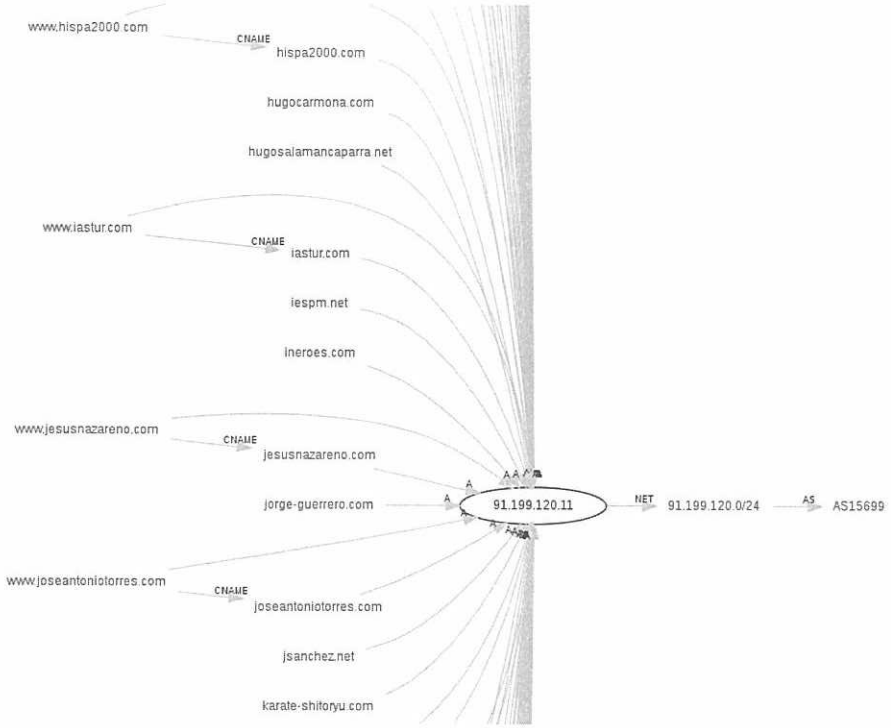


Figura 5.90: http://www.robtex.com/

7. Busca otros sitios web que ofrezcan el mismo servicio.

5.16. Alojamiento virtual de sitios web en *Linux*

Realiza la siguiente configuración en el servidor *Apache* instalado en **ServidorLinuxXX**.

- Configura el servidor DNS de **ServidorW2008XX** o **ServidorW2012XX** para que resuelva los nombres `software.dawXX.net` y `hardware.dawXX.net`.
- Deshabilita el servidor virtual por defecto.
- Crea y habilita un servidor virtual para el dominio `software.dawXX.net`.
  - Directorio raíz **/var/www/html/software**.
    - Se servirá el fichero **index.html** si no se indica ningún fichero en la URL.
    - No se mostrará un listado del directorio raíz si no se solicita ningún fichero.
    - Podrán acceder todos los usuarios.
  - Directorio **/var/www/software/privado**.
    - Solo podrá acceder el usuario **linux** desde el equipo `192.168.1.X6` (**DesarrolloW7XX**).
    - Se mostrará un listado del directorio raíz si no se solicita ningún fichero.
  - El *log* de errores será **/var/log/apache2/software.error.log**.
  - El *log* de accesos será **/var/log/apache2/software.access.log**, con formato *combined*.

- Crea y habilita un servidor virtual para el dominio `hardware.dawXX.net`.
  - Directorio raíz **/var/www/html/hardware**.
    - Se servirá el fichero **index.html** si no se indica ningún fichero en la URL.
    - Se mostrará un listado del directorio raíz si no se solicita ningún fichero.
    - Podrán acceder todos los usuarios.
  - Se creará el alias **/alumno** que permitirá acceder al directorio virtual **/home/alumno**.
    - Se permitirá sobrescribir todas las directivas haciendo uso de ficheros **.htaccess** en **/home/alumno**.
  - El *log* de errores será **/var/log/apache2/hardware.error.log**.
  - El *log* de accesos será **/var/log/apache2/hardware.access.log**, que tendrá formato *combined*.

1. **Configuración del servidor DNS**

   1.1. Configura el servidor DNS de **ServidorW2008XX** o **ServidorW2012XX** en para que resuelva los nombres `software.dawXX.net` y `hardware.dawXX.net`. La dirección IP asociada a los nombres será la IP de **ServidoLinuxXX** es decir `192.168.1.X7`, Figura 5.91.

   1.2. Asegúrate que **DesarrolloW7XX** utiliza el servidor DNS que has configurado, Figura 5.92.

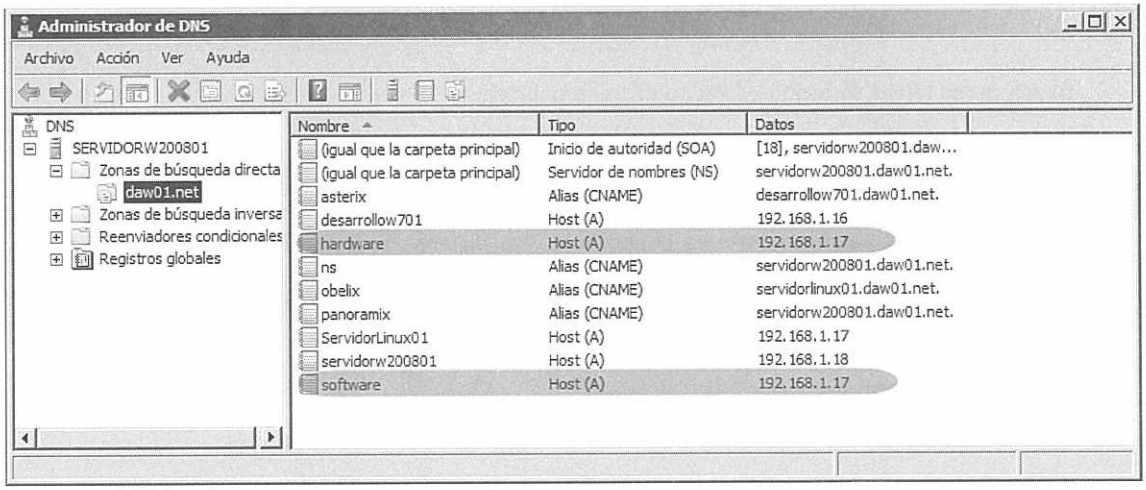


Figura 5.91: Configuración del servidor DNS en **ServidorW2008XX**

2. **Deshabilitar el servidor virtual por defecto**

   2.1. Deshabilita el servidor virtual por defecto.

```
sudo a2dissite 000-default
```

   2.2. Verifica que dentro del directorio **/etc/apache2/sites-enabled** se ha borrado el fichero **000-default.conf**.

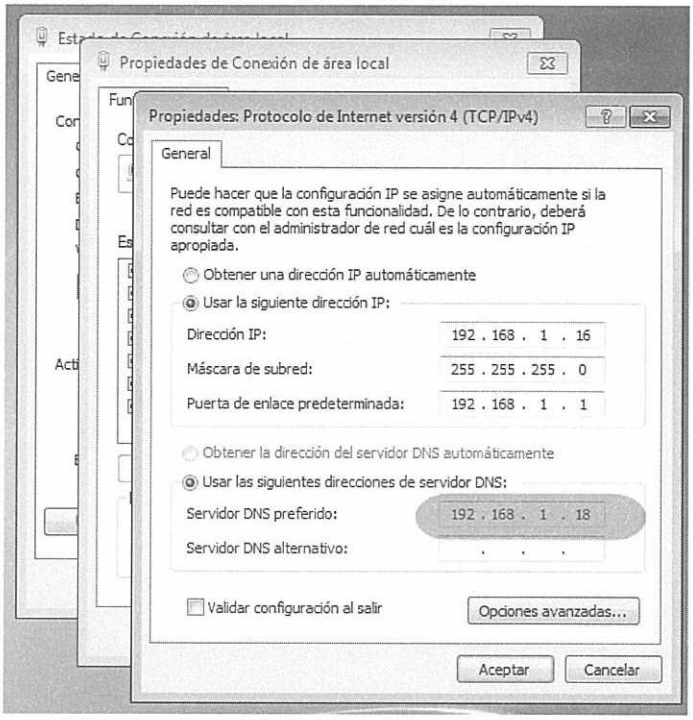


Figura 5.92: Servidor DNS en **DesarrolloW7XX**

2.3. Reinicia el servidor para que los cambios tengan efecto. Observa que se muestra un mensaje que indica que no hay servidores virtuales configurados.

3. **Servidor virtual para el dominio** `software.dawXX.net`

3.1. Crea el directorio **/var/www/html/software**.

3.2. Crea el fichero de texto **/var/www/html/software/index.html** con contenido que quieras.

3.3. Crea el directorio **/var/www/html/software/privado** .

3.4. Crea el fichero de texto **/var/www/html/software/privado/privado.html** con contenido que quieras.

3.5. Crea el fichero **/etc/apache2/software.digest** y añade el usuario **linux** al dominio software (la opción -c es para crear el fichero).

```
sudo htdigest -c /etc/apache2/software.digest software linux
```

3.6. Crea el fichero **/etc/apache/site-available/software.conf** con las siguientes directivas, Figura 5.93.

3.7. Habilita el servidor virtual de software.

```
sudo a2ensite software
```

3.8. Verifica que dentro del directorio **/etc/apache2/sites-enabled** se ha creado el enlace **software.conf**, Figura 5.94.

```
<VirtualHost *:80>
        ServerName software.daw01.net

        DocumentRoot /var/www/html/software

        ErrorLog ${APACHE_LOG_DIR}/software.error.log
        CustomLog ${APACHE_LOG_DIR}/software.access.log combined


        <Directory /var/www/html/software>
                DirectoryIndex index.html
                Options Indexes FollowSymLinks
                AllowOverride None
                Require all granted
        </Directory>

        <Directory /var/www/html/software/privado>
                Options Indexes FollowSymLinks
                AllowOverride None
                AuthType Digest
                AuthName "software"
                AuthDigestProvider file
                AuthUserFile /etc/apache2/software.digest
                <RequireALL>
                        Require user linux
                        Require ip 192.168.1.16
                </RequireALL>
        </Directory>

</VirtualHost>
```

Figura 5.93: Fichero de configuración del servidor virtual software.daw01.net

```
alumno@ServidorLinux01:/etc/apache2/sites-enabled$ ls -l
total 0
lrwxrwxrwx 1 root root 32 jun  8 16:37 software.conf -> ../sites-available/softw
are.conf
```

Figura 5.94: Directorio **/etc/apache2/sites-enabled**

3.9. Reinicia el servidor para que los cambios tengan efecto.

3.10. Desde **DesarrolloWindosXX** accede a `http://software.dawXX.net`, `http://software.dawXX.net/privado`, Figuras 5.95 y 5.96.

Figura 5.95: Acceso al servidor virtual software.daw01.net

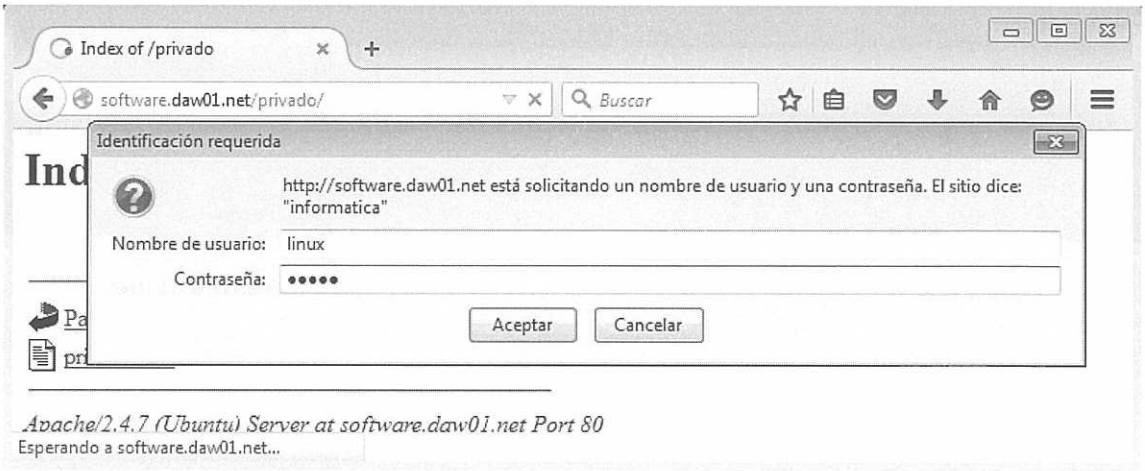


Figura 5.96: Acceso al servidor virtual **software.daw01.net**

4. **Servidor virtual para el dominio `hardware.dawXX.net`**

4.1. Crea el directorio **/var/www/html/hardware**.

4.2. Crea el fichero de texto **/var/www/html/hardware/index.html** con contenido que quieras.

4.3. Crea el fichero **/etc/apache/site-available/hardware.conf** con las siguientes directivas, Figura 5.97.

4.4. Habilita el servidor virtual de hardware.

```
sudo a2ensite hardware
```

```
<VirtualHost *:80>
        ServerName hardware.daw01.net

        DocumentRoot /var/www/html/hardware

        ErrorLog ${APACHE_LOG_DIR}/hardware.error.log
        CustomLog ${APACHE_LOG_DIR}/hardware.access.log combined

        <Directory /var/www/html/hardware>
                DirectoryIndex index.html
                Options Indexes FollowSymLinks
                AllowOverride None
                Require all granted
        </Directory>

        Alias /alumno /home/alumno
        <Directory /var/www/html/software/privado>
                AllowOverride All
        </Directory>

</VirtualHost>
```

Figura 5.97: Fichero de configuración del servidor virtual **hardware.daw01.net**

```
alumno@ServidorLinux01:/etc/apache2/sites-enabled$ ls -l
total 0
lrwxrwxrwx 1 root root 32 jun  8 16:49 hardware.conf -> ../sites-available/hardware.conf
lrwxrwxrwx 1 root root 32 jun  8 16:37 software.conf -> ../sites-available/software.conf
alumno@ServidorLinux01:/etc/apache2/sites-enabled$
```

Figura 5.98: Directorio **/etc/apache2/sites-enabled**

4.5. Verifica que dentro del directorio **/etc/apache2/sites-enabled** se ha creado el enlace **hardware.conf**, Figura 5.98.

4.6. Reinicia el servidor para que los cambios tengan efecto.

4.7. Desde **DesarrolloW7XX** accede a `http://hardware.dawXX.net`, Figura 5.99.

Figura 5.99: Acceso al servidor virtual **hardware.daw01.net**

5.17. Alojamiento virtual de sitios web en *Windows*

Realiza la siguiente configuración en el servidor *Apache* instalado en **ServidorW2008XX**.

- Configura el servidor DNS de **ServidorW2008XX** para que resuelva los nombres `www.tierra.com` y `www.marte.com`.
- Crea y habilita un servidor virtual para el dominio `www.tierra.com`.
  - Directorio raíz **C:\Program Files\Apache Software Foundation\Apache2.2\htdocs\tierra**.
    - Se servirá el fichero **index.html** si no se indica ningún fichero en la URL.
    - No se mostrará un listado del directorio raíz si no se solicita ningún fichero.
    - Podrán acceder todos los usuarios.
  - El *log* de errores será **C:\Program Files\Apache Software Foundation\Apache2.2\logs\tierra.error.log**.
  - El *log* de accesos será **C:\Program Files\Apache Software Foundation\Apache2.2\logs\tierra.access.log**, con formato *combined*.
- Crea y habilita un servidor virtual para el dominio `www.marte.com`.
  - Directorio raíz **C:\Program Files\Apache Software Foundation\Apache2.2\htdocs\marte**.
    - Se servirá el fichero **index.html** si no se indica ningún fichero en la URL.
    - Se mostrará un listado del directorio raíz si no se solicita ningún fichero.

  - Podrán acceder todos los usuarios.
- El *log* de errores será **C:\Program Files\Apache Software Foundation\Apache2.2\logs\marte.error.log**
- El *log* de accesos será **C:\Program Files\Apache Software Foundation\Apache2.2\logs\marte.access.log**, con formato *combined*.

1. **Configuración del servidor DNS**

   1.1. Configura el servidor DNS de **ServidorW2008XX** para que resuelva los nombres `www.marte.com` y `www.tierra.com`. La dirección IP asociada a los nombres será la IP de **ServidoWindowsXX** es decir `192.168.1.X8`, Figuras 5.100 y 5.101.

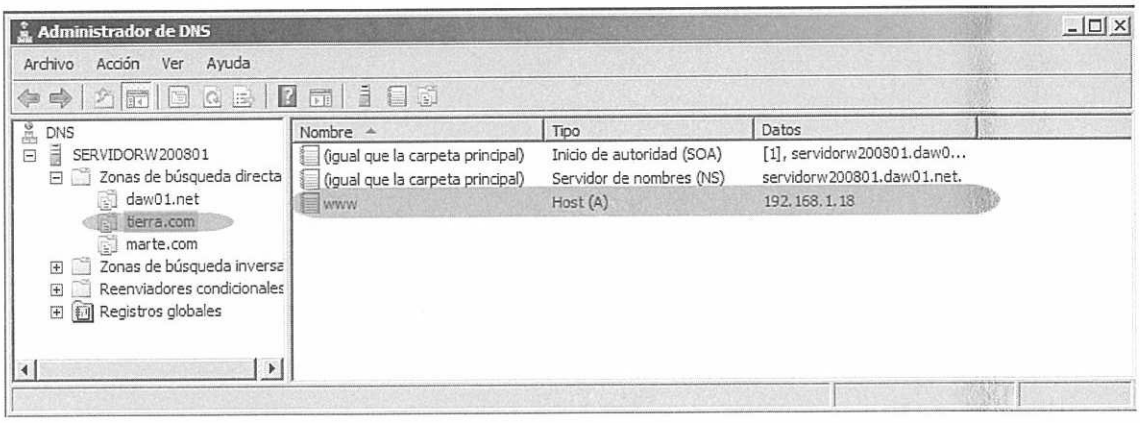


Figura 5.100: Configuración del servidor DNS en **ServidorW2008XX**

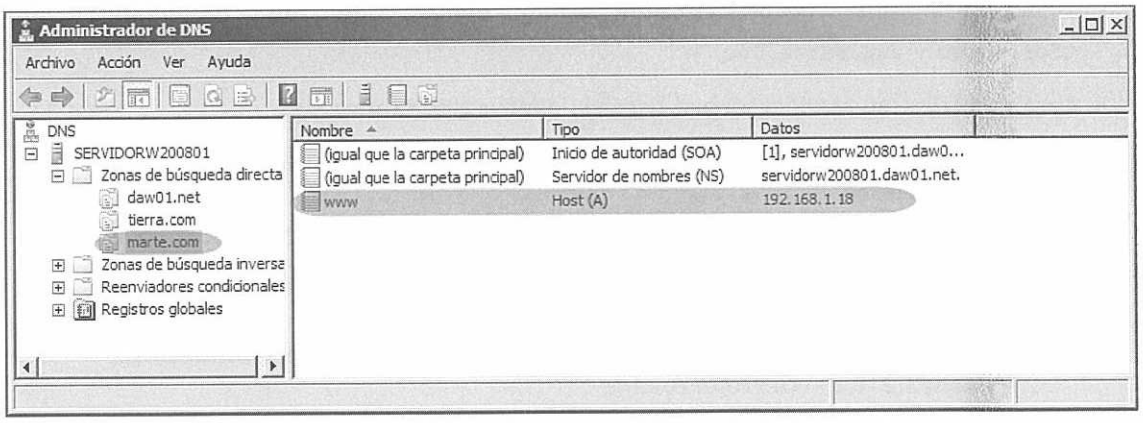


Figura 5.101: Configuración del servidor DNS en **ServidorW2008XX**

   1.2. Asegúrate que **DesarrolloW7XX** utiliza el servidor DNS que has configurado.

2. Configuración de servidores virtuales

2.1. Edita el fichero **C:\Program Files\Apache Software Foundation\Apache2.2\conf\httpd.conf** y eliminana el comentario de la directiva `Include` del fichero **conf/extra/httpd-vhost.conf**, Figura 5.102.

```
# Real-time info on requests and configuration
#Include conf/extra/httpd-info.conf

# Virtual hosts
Include conf/extra/httpd-vhosts.conf

# Local access to the Apache HTTP Server Manual
#Include conf/extra/httpd-manual.conf

# Distributed authoring and versioning (WebDAV)
#Include conf/extra/httpd-dav.conf

# Various default settings
#Include conf/extra/httpd-default.conf
```

Figura 5.102: Fichero httpd.conf

2.2. Crea el directorio **C:\Program Files\Apache Software Foundation\Apache2.2\htdocs\tierra**.

2.3. Crea el fichero de texto **C:\Program Files\Apache Software Foundation\Apache2.2\htdocs\tierra\index.html** con contenido que quieras.

2.4. Crea el directorio **C:\Program Files\Apache Software Foundation\Apache2.2\htdocs\marte**.

2.5. Crea el fichero de texto **C:\Program Files\Apache Software Foundation\Apache2.2\htdocs\marte\marte.html** con contenido que quieras.

2.6. Edita el fichero **C:\Program Files\Apache Software Foundation\Apache2.2\conf\extras\httpd-vhost.conf** y configura las siguientes directivas, Figura 5.103.

2.7. Reinicia el servidor para que los cambios tengan efecto.

2.8. Desde **DesarrolloW7XX** accede a `http://www.tierra.com` y `http://www.marte-com`, Figuras 5.104 y 5.105.

```
#
# Use name-based virtual hosting.
#
NameVirtualHost *:80

#
# VirtualHost example:
# Almost any Apache directive may go into a VirtualHost container.
# The first VirtualHost section is used for all requests that do not
# match a ServerName or ServerAlias in any <VirtualHost> block.
#
<VirtualHost *:80>

    ServerName www.tierra.com
    DocumentRoot "C:/Program Files/Apache Software Foundation/Apache2.2/htdocs/tierra"

   <Directory "C:/Program Files/Apache Software Foundation/Apache2.2/htdocs/tierra">
        DirectoryIndex index.html
        Options Indexes FollowSymLinks
        AllowOverride None
        Order allow,deny
        Allow from all
    </Directory>

    ErrorLog "logs/tierra.error.log"
    CustomLog "logs/tierra.access.log" common

</VirtualHost>

<VirtualHost *:80>
    ServerName www.marte.com
    DocumentRoot "C:/Program Files/Apache Software Foundation/Apache2.2/htdocs/marte"

   <Directory "C:/Program Files/Apache Software Foundation/Apache2.2/htdocs/marte">
        DirectoryIndex index.html
        Options Indexes FollowSymLinks
        AllowOverride None
        Order allow,deny
        Allow from all
    </Directory>

    ErrorLog "logs/marte.error.log"
    CustomLog "logs/marte.access.log" common

</VirtualHost>
```

Figura 5.103: Fichero de configuración de los servidores virtuales

Figura 5.104: Acceso al servidor virtual **www.tierra.com**

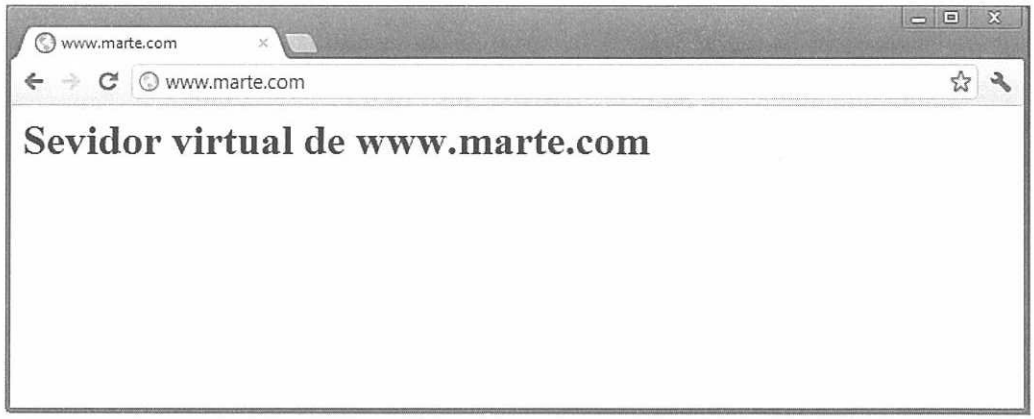


Figura 5.105: Acceso al servidor virtual **www.marte.com**

5.18. HTTPS y certificados digitales

1. Certificado digital verificado

   1.1. Inicia sesión en **DesarrolloW7XX**.

   1.2. Inicia *Firefox*.

   1.3. Conéctate a `https://www.bbva.es`.

   1.4. Observa en la URL que el protocolo usado es https.

   1.5. Pincha en la parte izquierda de la URL.

   1.6. Pincha sobre **Más información** para consular el certificado digital que ha enviado el servidor web y responde de a las siguientes preguntas, Figura 5.106.

   a ¿Qué algoritmo de clave simétrica se ha utilizado para cifrar la información que viaja por la red? *AES* ¿Cuál es la longitud de la clave utilizada? *128 bits*.

   b ¿Cuál es el periodo de validez del certificado? *Del 18/08/2014 al 14/08/2015*.

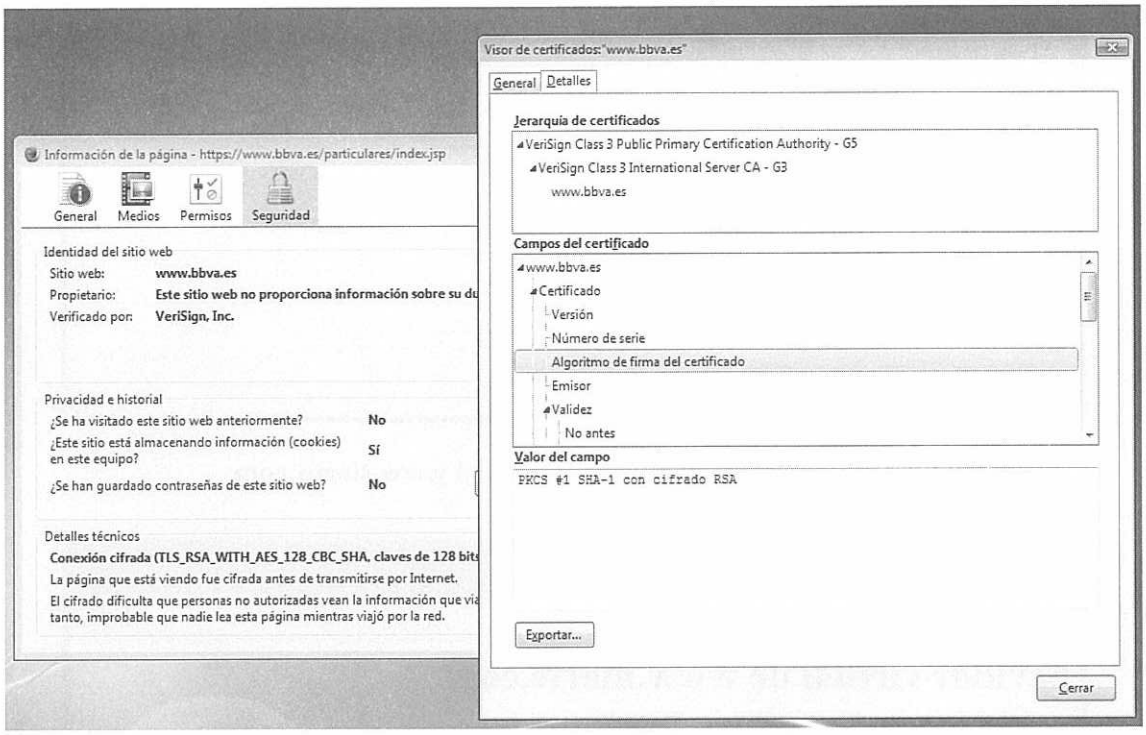


Figura 5.106: Certificado digital

c ¿Qué función resumen (hash) ha utilizado la autoridad de certificación para firmar el certificado? *SHA1*.

d ¿Qué algoritmo de clave asimétrica ha utilizado la autoridad de certificación para firmar el certificado? *RSA*.

e ¿De qué tamaño es la clave pública del certificado? *2048 bits*.

f ¿Qué autoridad de certificación ha firmado el certificado? *VeriSign Class 3 International Server CA - G3* ¿De quién depende? *VeriSign Class 3 Public Primary Certification Authority - G5*.

1.7. En el menú de *Firefox* accede a **Opciones, Opciones, Pestaña Avanzado, Pestaña Cifrado, Ver certificados** y busca el certificado de la autoridad certificadora que ha firmado el certificado, Figura 5.107.

2. <u>Certificado no verificado</u>

2.1. Inicia Firefox.

2.2. Conéctate a la url que indique el profesor.

2.3. El navegador muestra un mensaje de error indicando que no ha podido verificar el certificado que le ha enviado el servidor web, Figura 5.108.

2.4. Pincha en **Entiendo los riesgos**.

2.5. Pincha en **Añadir Excepción**, Figura 5.109. Observa que está marcada la opción **Guardar excepción de forma permanente**.

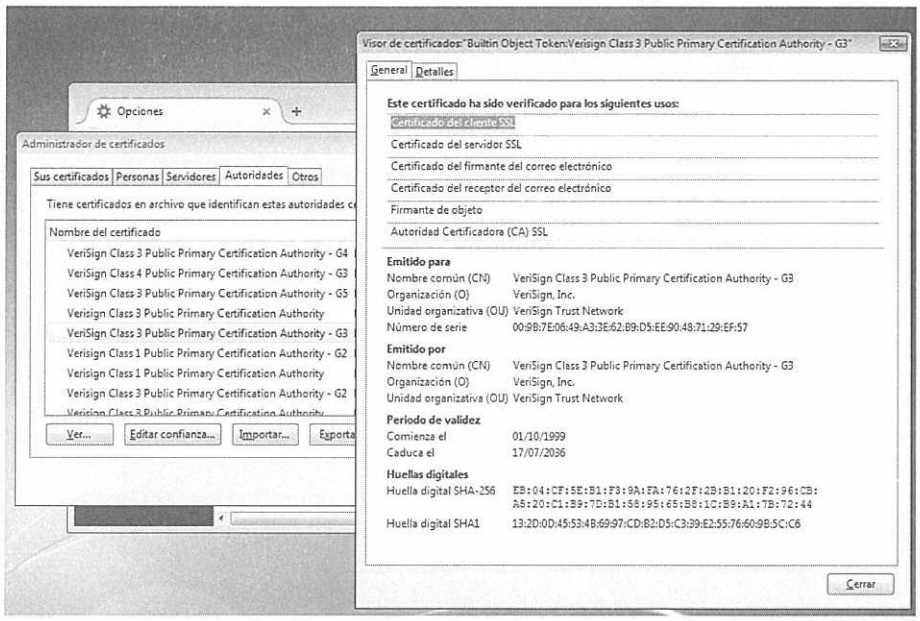


Figura 5.107: Certificado digital de la autoridad de certificación

Figura 5.108: Aviso de certificado digital no verificado

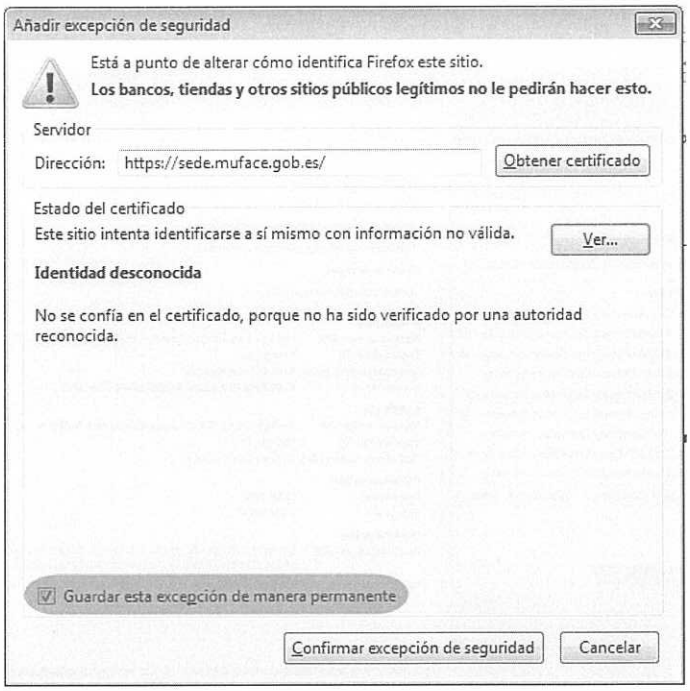


Figura 5.109: Ver el certificado digital no verificado

2.6. Pincha en **Obtener certificado** y en **Ver** para mostrar los datos del certificado digital que ha enviado el navegador.

2.7. Pincha en confirmar excepción de seguridad.

2.8. En el menú de Firefox accede a **Opciones, Opciones, pestaña Avanzado, Pestaña Cifrado, Ver certificados** busca el certificado del servidor que has aceptado y elimínalo, Figura 5.110.

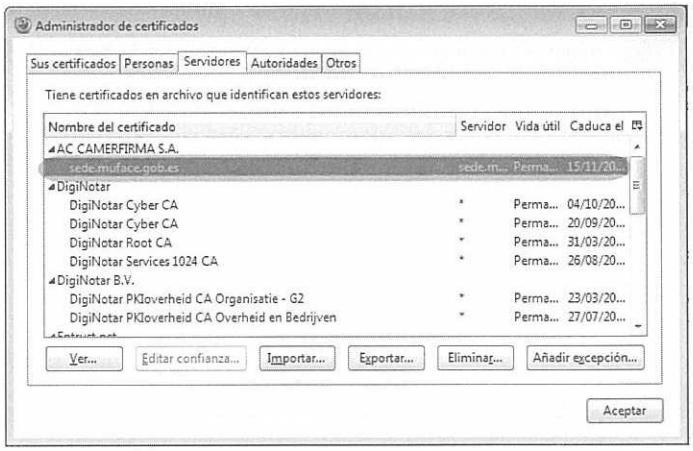


Figura 5.110: Eliminar certificado del servidor

5.19. Servidor virtual HTTPS por defecto en *Linux*

Realiza la siguiente configuración en el servidor *Apache* instalado en **ServidorLinuxXX**.

- Habilita el **servidor virtual por defecto**.
- Deshabilita los **servidores virtuales** creados en las prácticas anteriores.
- Habilita el modulo *mod_ssl*.
- Habilita el servidor virtual *ssl* por defecto.

Prueba la configuración.

1. Inicia una sesión en **ServidorLinuxXX** con un usuario con privilegios de administración.
2. Habilita el servidor virtual por defecto de *Apache*.

```
sudo a2ensite 000-default
```

3. Verifica que dentro del directorio **/etc/apache2/sites-enabled** se ha creado el enlace **000-default.conf**.
4. Deshabilita los servidores virtuales creados en prácticas anteriores.

```
sudo a2dissite software
sudo a2dissite hardware
```

5. Reinicia el servidor para que los cambios tengan efecto.
6. Habilita el módulo *modssl* que permite usar *https*, Figura 5.111.

```
sudo a2enmod ssl
```

```
alumno@ServidorLinux01:/etc/apache2$ sudo a2enmod ssl
[sudo] password for alumno:
Considering dependency setenvif for ssl:
Module setenvif already enabled
Considering dependency mime for ssl:
Module mime already enabled
Considering dependency socache_shmcb for ssl:
Module socache_shmcb already enabled
Enabling module ssl.
See /usr/share/doc/apache2/README.Debian.gz on how to configure SSL and create s
elf-signed certificates.
To activate the new configuration, you need to run:
  service apache2 restart
alumno@ServidorLinux01:/etc/apache2$
```

Figura 5.111: Habilitar el modulo modssl

7. Reinicia el servidor para que los cambios tengan efecto.

```
# If you just change the port or add more ports here, you will likely also
# have to change the VirtualHost statement in
# /etc/apache2/sites-enabled/000-default.conf

Listen 80

<IfModule ssl_module>
        Listen 443
</IfModule>

<IfModule mod_gnutls.c>
        Listen 443
</IfModule>

# vim: syntax=apache ts=4 sw=4 sts=4 sr noet
```

Figura 5.112: Fichero **/etc/apache2/port.conf**

8. Consulta el fichero **/etc/apache2/port.conf** y observa que si habilita el modulo *ssl* el servidor escuchará en el puerto 443, Figura 5.112.

9. Verifica que el servidor escucha en los puertos 80/TCP y 443/TCP.

   ```
   netstat -ltn
   ```

10. Accede al directorio **/etc/apache2/sites-availables** y observa que existe un fichero denominado **default-ssl.conf** que contiene la configuración por defecto de un servidor HTTPS.

11. Habilita el servidor virtual ssl defecto (default-ssl.conf) de *Apache*.

    ```
    sudo a2ensite default-ssl
    ```

12. Reinicia el servidor para que los cambios tengan efecto.

13. Consulta el fichero **/etc/apache2/sietes-availabes/default-ssl.conf** y observa su configuración. Fíjate en las directivas que habilitan SSL y que definen la ruta del certificado digital que usurará el servidor, Figuras 5.113 y 5.114.

    El servidor utiliza por defecto un certificado digital autofirmado que se ha creado al instalar *Apache*. Un certificado autofirmado no está firmado por una autoridad de certificación (tercera parte de confianza) y por tanto, no existen mecanismos automáticos que garanticen su autenticidad. Por eso los navegadores nos pedirán confirmación cuando el servidor se lo envíe.

14. Desde **DesarrolloW7XX** abre el navegador y establece una conexión a `http:\\192.168.1.X7`, Figura 5.115.

15. Desde **DesarrolloW7XX** abre el navegador y establece una conexión a `https:\\192.168.1.X7`, Figuras 5.116 y 5.117.

```
<IfModule mod_ssl.c>
        <VirtualHost _default_:443>
                ServerAdmin webmaster@localhost

                DocumentRoot /var/www/html

                # Available loglevels: trace8, ..., trace1, debug, info, notice$
                # error, crit, alert, emerg.
                # It is also possible to configure the loglevel for particular
                # modules, e.g.
                #LogLevel info ssl:warn

                ErrorLog ${APACHE_LOG_DIR}/error.log
                CustomLog ${APACHE_LOG_DIR}/access.log combined

                # For most configuration files from conf-available/, which are
                # enabled or disabled at a global level, it is possible to
                # include a line for only one particular virtual host. For exam$
                # following line enables the CGI configuration for this host on$
                # after it has been globally disabled with "a2disconf".
                #Include conf-available/serve-cgi-bin.conf
```

Figura 5.113: Fichero **/etc/apache2/sites-availabes/default-ssl.conf**

```
                #   SSL Engine Switch:
                #   Enable/Disable SSL for this virtual host.
                SSLEngine on

                #   A self-signed (snakeoil) certificate can be created by inst$
                #   the ssl-cert package. See
                #   /usr/share/doc/apache2/README.Debian.gz for more info.
                #   If both key and certificate are stored in the same file, on$
                #   SSLCertificateFile directive is needed.
                SSLCertificateFile      /etc/ssl/certs/ssl-cert-snakeoil.pem
                SSLCertificateKeyFile /etc/ssl/private/ssl-cert-snakeoil.key

                #   Server Certificate Chain:
                #   Point SSLCertificateChainFile at a file containing the
                #   concatenation of PEM encoded CA certificates which form the
                #   certificate chain for the server certificate. Alternatively
```

Figura 5.114: Fichero **/etc/apache2/sites-availabes/default-ssl.conf**

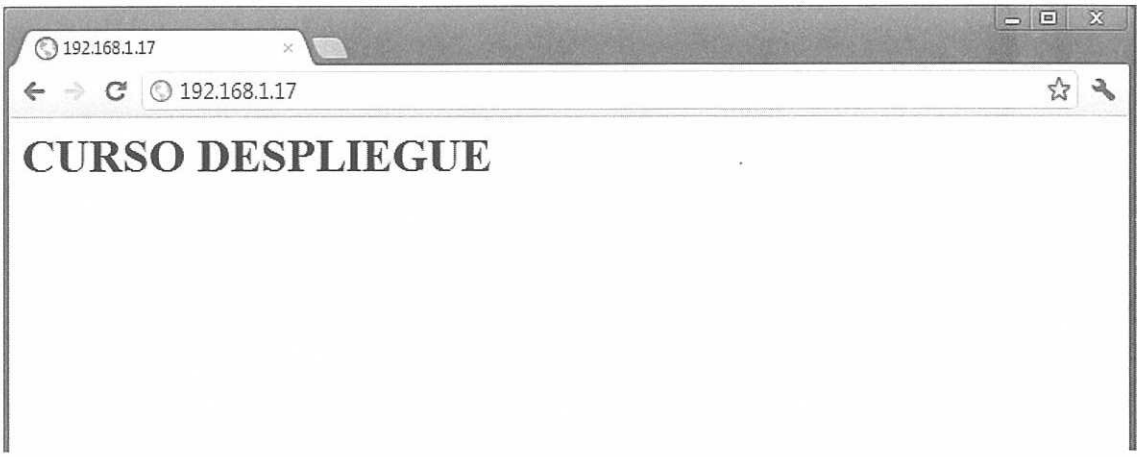


Figura 5.115: Conexión http

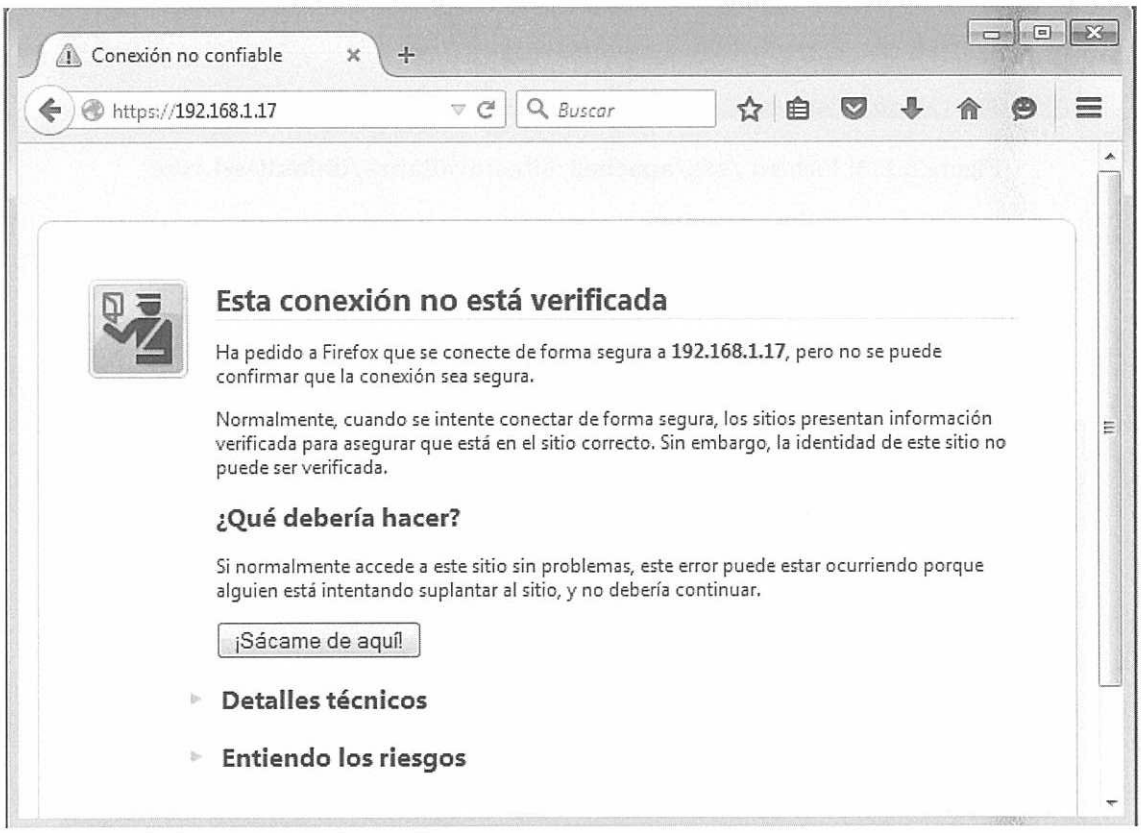


Figura 5.116: Conexión https

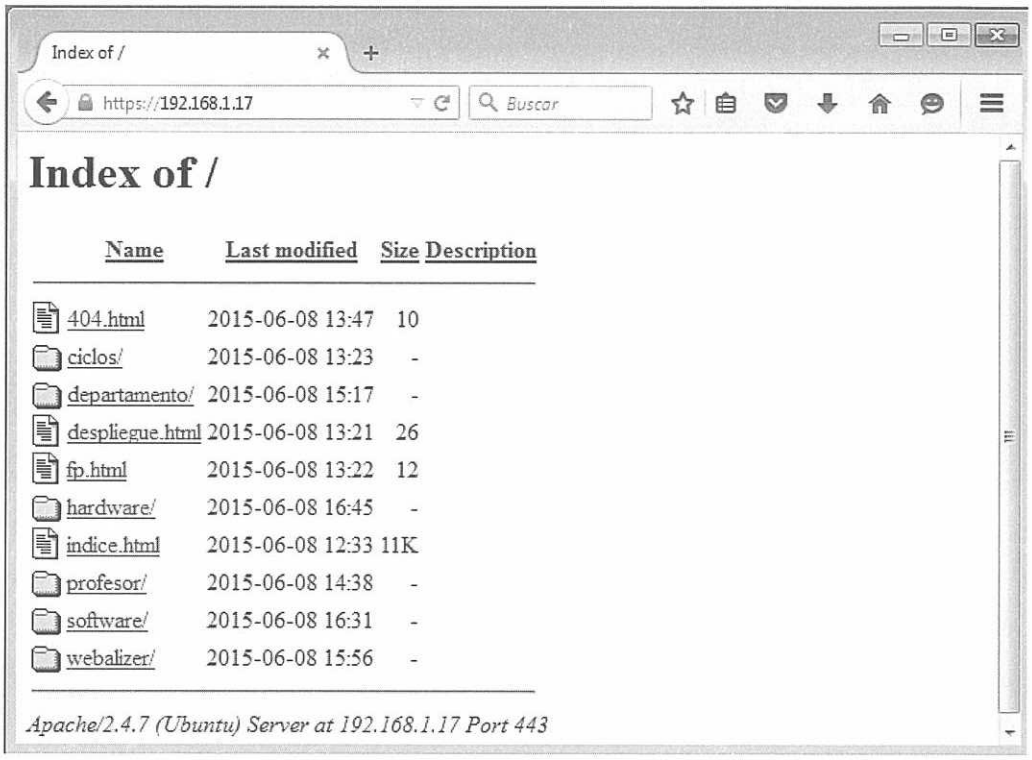


Figura 5.117: Conexión https

5.20. Servidor virtual HTTPS en *Linux*

Realiza la siguiente configuración en el servidor *Apache* instalado en **ServidorLinuxXX**.

- Deshabilita el servidor virtual *ssl* por defecto (*default-ssl*).
- Crea un certificado digital autofirmado con *openssl* para el dominio `seguro.dawXX.net`.
- Crea y habilita un servidor virtual *https* para el dominio `seguro.dawXX.net`
  - Directorio raíz **/var/www/html/seguro/**.
    - Se servirá el fichero **index.html** si no se indica ningún fichero en la URL.
    - Se mostrará un listado del directorio raíz si no se solicita ningún fichero.
    - Podrán acceder todos los usuarios.
  - El *log* de errores será **/var/log/apache2/seguro.error.log**.
  - El *log* de accesos será **/var/log/apache2/seguro.access.log**, con formato *combined*.

Prueba la configuración.

1. Configura el servidor DNS de **ServidorW2008XX** o **ServidorW2012XX** para que resuelva el nombre `seguro.dawXX.net`. La dirección IP asociada al nombre será la IP de **ServidoLinuxXX** es decir, `192.168.1.X7`, Figura 5.118.

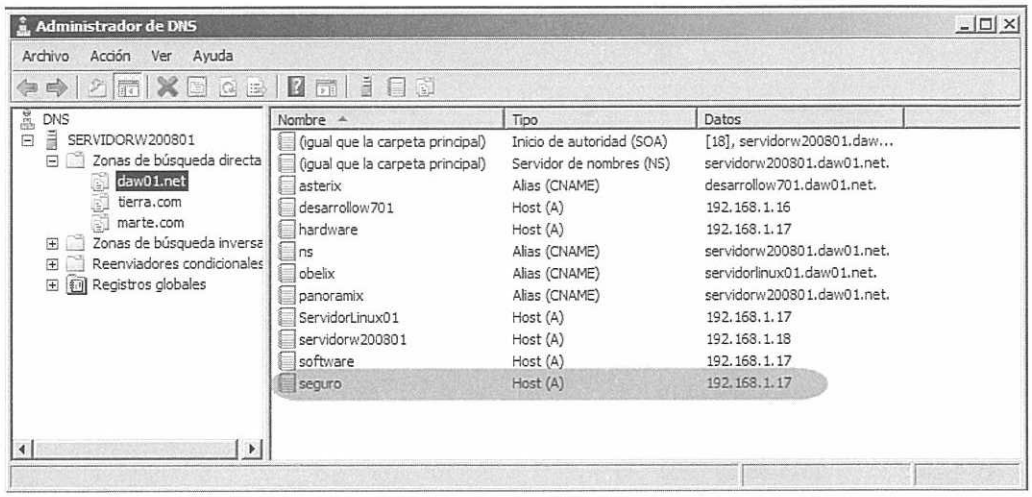


Figura 5.118: Configuración del servidor DNS en **ServidorWindowsXX**

2. Asegúrate que **DesarrolloW7XX** utiliza el servidor DNS que has configurado.
3. Inicia una sesión en **ServidorLinuxXX** con un usuario con privilegios de administración.
4. Crea el directorio **/var/www/html/seguro**.
5. Crea el fichero de texto **/var/www/html/seguro/index.html** con el contenido que quieras.
6. Crea un certificado digital autofirmado usando openssl.
   6.1. Sitúate en el directorio **home** del usuario con el que has iniciado sesión.
   6.2. Crea una clave privada RSA de 2048 bit, Figura 5.119.

```
openssl genrsa -out seguro.key 2048
```

```
alumno@ServidorLinux01:~$ openssl genrsa -out seguro.key 2048
Generating RSA private key, 2048 bit long modulus
.............................+++
.......................................................................
.................................................+++
e is 65537 (0x10001)
alumno@ServidorLinux01:~$
```

Figura 5.119: Creación de una clave privada

   6.3. Genera una solicitud de certificado (CSR, *Certificate Signing Request*).

```
openssl req -new -key seguro.key -out seguro.csr
```

   Introduce los datos del certificado, Figura 5.120

```
alumno@ServidorLinux01:~$ openssl req -new -key seguro.key -out seguro.csr
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.
-----
Country Name (2 letter code) [AU]:ES
State or Province Name (full name) [Some-State]:Madrid
Locality Name (eg, city) []:Madrid
Organization Name (eg, company) [Internet Widgits Pty Ltd]:daw01
Organizational Unit Name (eg, section) []:daw01
Common Name (e.g. server FQDN or YOUR name) []:seguro.daw01.net
Email Address []:admin@daw01.net

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
alumno@ServidorLinux01:~$ _
```

Figura 5.120: Creación de la solicitud del certificado

Esta solicitud de certificado se la podrías enviar a una autoridad de certificación para que generase el certificado (CRT). En este caso lo vamos a firmar nosotros, vamos a crear un certificado autofirmado.

6.4. Crea el certificado digital autofirmado usando la clave privada, Figura 5.121.

```
openssl x509 -req -days 365 -in seguro.csr -signkey seguro.key -out seguro.crt
```

```
alumno@ServidorLinux01:~$ openssl x509 -req -days 365 -in seguro.csr -signkey se
guro.key -out seguro.crt
Signature ok
subject=/C=ES/ST=Madrid/L=Madrid/O=daw01/OU=daw01/CN=seguro.daw01.net/emailAddre
ss=admin@daw01.net
Getting Private key
alumno@ServidorLinux01:~$ _
```

Figura 5.121: Creación del certificado digital autofirmado

7. Copia la clave y el certificado en los directorios que utiliza por defecto *Apache* y configura los permisos adecuados.

```
sudo mv seguro.key /etc/ssl/private/
sudo mv seguro.crt /etc/ssl/certs/
sudo chown root:ssl-cert /etc/ssl/private/seguro.key
sudo chmod 640 /etc/ssl/private/seguro.key
sudo chown root:root /etc/ssl/certs/seguro.crt
```

8. Crea el fichero **/etc/apache/site-available/seguro.conf** con las siguientes directivas, Figura 5.122.

```
<IfModule mod_ssl.c>
        <VirtualHost _default_:443>
                ServerName seguro.daw01.net

                DocumentRoot /var/www/html/seguro

                ErrorLog ${APACHE_LOG_DIR}/seguro.error.log
                CustomLog ${APACHE_LOG_DIR}/seguro.access.log combined

                <Directory /var/www/html/seguro>
                        Options Indexes FollowSymLinks
                        AllowOverride None
                        Require all granted
                </Directory>

                SSLEngine on
                SSLCertificateFile      /etc/ssl/certs/seguro.crt
                SSLCertificateKeyFile   /etc/ssl/private/seguro.key

        </VirtualHost>
</IfModule>
```

Figura 5.122: Fichero de configuración del servidor seguro

9. Deshabilita el servidor ssl por defecto.

   `sudo a2dissite default-ssl`

10. Habilita el servidor virtual seguro.

    `sudo a2ensite seguro`

11. Verifica que dentro del directorio **/etc/apache2/sites-enabled** se ha creado el enlace **seguro.conf**.

12. Reinicia el servidor para que los cambios tengan efecto.

13. Desde **DesarrolloW7XX** abre el navegador y establece una conexión a `https:\\seguro.dawXX.net`, Figuras 5.123, 5.124, 5.125 y 5.126.

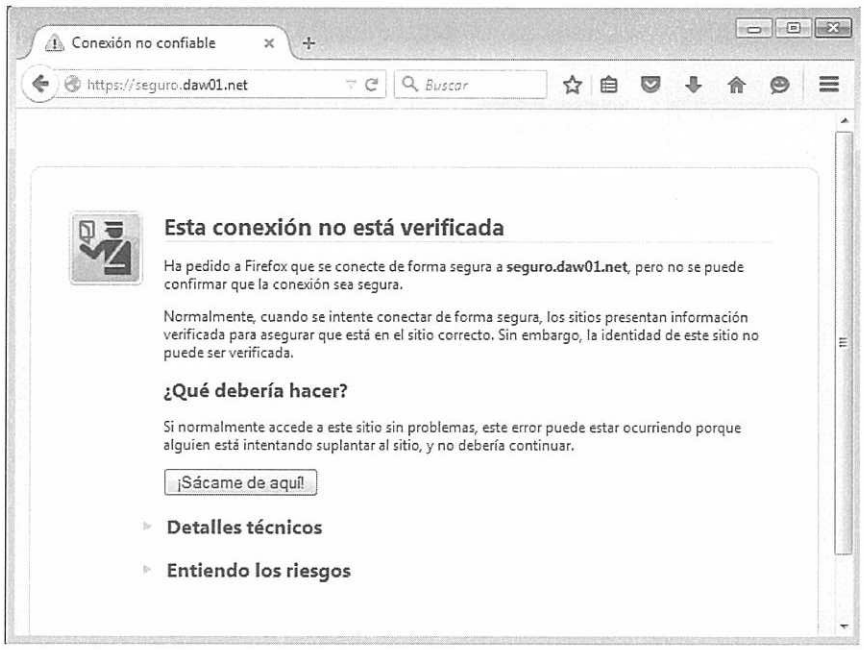


Figura 5.123: Conexión https

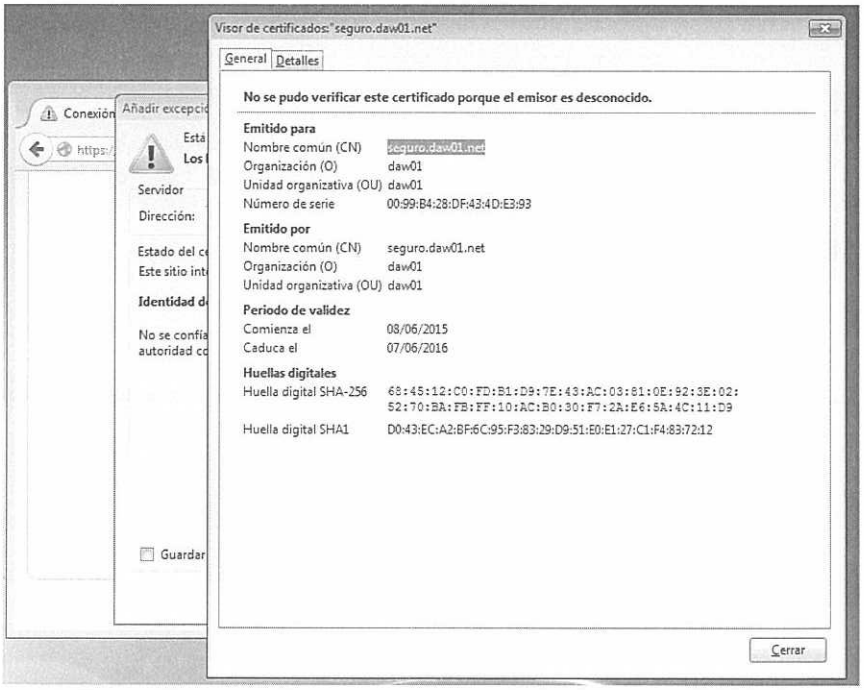


Figura 5.124: Conexión https

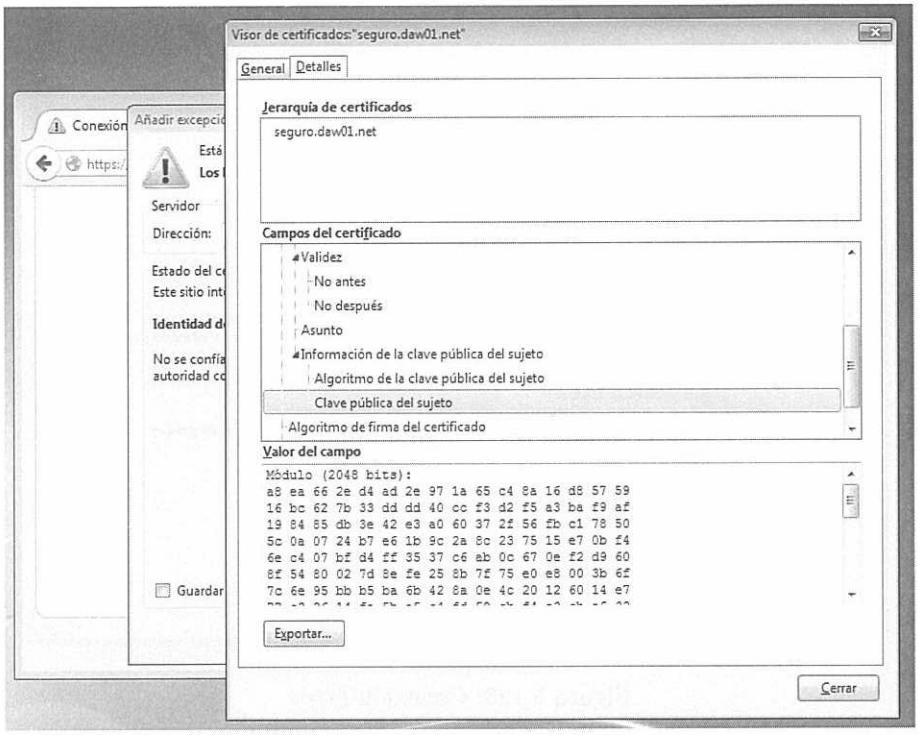


Figura 5.125: Conexión https

Figura 5.126: Conexión https

5.21. Servidor virtual HTTPS por defecto en *Windows*

Realiza la siguiente configuración en el servidor *Apache* instalado en **ServidorW2008XX**.

- Deshabilita los **servidores virtuales** creados en las prácticas anteriores.
- Habilita el modulo *mod_ssl*.
- Habilita el servidor virtual *ssl* por defecto

Prueba la configuración.

1. Inicia una sesión en **ServidorW2008XX** con un usuario con privilegios de administración.
2. Deshabilita los servidores virtuales creados en prácticas anteriores.
   2.1. Edita el fichero **C:\Program Files\Apache Software Foundation\Apache2.2\conf\httpd.conf** y comenta la directiva `Include` del fichero **conf/extra/httpd-vhost.conf**.
   2.2. Reinicia el servidor para que los cambios tengan efecto.
3. Edita el fichero **C:\Program Files\Apache Software Foundation\Apache2.2\conf\httpd.conf** y habilita el módulo **mod_ssl** eliminando el comentario de la directivas `LoadModule`, Figura 5.127.

```
#LoadModule mime_magic_module modules/mod_mime_magic.so
LoadModule negotiation_module modules/mod_negotiation.so
#LoadModule proxy_module modules/mod_proxy.so
#LoadModule proxy_ajp_module modules/mod_proxy_ajp.so
#LoadModule proxy_balancer_module modules/mod_proxy_balancer.so
#LoadModule proxy_connect_module modules/mod_proxy_connect.so
#LoadModule proxy_ftp_module modules/mod_proxy_ftp.so
#LoadModule proxy_http_module modules/mod_proxy_http.so
#LoadModule proxy_scgi_module modules/mod_proxy_scgi.so
#LoadModule reqtimeout_module modules/mod_reqtimeout.so
#LoadModule rewrite_module modules/mod_rewrite.so
LoadModule setenvif_module modules/mod_setenvif.so
#LoadModule speling_module modules/mod_speling.so
LoadModule ssl_module modules/mod_ssl.so
#LoadModule status_module modules/mod_status.so
#LoadModule substitute_module modules/mod_substitute.so
#LoadModule unique_id_module modules/mod_unique_id.so
LoadModule userdir_module modules/mod_userdir.so
#LoadModule usertrack_module modules/mod_usertrack.so
#LoadModule version_module modules/mod_version.so
```

Figura 5.127: Habilitar el módulo mod_ssl

4. Habilita el servidor virtual ssl defecto (default-ssl) de *Apache*. Edita el fichero **C:\Program Files\Apache Software Foundation\Apache2.2\conf\httpd.conf** y eliminana el comentario de la directiva `Include` del fichero **conf/extra/httpd-ssl.conf**, Figura 5.128.
5. Si observas en el fichero **C:\Program Files\Apache Software Foundation\Apache2.2\conf\extra\httpd-ssl.conf** existen dos directivas para definir el certificado digital y la clave privada del servidor (que debemos crear), Figura 5.129.
6. Crea un certificado digital autofirmado usando *openssl*.
   6.1. Abre un terminal.
   6.2. Accede al directorio **C:\Program Files\Apache Software Foundation\Apache2.2\conf**.
   6.3. Ejecuta el comando **C:\Program Files\Apache Software Foundation\Apache2.2\bin\openssl**.

```
#Include conf/extra/httpd-vhosts.conf

# Local access to the Apache HTTP Server Manual
#Include conf/extra/httpd-manual.conf

# Distributed authoring and versioning (WebDAV)
#Include conf/extra/httpd-dav.conf

# Various default settings
#Include conf/extra/httpd-default.conf

# Secure (SSL/TLS) connections
Include conf/extra/httpd-ssl.conf
#
# Note: The following must must be present to support
#       starting without SSL on platforms with no /dev/random equivalent
#       but a statically compiled-in mod_ssl.
#
<IfModule ssl_module>
SSLRandomSeed startup builtin
```

Figura 5.128: Habilitar el servidor virtual https

```
#SSLCipherSuite RC4-SHA:AES128-SHA:HIGH:MEDIUM:!aNULL:!MD5
#SSLHonorCipherOrder on

#   Server Certificate:
#   Point SSLCertificateFile at a PEM encoded certificate.  If
#   the certificate is encrypted, then you will be prompted for a
#   pass phrase.  Note that a kill -HUP will prompt again.  Keep
#   in mind that if you have both an RSA and a DSA certificate you
#   can configure both in parallel (to also allow the use of DSA
#   ciphers, etc.)
SSLCertificateFile "C:/Program Files/Apache Software Foundation/Apache2.2/conf/server.crt"
#SSLCertificateFile "C:/Program Files/Apache Software Foundation/Apache2.2/conf/server-dsa.crt"

#   Server Private Key:
#   If the key is not combined with the certificate, use this
#   directive to point at the key file.  Keep in mind that if
#   you've both a RSA and a DSA private key you can configure
#   both in parallel (to also allow the use of DSA ciphers, etc.)
SSLCertificateKeyFile "C:/Program Files/Apache Software Foundation/Apache2.2/conf/server.key"
#SSLCertificateKeyFile "C:/Program Files/Apache Software Foundation/Apache2.2/conf/server-dsa.key"
```

Figura 5.129: Fichero **httpd-ssl.conf**

```
Administrador: Símbolo del sistema - "c:\Program Files\Apache Software Foundation\Apache2.2\bin\...

C:\Program Files\Apache Software Foundation\Apache2.2\conf>"c:\Program Files\Apa
che Software Foundation\Apache2.2\bin\openssl.exe"
OpenSSL> genrsa -out server.key 2048
Loading 'screen' into random state - done
Generating RSA private key, 2048 bit long modulus
.......................+++
...................................+++
e is 65537 (0x10001)
OpenSSL> _
```

Figura 5.130: Creación de una clave privada

6.4. Crea una clave privada RSA de 2048 bit, Figura 5.130.

```
Openssl> genrsa -out server.key 2048
```

6.5. Genera una solicitud de certificado (CSR, *Certificate Signing Request*).

```
Openssl> req -config openssl.cnf -new -key server.key -out server.csr
```

Introduce los datos del certificado, Figura 5.131

```
Administrador: Símbolo del sistema - "c:\Program Files\Apache Software Foundation\Apache2.2\bin\...
OpenSSL> req -new -key server.key -out server.csr
Unable to load config info from /usr/local/ssl/openssl.cnf
error in req
OpenSSL> req -config openssl.cnf -new -key server.key -out server.csr
Loading 'screen' into random state - done
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.
-----
Country Name (2 letter code) [AU]:ES
State or Province Name (full name) [Some-State]:Madrid
Locality Name (eg, city) []:Madrid
Organization Name (eg, company) [Internet Widgits Pty Ltd]:daw01.net
Organizational Unit Name (eg, section) []:daw01.net
Common Name (e.g. server FQDN or YOUR name) []:servidorwindows01.daw01.net
Email Address []:admin@daw01.net

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
OpenSSL>
```

Figura 5.131: Creación de la solicitud del certificado

Esta solicitud de certificado se la podrías enviar a una autoridad de certificación para que generase el certificado (CRT). En este caso lo vamos a firmar nosotros, vamos a crear un certificado autofirmado.

6.6. Crea el certificado digital autofirmado usando la clave privada, Figura 5.132.

```
Openssl> x509 -req -days 365 -in server.csr -signkey server.key -out server.crt
```

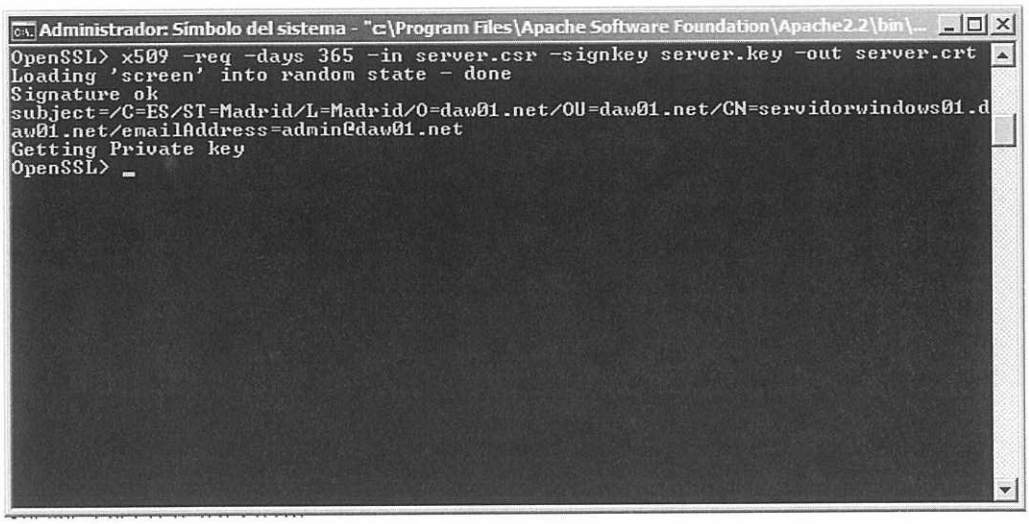


Figura 5.132: Creación del certificado digital autofirmado

7. Reinicia el servidor para que los cambios tengan efecto.

8. Verifica que el servidor escucha en los puertos 80/TCP y 443/TCP.

```
netstat -a -p TCP -n
```

9. Desde **DesarrolloW7XX** abre el navegador y establece una conexión a `http:\\192.168.1.X8`, Figura 5.133.

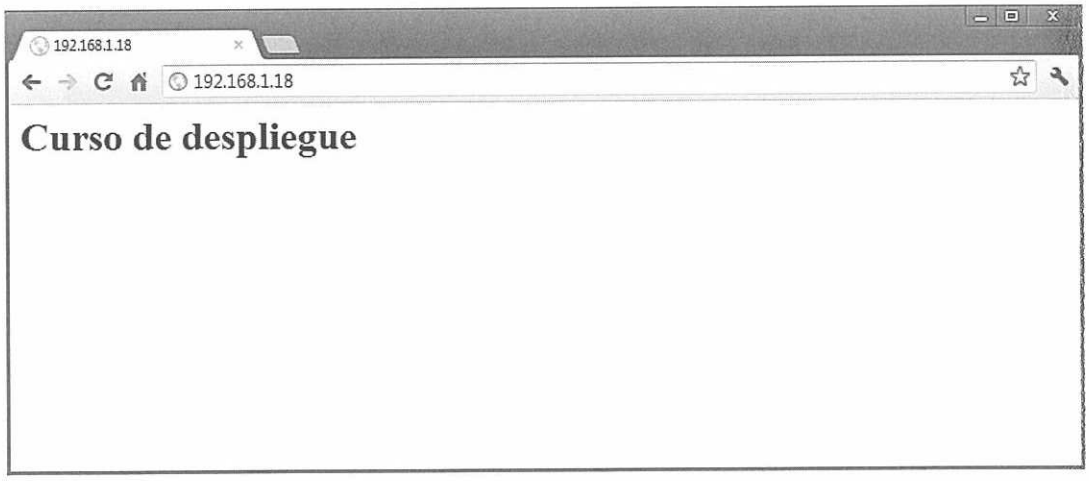


Figura 5.133: Conexión http

10. Desde **DesarrolloW7XX** abre el navegador y establece una conexión a `https:\\192.168.1.X8`, Figuras 5.134 y 5.135.

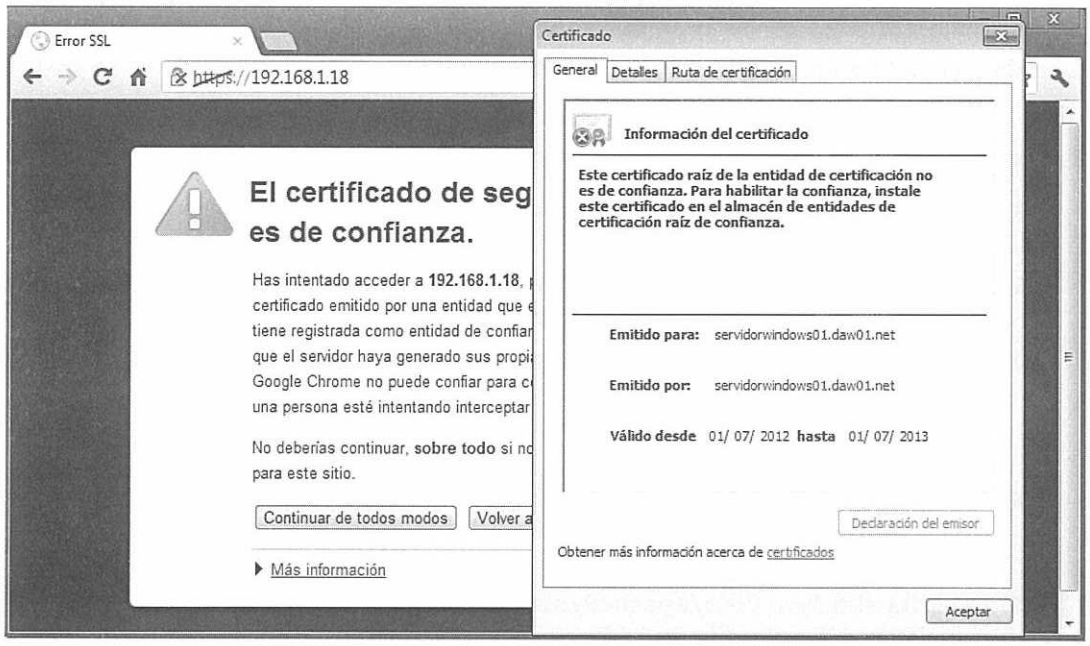


Figura 5.134: Conexión https

Figura 5.135: Conexión https

5.22. WebDav

En la máquina **ServidorLinuxXX** crea el directorio **/var/www/html/webdav** y configura el servidor virtual por defecto para que el directorio sea accesible desde clientes WebDav. Deberás configurar la autenticación HTTP *Digest* sobre el directorio **/var/www/html/webdav** para que solo puedan acceder los usuarios **admin1** y **admin2**. También tendrás que configurar los permisos adecuados para que los clientes puedan consultar, borrar, modificar, etc. ficheros del directorio.

1. **Habilitar los módulos necesarios para WebDav**

   1.1. Inicia una sesión en **ServidorLinuxXX** con un usuario con privilegios de administrador.

   1.2. Consulta el directorio **/etc/apache2/mods-available** y observa que entre los módulos disponibles para cargar están **dav** y **dav_fs**.

   1.3. Habilita los módulos ejecutando los siguientes comandos:

   ```
   sudo a2enmod dav
   sudo a2enmod dav_fs
   ```

   1.4. Verifica que dentro del directorio **/etc/apache2/mods-enabled** se han creado enlaces simbólicos de los módulos **dav** y **dav_fs** (ficheros **.conf** y **.load**) hacia **/etc/apache2/mod_availables**

   1.5. Consulta el fichero **/etc/apache2/mod_availables/dav_fs.conf** y observa su configuración por defecto, Figura 5.136.

   ```
   DAVLockDB ${APACHE_LOCK_DIR}/DAVLock
   ```

   Figura 5.136: Fichero **/etc/apache2/mod-availables/dav_fs.conf**

   La directiva **DAVLockDB** define la ubicación de la base de datos que almacenará la información para controlar los bloqueos (control del acceso simultaneo a los mismos ficheros por parte de múltiples clientes).

   1.6. Reinicia el servidor para que los cambios tengan efecto.

2. **Configuración del directorio**

   2.1. Crea el directorio **/var/www/html/webdav** y dentro el fichero **prueba.html**.

   2.2. Edita el fichero **/etc/apache/sites-available/000-default.conf** y añade las siguientes directivas, Figura 5.137.

```
<Directory /var/www/html/departamento>
        Options Indexes FollowSymLinks
        AllowOverride None
        AuthType Digest
        AuthName "informatica"
        AuthDigestProvider file
        AuthUserFile /etc/apache2/digest
        Require user admin1 admin2
</Directory>

<Directory /var/www/html/webdav>
        Dav On
        Options Indexes FollowSymLinks
        AllowOverride None
        AuthType Digest
        AuthName "informatica"
        AuthDigestProvider file
        AuthUserFile /etc/apache2/digest
        Require user admin1 admin2
</Directory>
```

Figura 5.137: Fichero **/etc/apache/sites-available/000-default.conf**

Como puedes observar se ha utilizado el fichero **/etc/apache2/digest** de usuarios y contraseñas creado en prácticas anteriores. Si no has realizado la práctica correspondiente deberías crearlo según se explica a continuación.

a Crea el fichero y añade el usuario **admin1** al dominio `informatica` (la opción -c es para crear el fichero).

```
sudo htdigest -c /etc/apache2/digest informatica admin1
```

b Añade el usuario **admin2** (no se usa la opción -c porque el fichero ya existe).

```
sudo htdigest /etc/apache2/digest informatica admin2
```

2.3. Reinicia el servidor para que los cambios tengan efecto.

3. <u>Conexión desde un cliente WebDav</u>

Para conectarse a través de WebDav es necesario utilizar un cliente específico. **Windows7** tiene integrado un cliente WebDav en sus explorador de archivos (tiene algunos problemas con conexiones sobre HTTPS).

3.1. En **DesarrolloW7XX** accede a **Inicio, Equipo, botón secundario del ratón, Conectar a Unidad de Red.**

3.2. Introduce los siguientes parámetros para establecer una conexión, Figura 5.138.

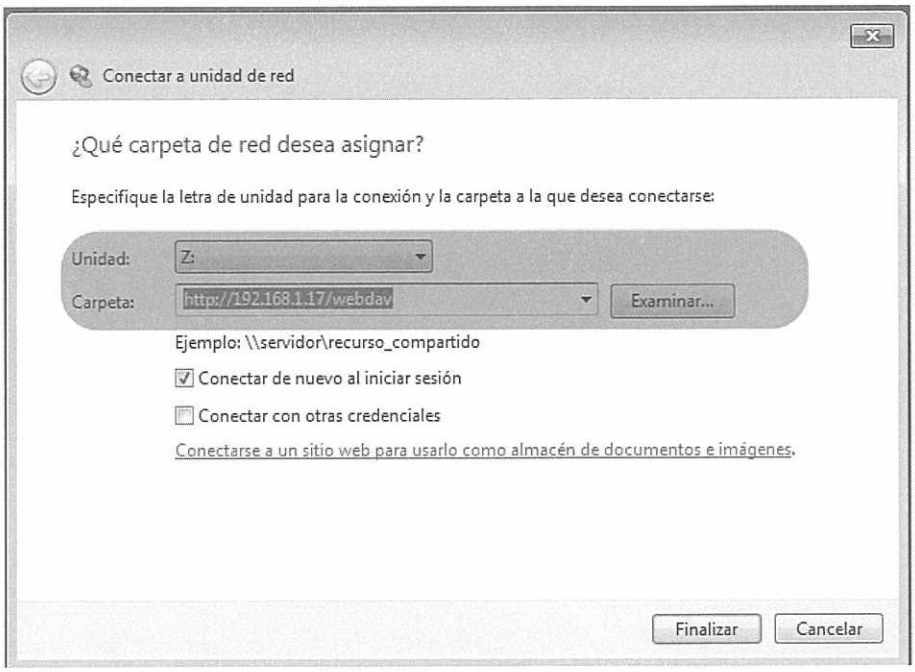


Figura 5.138: Configuración de la conexión WebDav desde **DesarrrolloW7XX**

3.3. Pincha en **Conectar** y establece una conexión como **admin1**, Figuras 5.139 y 5.140.

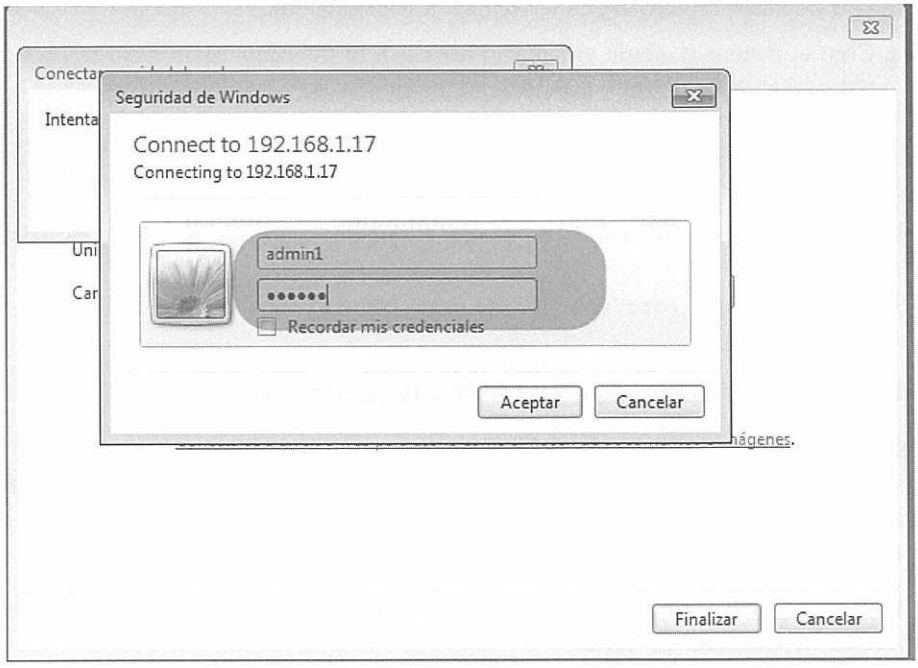


Figura 5.139: Conexión como *admin1*

Figura 5.140: Conexión como **admin1**

3.4. Intenta crear un nuevo archivo. Observa que no tienes permisos.

4. <u>Configuración de los permisos adecuados</u>

4.1. En **ServidorLinuxXX** consulta el fichero **/etc/apache2/envars**. Observa el usuario y el grupo con los que se ejecuta *Apache*, Figura 5.141.

```
# envvars - default environment variables for apache2ctl

# this won't be correct after changing uid
unset HOME

# for supporting multiple apache2 instances
if [ "${APACHE_CONFDIR##/etc/apache2-}" != "${APACHE_CONFDIR}" ] ; then
        SUFFIX="-${APACHE_CONFDIR##/etc/apache2-}"
else
        SUFFIX=
fi

# Since there is no sane way to get the parsed apache2 config in scripts, some
# settings are defined via environment variables and then used in apache2ctl,
# /etc/init.d/apache2, /etc/logrotate.d/apache2, etc.
export APACHE_RUN_USER=www-data
export APACHE_RUN_GROUP=www-data
export APACHE_PID_FILE=/var/run/apache2$SUFFIX.pid
export APACHE_RUN_DIR=/var/run/apache2$SUFFIX
export APACHE_LOCK_DIR=/var/lock/apache2$SUFFIX
# Only /var/log/apache2 is handled by /etc/logrotate.d/apache2.
export APACHE_LOG_DIR=/var/log/apache2$SUFFIX
```

Figura 5.141: Fichero **/etc/apache2/envars**

4.2. Accede al directorio **/var/www/html**.

4.3. Consulta los permisos del directorio **/var/www/html/webdav**.

4.4. Cambia el grupo de **/var/www/html/webdav** a **www-data** y otórgale privilegios de escritura.

```
sudo chown root:www-data /var/www/html/webdav
sudo chmod 775 /var/www/html/webdav
```

4.5. Desde **DesarrolloW7XX** observa que ahora es posible crear un nuevo archivo, Figura 5.142.

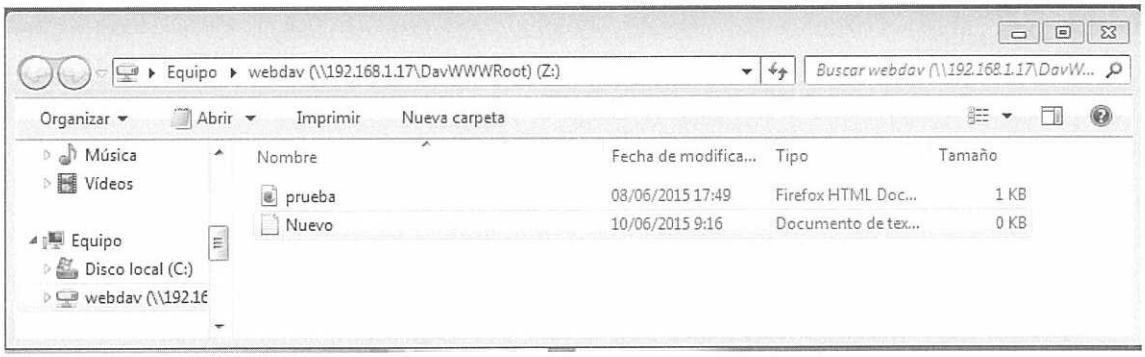


Figura 5.142: Nuevo archivo

4.6. Prueba a copiar, borrar, renombrar, etc.

Capítulo 6

Despliegue en entornos WAMP-LAMP

6.1. Instalación de un servidor WAMP

Instala un servidor **WAMP** en la máquina **DesarrolloW7XX** usando los paquetes proporcionados en `http://www.wampserver.com`.

1. **Instalación**

   1.1. Inicia sesión con un usuario con privilegios de administrador en **DesarrolloW7XX**.

   1.2. Inicia una navegador y accede a `http://www.wampserver.com/`.

   1.3. Descarga la versión de 32 bits, Figura 6.1 y observa la versiones del software que instalas. Fíjate que también se instalan la aplicaciónes Web ***phpMyadmin*** y ***SQLBudy*** que permiten administrar bases de datos MySQL.

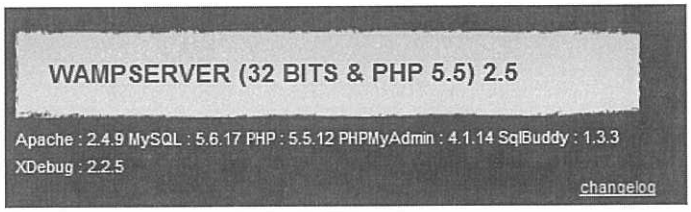


Figura 6.1: Descarga de *Wampserver*

   1.4. Para instalar el software es necesario instalar previamente *Visual C++ Redistributable para Visual Studio 2012 Update 4* (`http://www.microsoft.com/es-ES/download/details.aspx?id=30679`), Figura 6.2.

Figura 6.2: Descarga de *Visual C++ 2010 SP1 Redistributable Package x86*

1.5. Descarga e instala *Visual C++ Redistributable para Visual Studio 2012 Update 4*.

1.6. Descarga y ejecuta el instalador de *Wampserver*.

1.7. Pincha en ***Next*** para iniciar el asistente de instalación.

1.8. Acepta los términos de la licencia y pincha en ***Next***.

1.9. Observa el directorio de instalación por defecto **C:\wamp** y pincha en ***Next***.

1.10. Selecciona si quieres accesos directos y pincha en ***Next***.

1.11. Pincha en ***Install*** para continuar.

1.12. Indica si quieres o no usar *Firefox* como navegador por defecto.

1.13. Acepta para configurar en el *Firewall* de *Windows* que se permitan conexiones a *Apache*.

1.14. Deja las opciones por defecto en la configuración del servidor SMTP que usará la función mail de PHP y pincha en ***Next***.

1.15. Una vez terminada la instalación pincha en ***Finish***. Observa que está seleccionada la opción que arrancará ***Wamp*** al finalizar, Figura 6.3.

Figura 6.3: *Instalación terminada*

1.16. Comprueba que en la barra de tareas abajo a la izquierda hay un icono que muestra una herramienta, para iniciar/parar y configurar los distintos servidores que se han instalado, Figura 6.4.

Figura 6.4: Herramienta de administración

1.17. Haz clic con el botón derecho sobre el icono de la herramienta de administración y accede a ***Language*** y selecciona (si quieres) *Spanish*.

2. ***Apache***

2.1. Accede en la herramienta de administración al menú de *Apache*, Figura 6.5. Observa que desde aquí se puede iniciar y para el servidor, consultar y habilitar módulos, añadir Alias y acceder al fichero de configuración `httpd.conf`.

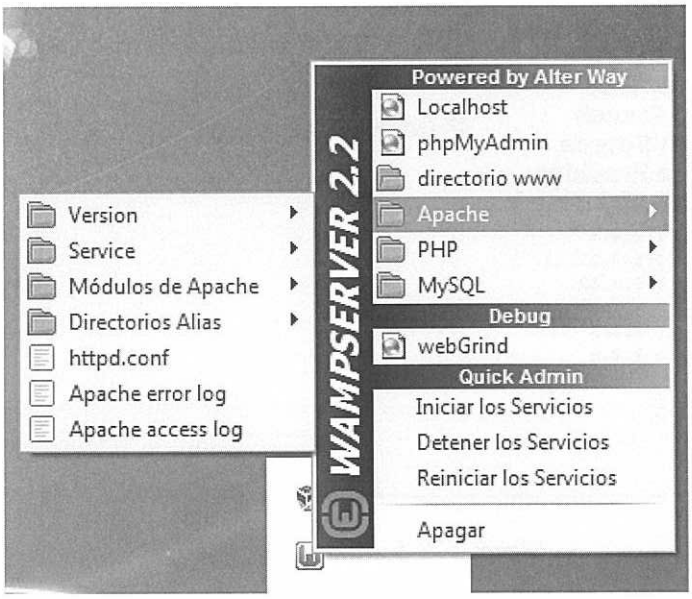


Figura 6.5: Configuración de *Apache*

2.2. Consulta el contenido del fichero **C:\wamp\bin\apache2.4.9\conf\httpd.conf** y observa cuál es el `DocumentRoot`, Figura 6.6.

```
#
# DocumentRoot: The directory out of which you will serve your
# documents. By default, all requests are taken from this directory, but
# symbolic links and aliases may be used to point to other locations.
#
DocumentRoot "c:/wamp/www/"
```

Figura 6.6: Fichero httpd.conf

2.3. Consulta el contenido del directorio **C:\wamp\www**.

2.4. Abre un navegador y accede a `http://127.0.0.1` o `http://localhost` (como no se especifica el puerto se utiliza por defecto el puerto 80), Figura 6.7.

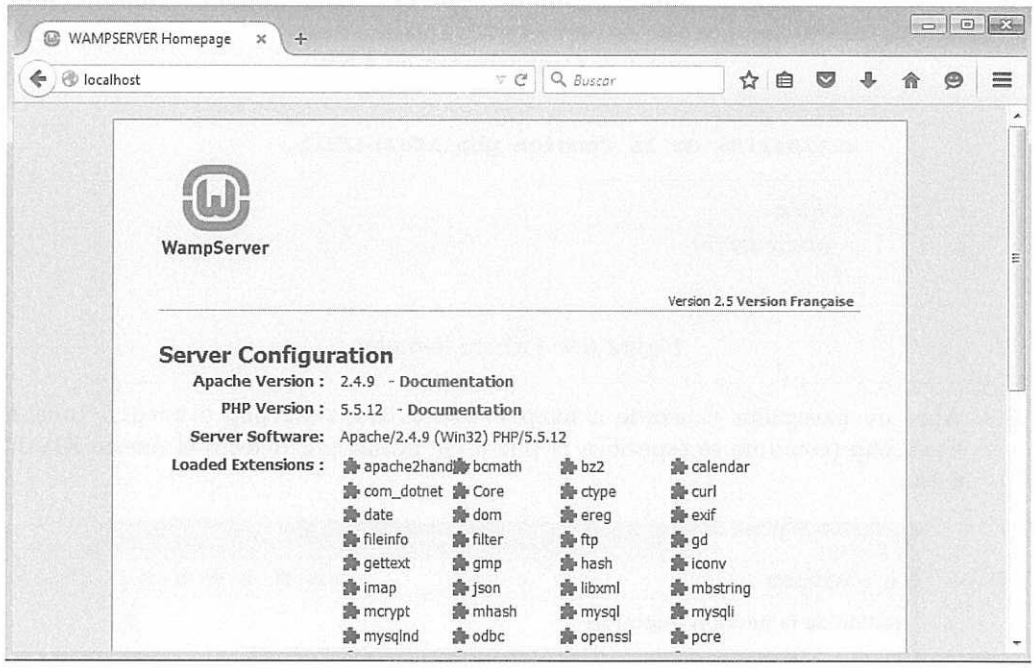


Figura 6.7: Conexión a *Apache*

3. ***PHP***

3.1. Accede en la herramienta de administración al menú de *PHP*, Figura 6.8. Observa las diferentes opciones de configuración.

3.2. En el directorio **C:\wamp\www** crea el fichero **test.php** con el siguiente contenido, Figura 6.9.

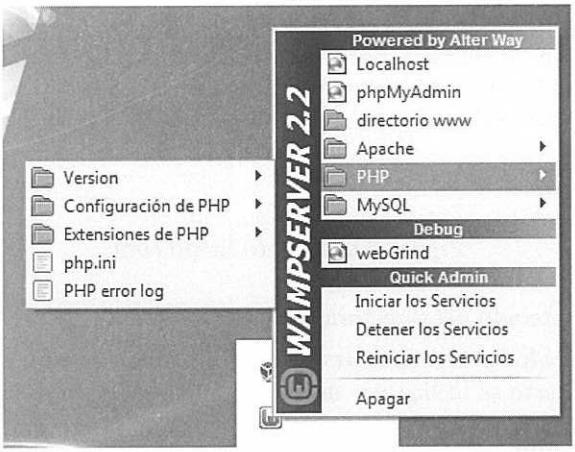


Figura 6.8: Configuración de *PHP*

```
<H1>Salida de la funcion phpinfo()</H1>

 <?php
  phpinfo();
 ?>
```

Figura 6.9: Fichero test.php

3.3. Abre un navegador y accede a `http://127.0.0.1/test.php` o `http://localhost/test.php` (como no se especifica el puerto se utiliza por defecto el puerto 80), Figura 6.10.

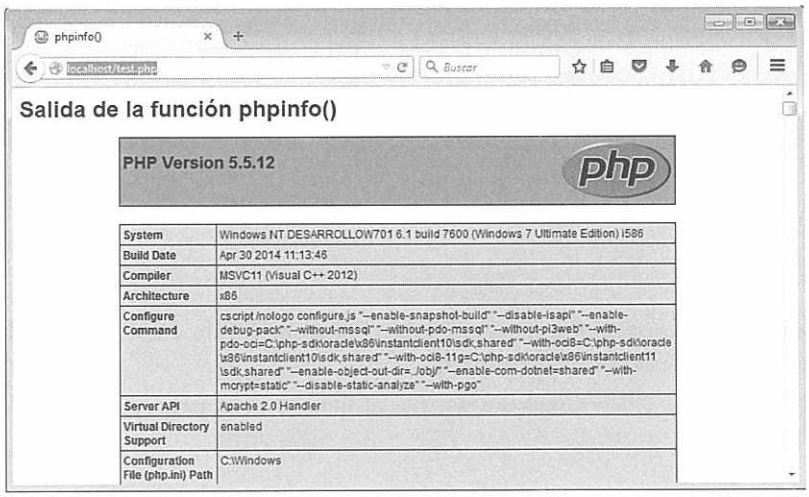


Figura 6.10: Prueba de PHP

4. ***MySQL***

4.1. Accede en la herramienta de administración al menú de *MySQL*, Figura 6.11. Observa las diferentes opciones de configuración.

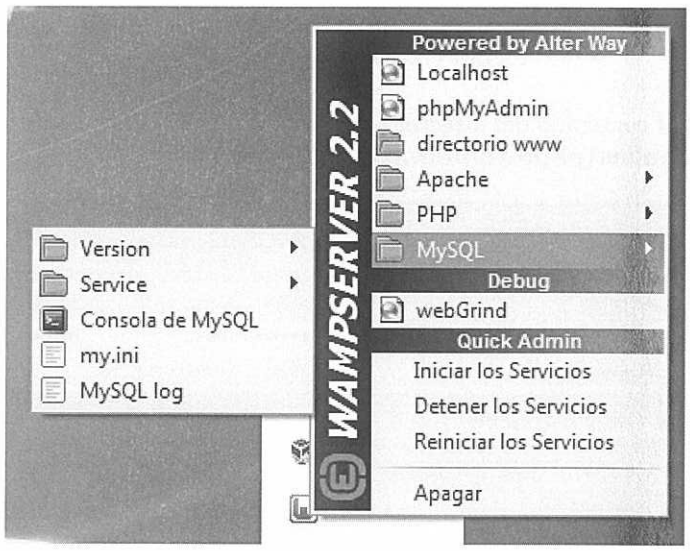


Figura 6.11: Configuración de *MySQL*

4.2. Abre un terminal y comprueba que el servidor *MySQL* está escuchando peticiones en el puerto **3306/TCP.**

```
netstat -a -p TCP -n
```

4.3. El usuario **root** de *MySQL* no tiene definida ninguna contraseña. Asígnale una contraseña para el acceso desde el equipo local.

a. Accede a la herramienta de administración y accede al menú de *MySQL* y pincha en **Consola de MySQL**.

b. Como por defecto se conecta como **root** deja el *password* vacío y pulsa ***Enter***.

c. Ejecuta los siguientes comando para asignar una contraseña al usuario **root**.

```
mysql> set password for 'root'@'127.0.0.1'=password('despliegue');
mysql> set password for 'root'@'localhost'=password('despliegue');
mysql> set password for 'root'@'::1'=password('despliegue');
```

d. Puedes consultar la tabla de usuarios con el siguiente comando.

```
mysql> select host, user, password from mysql.user;
```

e. Borra el acceso con un usuario anónimo.

```
mysql>  drop user ''@'localhost';
```

f. Solo se puede acceder como usuario **root** desde el equipo local.

5. ***Aplicaciones. PHPMyAdmin***

5.1. Consulta el contenido del fichero **C:\wamp\bin\apache2.4.9\conf\httpd.conf** y observa que la final de fichero se incluye el directorio **c:/wamp/alias/***, Figura 6.12.

```
#IncludeOptional "c:/wamp/vhosts/*"
Include "c:/wamp/alias/*"
```

Figura 6.12: Fichero httpd.conf

5.2. Consulta el contenido del directorio **c:\wamp\alias** y el contenido del fichero **c:\wamp\alias\phpmyadmin.conf**, Figura 6.13.

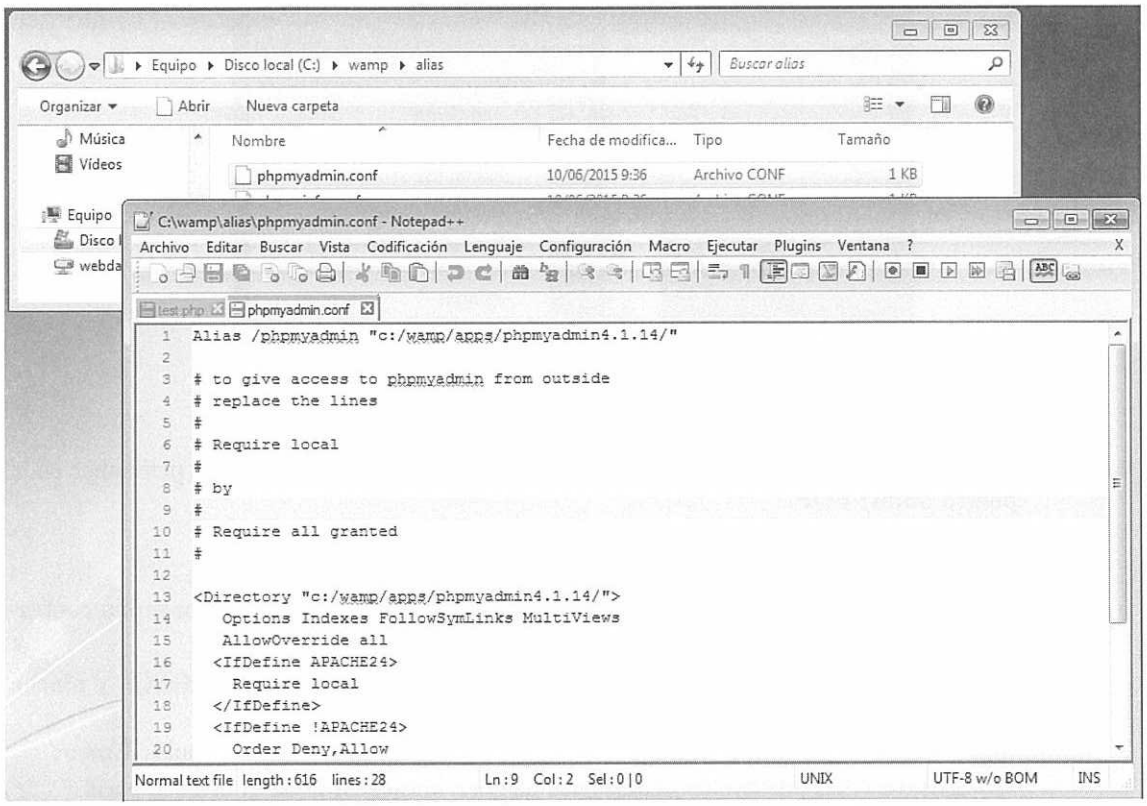


Figura 6.13: Fichero phpmyadmin.conf

5.3. Abre un navegador y accede a `http://127.0.0.1/phpmyadmin` o a `http://localhost/phpmyadmin`. Observa el mensaje de error. Se produce porque anteriormente hemos asignado un *password* al usuario **root** y *phpmyadmin* intenta conectarse como usuario **root** pero sin *password*. Si no hubiésemos puesto el *password* si se podría conectar al MySQL, Figura 6.14.

Figura 6.14: Error de conexión a *MySQL* en *phpmyadmim*

5.4. Edita el fichero **C:\wamp\apps\phpmyadmin4.1.14\config.inc.php** y configura la autenticación para que pida usuario y contraseña en lugar de que use por defecto el usuario **root**, Figura 6.15.

```
/*
 * Servers configuration
 */
$i = 0;

/*
 * First server
 */
$i++;
/* Authentication type */
$cfg['Servers'][$i]['verbose'] = 'mysql wampserver';
//$cfg['Servers'][$i]['auth_type'] = 'cookie';
$cfg['Servers'][$i]['auth_type'] = 'cookie';
$cfg['Servers'][$i]['user'] = 'root';
$cfg['Servers'][$i]['password'] = '';
/* Server parameters */
$cfg['Servers'][$i]['host'] = '127.0.0.1';
$cfg['Servers'][$i]['connect_type'] = 'tcp';
$cfg['Servers'][$i]['compress'] = false;
/* Select mysql if your server does not have mysqli */
$cfg['Servers'][$i]['extension'] = 'mysqli';
$cfg['Servers'][$i]['AllowNoPassword'] = true;
```

Figura 6.15: Configuración de *phpmyadmim*

5.5. Abre un navegador y accede a `http://127.0.0.1/phpmyadmin` o `http://localhost/phpmyadmin`, Figura 6.16.

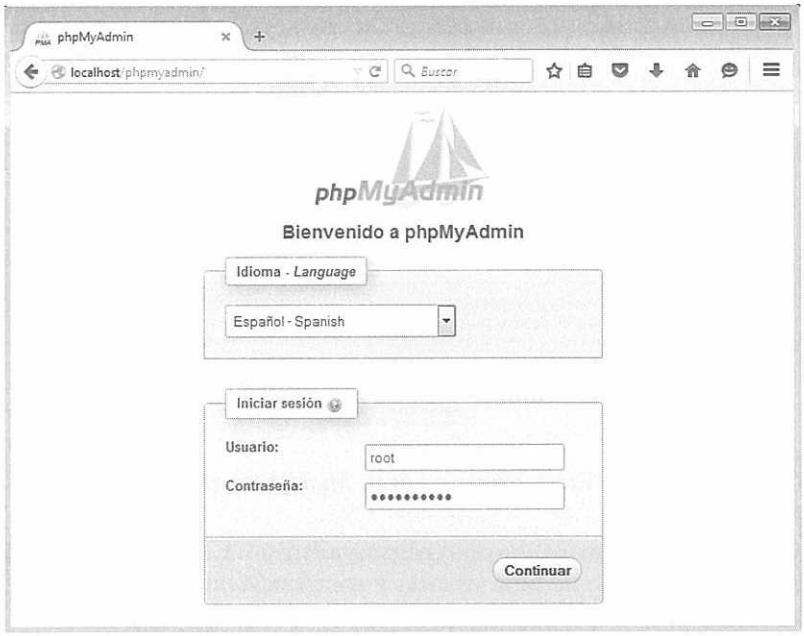


Figura 6.16: Conexión a *phpmyadmim*

5.6. Inicia sesión con el usuario **root**, Figura 6.17.

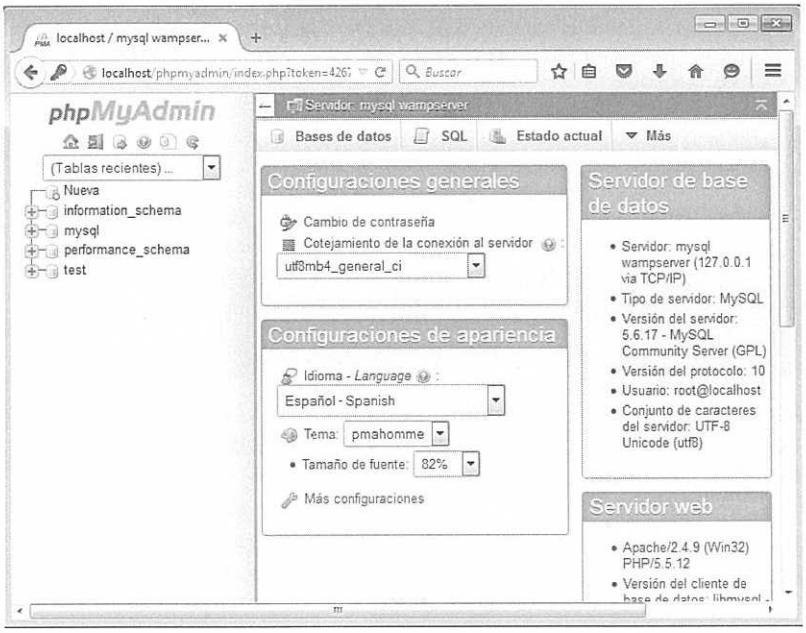


Figura 6.17: Inicio de sesión en *phpmyadmim*

6.2. Instalación de un servidor LAMP

Instala un servidor **LAMP** en la máquina **ServidorLinuxXX** usando los paquetes disponibles en los repositorios oficiales de *Ubuntu*. Posteriormente instala la aplicación ***PHPMyAdmin*** desde los repositorios de *Ubuntu*.

1. **Instalación**

   1.1. Inicia sesión con un usuario con privilegios de administrador en **ServidorLinuxXX**.

   1.2. Ejecuta el siguiente comando, selecciona ***LAMP Server*** y acepta los cambios, Figura 6.18 (recuerda que el servidor web *Apache* ya se ha instalado en las prácticas anteriores).

```
sudo tasksel
```

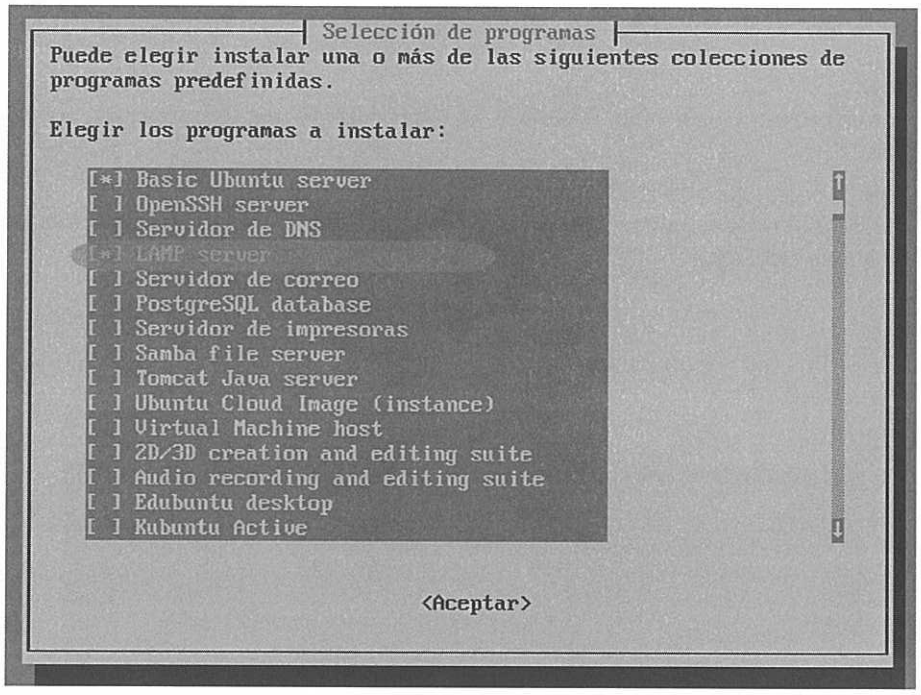


Figura 6.18: Instalación de un servidor LAMP

También es posible instalar los paquetes uno a uno en lugar de utilizar ***tasksel***.

```
sudo apt-get install apache2
sudo apt-get install php5 libapache2-mod-php5
sudo apt-get install php5-cli
sudo apt-get install mysql-server
sudo apt-get install php5-mysql
```

1.3. Introduce la clave **despliegue** para el usuario **root** de *Mysql*, Figura 6.19.

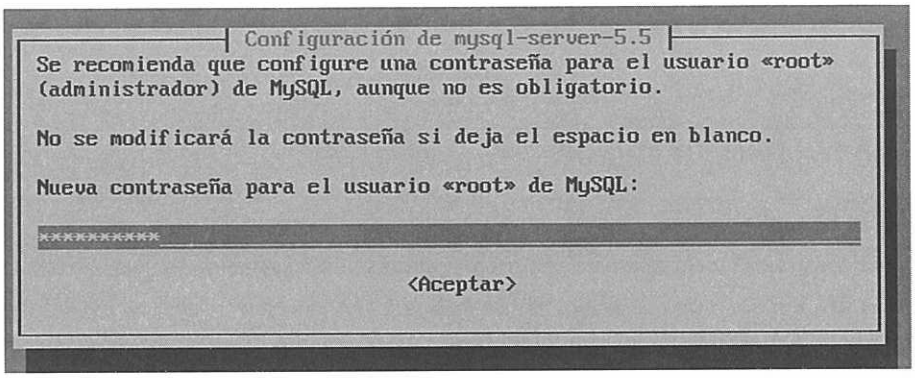


Figura 6.19: Clave del usuario root en *MySQL*

2. ***Apache***

   2.1. Accede al directorio **/etc/apache2** y observa que se mantiene la configuración de *Apache*.

   2.2. Desde **DesarrolloW7XX** abre un navegador y accede a `http://192.168.1.X7` para verificar el funcionamiento de *Apache* (asegúrate de que está correctamente configurado y de que tiene el servidor virtual por defecto activado).

3. ***PHP***

   3.1. En el directorio **/var/www/html** crea el fichero **test.php** con el siguiente contenido, Figura 6.20.

```
<H1>Salida de la función phpinfo()</H1>

<?php
  phpinfo();
?>
```

Figura 6.20: Fichero test.php

   3.2. Desde **DesarrolloW7XX** abre un navegador y accede a `http://192.168.1.X7/test.php`, Figura 6.21.

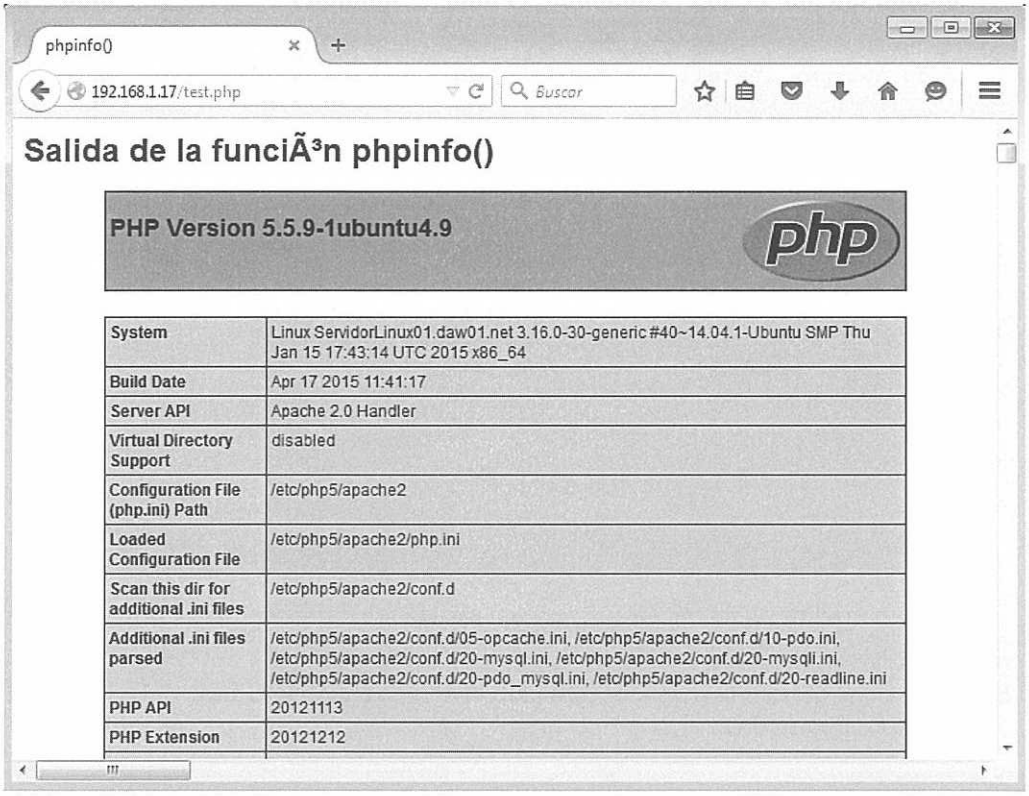


Figura 6.21: Prueba de PHP

4. ***MySQL***

   4.1. Comprueba que el servidor *MySQL* está escuchando peticiones en el puerto **3306/TCP.**

   ```
   netstat -ltn
   ```

   4.2. Recuerda que durante el proceso de instalación definimos una contraseña para el usuario **root**.

   4.3. Utiliza la consola de administración de MySQL para establecer una conexión como usuario **root**.

   ```
   mysql -u root -p
   ```

   4.4. Puedes consultar la tabla de usuarios con el siguiente comando.

   ```
   mysql> select host, user, password from mysql.user;
   ```

   4.5. Solo se puede acceder como usuario **root** desde el equipo local.

5. ***Aplicaciones. PHPMyAdmin***

   5.1. Es posible desplegar (instalar) la aplicación ***PHPMyAdmin*** desde los repositorios de *Ubuntu*.

5.2. Ejecuta el siguiente comando para instalarla.

```
sudo apt-get install phpmyadmin
```

5.3. Selecciona **apache2** y acepta, Figura 6.22.

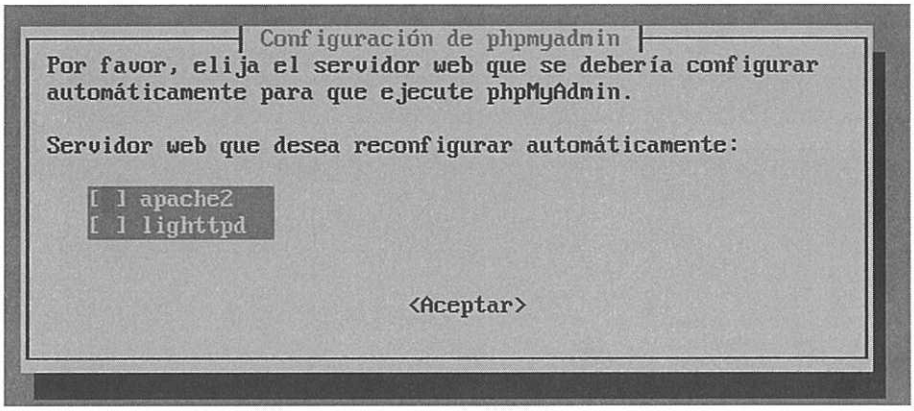


Figura 6.22: Configuración de *PHPMyAdmin* en Apache

5.4. Indica que quieres usar el asistente para crear y configurar la base de datos de *PHPMyAdmin*, Figura 6.23.

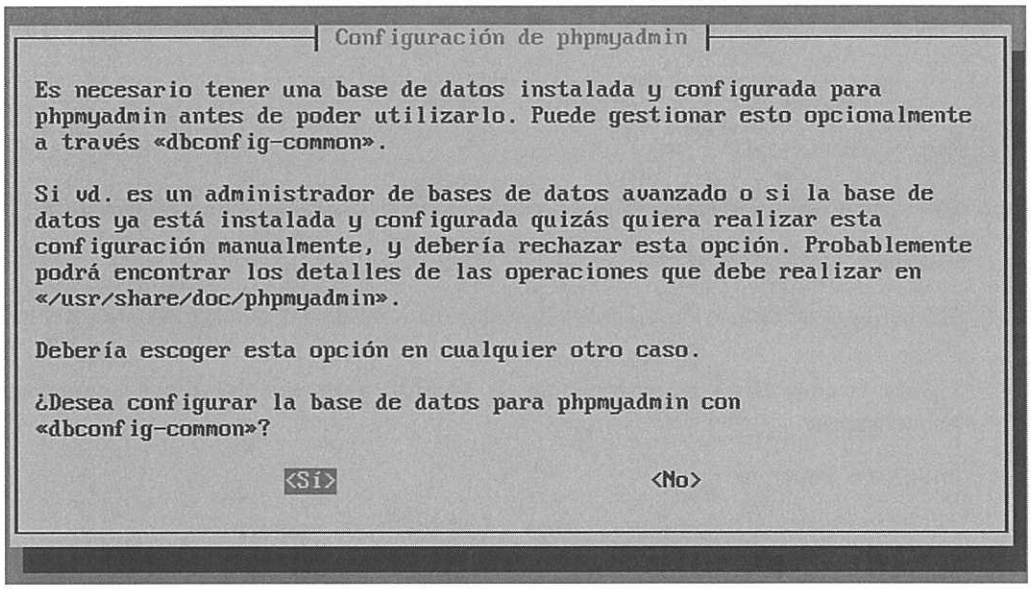


Figura 6.23: Configuración de la base de datos de *PHPMyAdmin*

5.5. Introduce la contraseña del usuario **root** en *MySQL* (pusimos **despliegue**), Figura 6.24.

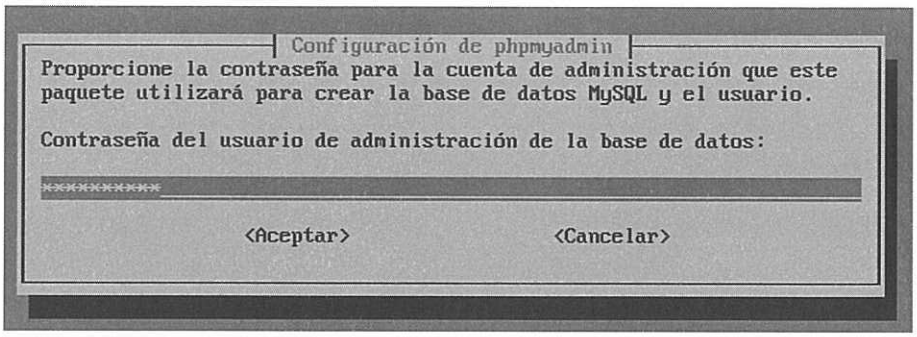


Figura 6.24: Contraseña del usuario root en *MySQL*

5.6. Introduce la contraseña del usuario **phpmyadmin** que se creará en *MySQL* (por ejemplo **phpmyadmin**), Figura 6.25.

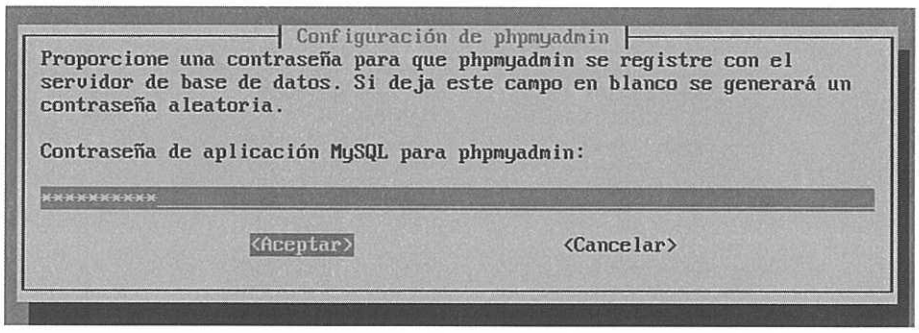


Figura 6.25: Contraseña del usuario phpmyadmin en *MySQL*

5.7. Consulta el contenido del directorio **/etc/phpmyadmin/apache.conf**, Figura 6.26, y observa su configuración.

Figura 6.26: Fichero phpmyadmin.conf

5.8. Crea un enlace simbólico en **/etc/apache2/conf-available** hacía **/etc/phpmyadmin/apache.conf**.

```
sudo ln -s /etc/phpmyadmin/apache.conf /etc/apache2/conf-available
```

5.9. Habilita la nueva configuración con el comando **a2enconf** (creará a un enlace simbólico de **/etc/apache2/conf-available/apache.conf** hacía **/etc/apache2/conf-enable/apache.conf**)

```
sudo a2enconf apache.conf
```

5.10. Reinicia apache.

5.11. Desde **DesarrolloW7XX** abre un navegador y accede a `http://192.168.1.X7/phpmyadmin`, Figura 6.27.

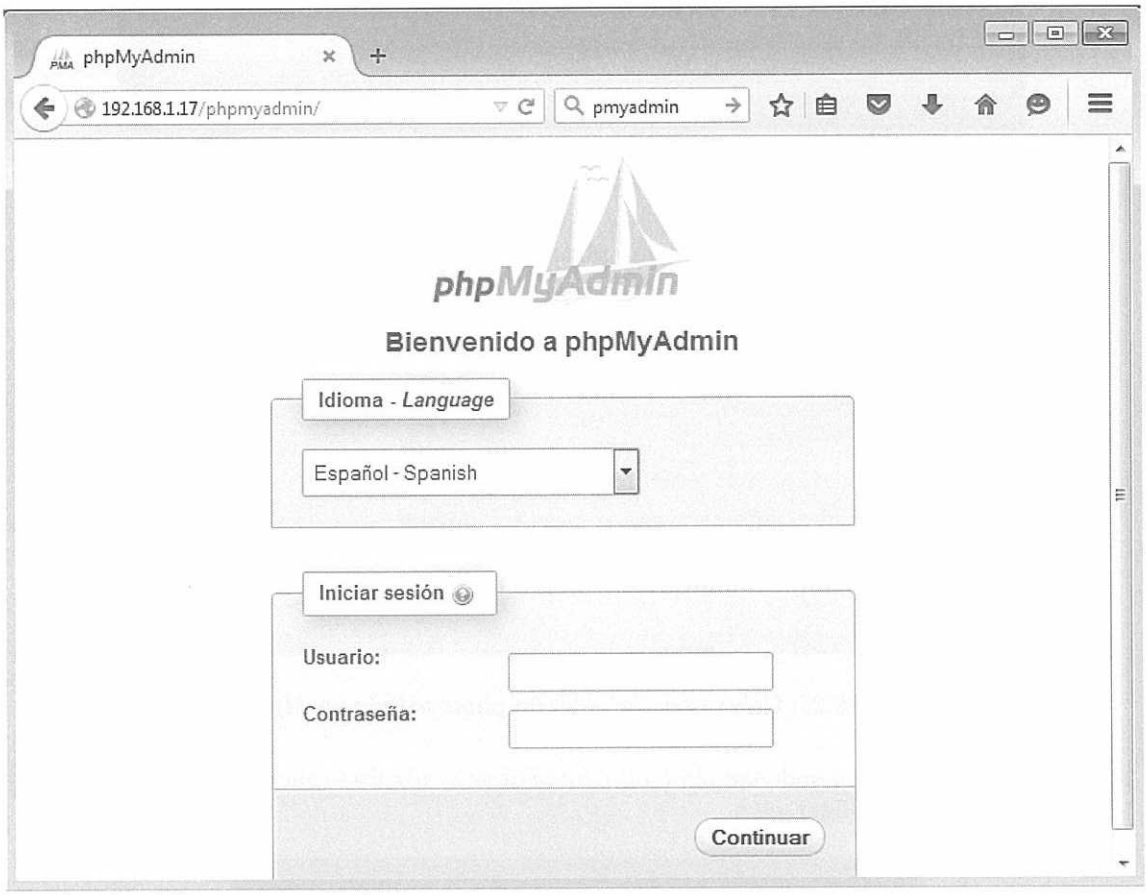


Figura 6.27: Conexión a *phpmyadmim*

6.3. Despliegue en un servidor local

Despliega la aplicación **películas** sobre el servidor WAMP de **DesarrolloW7XX**.

- El acceso a la aplicación se realizará a través de la URL `http://<servidor>/peliculas`.
- El código de la aplicación se almacenará en el directorio **C:\wamp\apps**.

1. **Obtener el código de la aplicación**

   1.1. Inicia sesión con un usuario con privilegios de administrador en **DesarrolloW7XX**.

   1.2. Obtén el código de la aplicación siguiendo las pasos que indique el profesor.

   1.3. Descomprime el fichero obtenido, Figura 6.28.

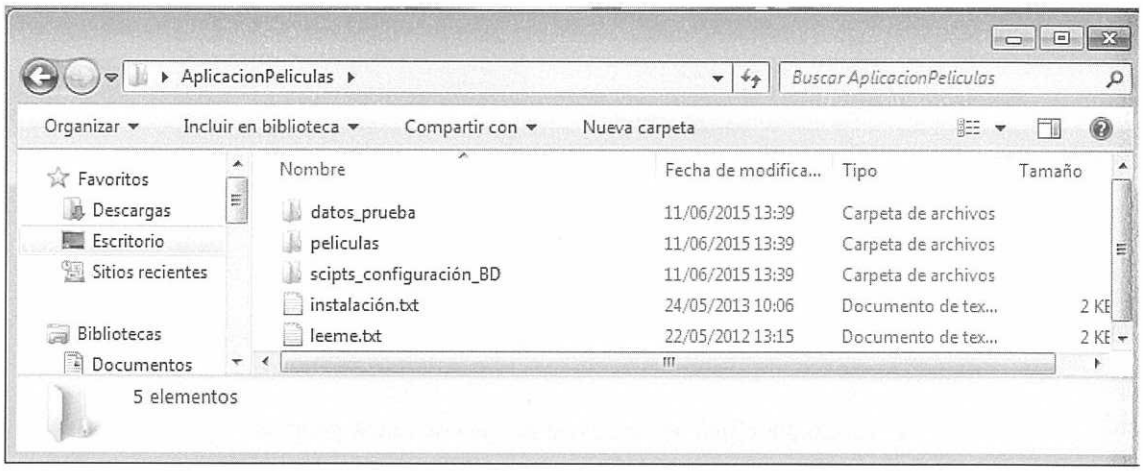


Figura 6.28: Aplicación

   1.4. Lee el contenido del fichero **leeme.txt** donde se explican los ficheros disponibles.

   1.5. Lee el contenido del fichero **instalación.txt** donde se explica cómo instalar la aplicación.

2. **Configuración de la base de datos**

   2.1. Inicia una navegador y accede a `http://localhost/phpmyadmin`.

   2.2. Inicia sesión con el usuario **root**.

   2.3. Accede a **Usuarios**.

   2.4. Pincha en **Agregar un nuevo usuario**.

   2.5. Introduce **peliculas** como nombre de usuario, **localhost** como servidor y **peliculas** como clave del usuario. Marca la opción **Crear base de datos con el mismo nombre y otorgar todos los privilegios** y pincha en **Crear Usuario**, Figura 6.29.

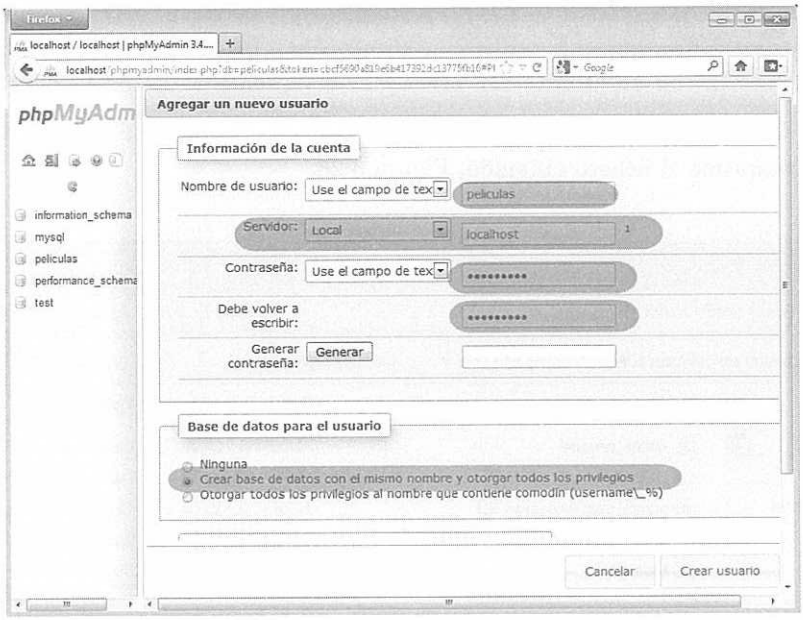


Figura 6.29: Crear el usuario y la base de datos peliculas

2.6. Se han creado el usuario **peliculas**, la base de datos **peliculas** y se han otorgado todos los privilegios al usuario sobre la base de datos, Figura 6.30.

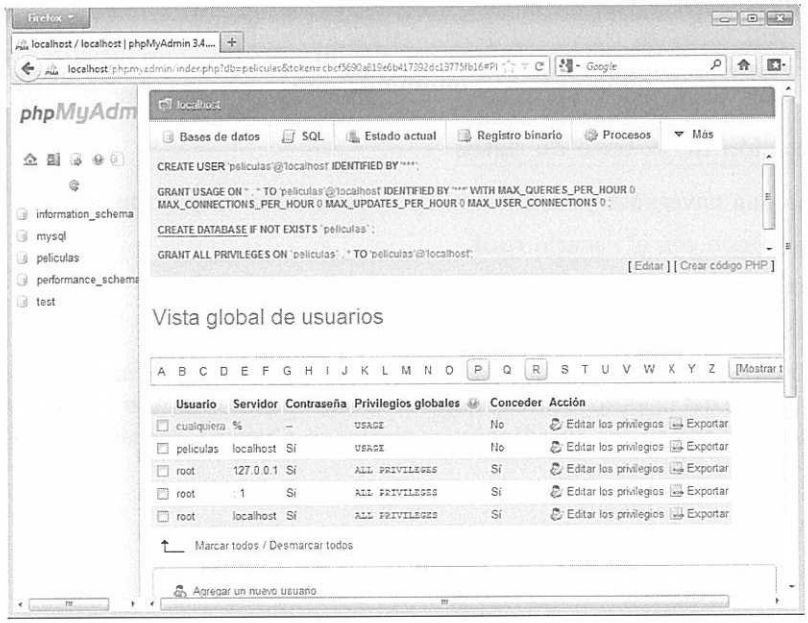


Figura 6.30: Usuario y la base de datos creados

2.7. Cierra la sesión de *phpmyadmin* del usuario **root.**

2.8. Inicia sesión en *phpmyadmin* con el usuario **peliculas**.

2.9. En la parte izquierda selecciona la base de datos **peliculas**.

2.10. Pincha en **Importar**.

2.11. Pincha en **Examinar** y selecciona el script **crear_tablas_peliculas.sql**. Pincha en **Continuar** para ejecutar el script, Figura 6.31.

Figura 6.31: Tablas creadas

2.12. Pincha de nuevo en **Examinar** y selecciona el script **insercion_admin_peliculas.sql**. Pincha en **Continuar** para ejecutar el script.

3. ***Desplegar la aplicación web***

3.1. Copia el directorio **peliculas** dentro del directorio **C:\wamp\apps**.

3.2. Crea en *Apache* el alias **/peliculas** que referencie a **C:\wamp\apps\peliculas**. Lo puedes hacer con la herramienta que ofrece el servidor *WAMP* o creando el fichero **peliculas.conf** dentro del directorio **c:\wamp\alias**, Figura 6.32 (repasa la práctica 6.1 si tienes dudas con esta configuración).

```
Alias /peliculas "c:/wamp/apps/peliculas/"


<Directory "c:/wamp/apps/peliculas/">
   Options Indexes FollowSymLinks MultiViews
   AllowOverride all
   Require all granted
</Directory>
```

Figura 6.32: Crear el alias /peliculas

3.3. Reinicia *Apache* para que se apliquen los cambios.

4. **Acceso a la aplicación**

4.1. Abre un navegador y accede a `http://localhost/peliculas`, Figura 6.33.

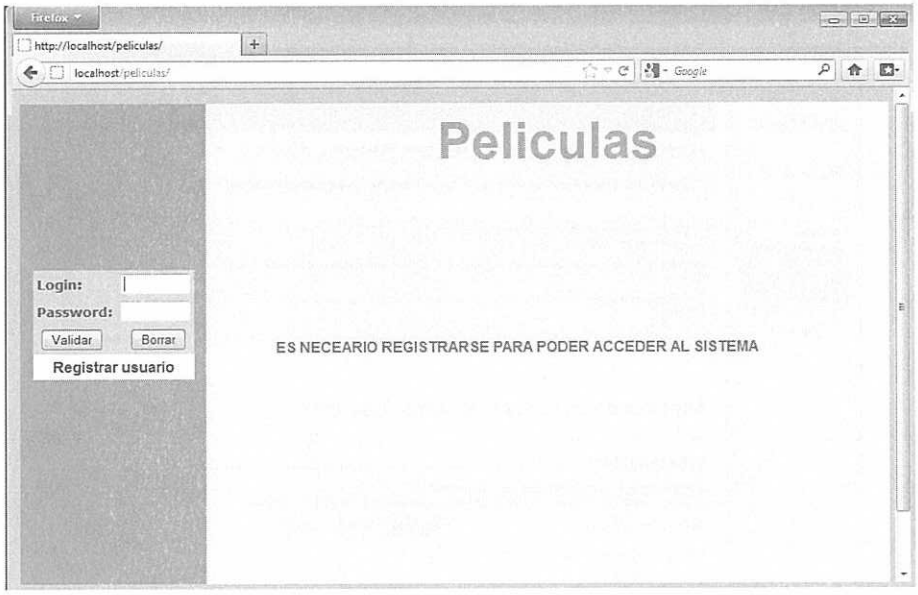


Figura 6.33: Conexión a *peliculas*

4.2. Inicia sesión con el usuario **admin** (contraseña **admin**), Figura 6.34.

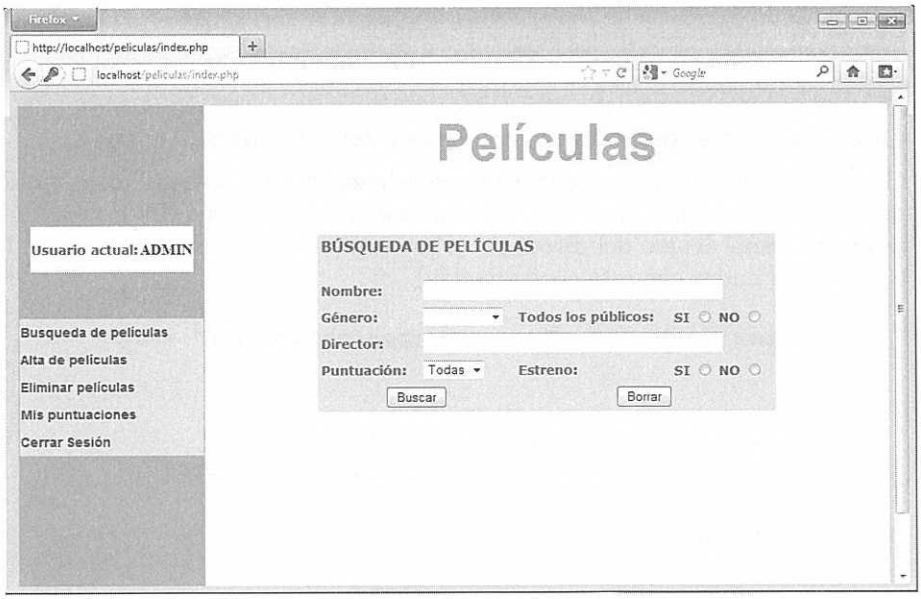


Figura 6.34: Inicio de sesión

6.4. Servidor *vsftpd* en *Linux*

Instala el servidor FTP *vsftpd* (`http://vsftpd.beasts.org/`) en la máquina **ServidorLinuxXX** y configúralo con las siguientes opciones:

- Se permitirá la conexión de usuarios anónimos.
- Se permitirá la conexión a los usuarios locales.
- Los usuarios locales podrán descargar y subir archivos.

1. **Instalación**

   1.1. Inicia una sesión en **ServidorLinuxXX** con un usuario con privilegios de administración.

   1.2. Instala el servidor desde los repositorios oficiales de *Ubuntu*.

   ```
   sudo apt-get update
   sudo apt-get install vsftpd
   ```

   Al instalar el servidor se crean:

   - Los archivos de configuración.
   - El usuario **ftp** que se incluye en el grupo **ftp**.
   - El directorio **/srv/ftp**
     - Su propietario es el usuario **root** y su grupo es **ftp**.
     - El directorio predeterminado de los usuarios anónimos.

   1.3. Comprueba que el servidor está iniciado y escuchando peticiones en el puero 21/TCP.

   ```
   ps -ef | grep vsftpd
   netstat -ltn
   ```

2. **Configuración por defecto**

   La configuración del servidor por defecto es la siguiente:

   - Permite solo el acceso a usuarios locales.
     - Pueden descargar archivos.
     - No pueden subir archivos.
   - Los usuarios anónimos están "enjaulados" en **/srv/ftp**.
     - Nombre de usuario **anonymous** o **ftp**.
     - *Password*: en blanco.
   - El fichero de *logs* por defecto es **/var/log/vsftpd.log**.

   2.1. Consulta el fichero de configuración de servidor (**/etc/vsftpd.conf**) y analiza las directivas habilitadas, Figura 6.35.

   - Está deshabilitado el acceso a los usuario anónimos (directiva `anonymous_enable`).
   - Está habilitado el acceso a los usuarios locales (directiva `local_enable`).
   - No se permite subir archivos al servidor (directiva `write_enable`).

```
listen=YES
#
# Run standalone with IPv6?
# Like the listen parameter, except vsftpd will listen on an IPv6 socket
# instead of an IPv4 one. This parameter and the listen parameter are mutually
# exclusive.
#listen_ipv6=YES
#
# Allow anonymous FTP? (Disabled by default)
anonymous_enable=NO
#
# Uncomment this to allow local users to log in.
local_enable=YES
#
# Uncomment this to enable any form of FTP write command.
#write_enable=YES
#
# Default umask for local users is 077. You may wish to change this to 022,
# if your users expect that (022 is used by most other ftpd's)
#local_umask=022
#
# Uncomment this to allow the anonymous FTP user to upload files. This only
# has an effect if the above global write enable is activated. Also, you will
# obviously need to create a directory writable by the FTP user.
#anon_upload_enable=YES
```

Figura 6.35: Fichero **/etc/vsftpd.conf**

3. <u>Conexión al servidor</u>

   3.1. En **DesarrolloW7XX** inicia el cliente *Filezilla* y establece una conexión al servidor como usuario anónimo (**alumno**), Figura 6.36. Verifica que puedes descargar archivos pero no subirlos.

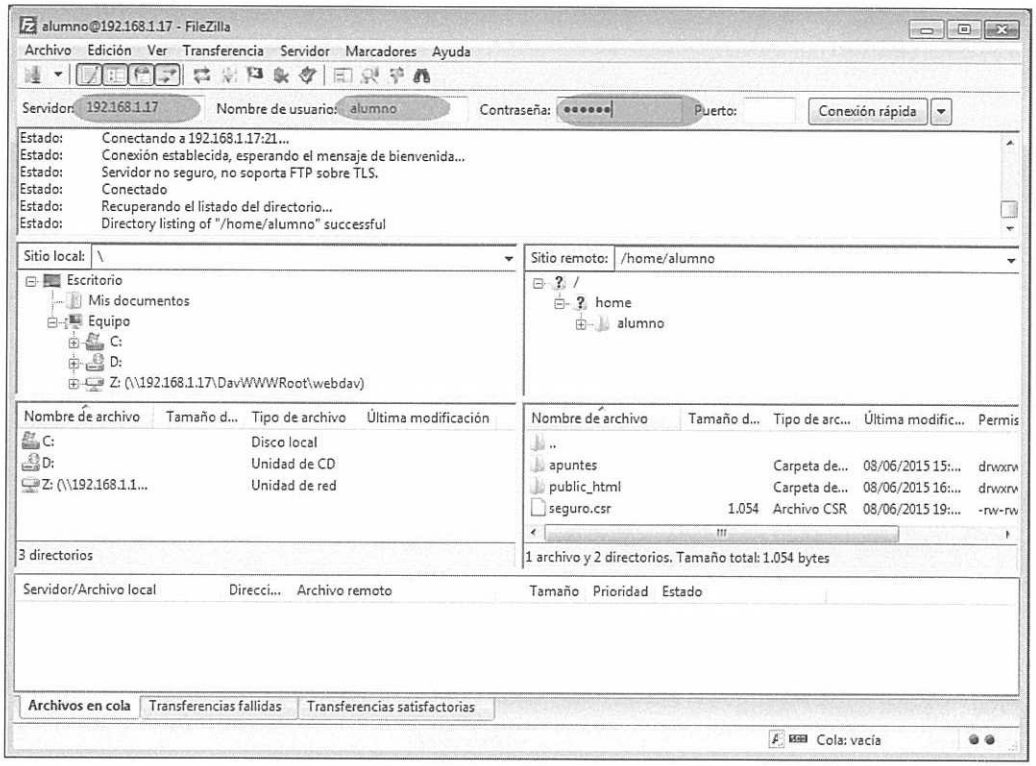


Figura 6.36: Conexión FTP como usuario alumno

   3.2. Verifica que no es posible conectarse como usuario **anonymous**,.

4. <u>Configuración</u>

   4.1. Haz una copia de seguridad del fichero de configuración principal (**/etc/vsftpd.conf**).

   4.2. Edita el fichero de configuración (**/etc/vsftpd.conf**) y habilita las directivas `anonymous_enable` con el valor YES, `local_enable` y `write_enable`, Figura 6.37.

   4.3. Reinicia el servidor para que se apliquen los cambios.

```
sudo service vsftpd stop
sudo service vsftpd start
```

   4.4. En **DesarrolloW7XX** inicia el cliente *Filezilla* y establece una conexión al servidor como usuario **alumno**) y verifica que puedes subir archivos.

   4.5. Crea dos archivos de texto dentro del directorio **/srv/ftp**.

   4.6. En **DesarrolloW7XX** inicia el cliente *Filezilla* y establece una conexión al servidor como usuario **anonymous**, Figura 6.38.

```
#
# Allow anonymous FTP? (Disabled by default)
anonymous_enable=YES
#
# Uncomment this to allow local users to log in.
local_enable=YES
#
# Uncomment this to enable any form of FTP write command.
write_enable=YES
#
```

Figura 6.37: Fichero **/etc/vsftpd.conf**

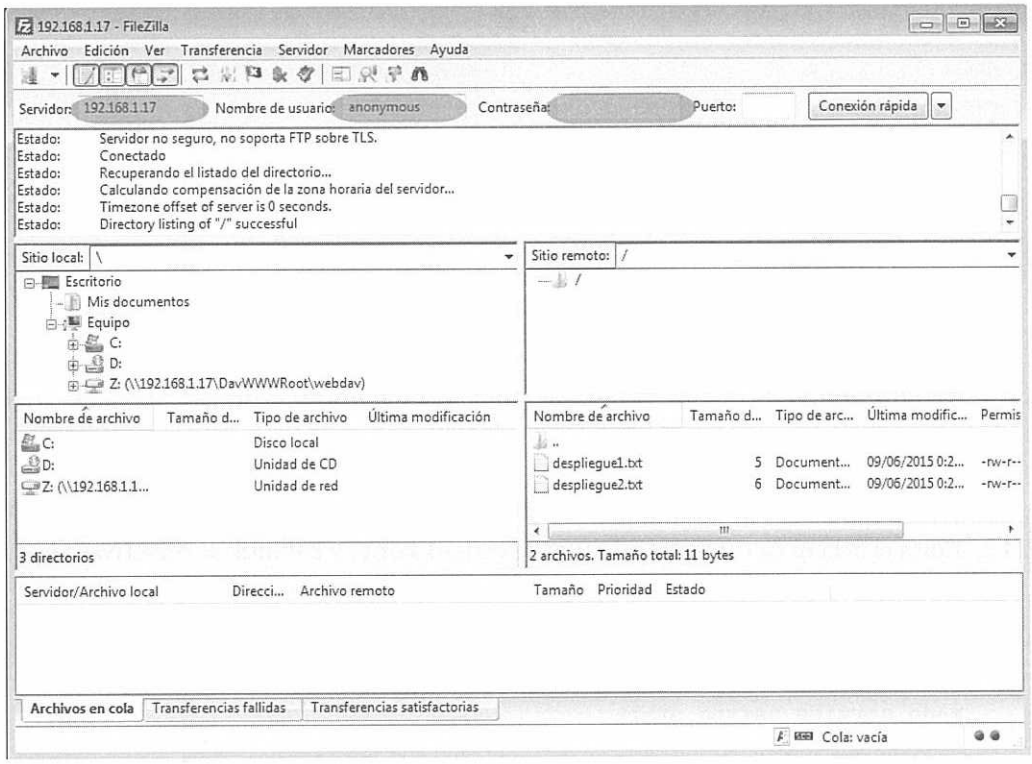


Figura 6.38: Conexión FTP como usuario **anonymous**

6.5. Servidor *OpenSSH* en *Linux*

Instala el servidor *OpenSSH* (`http://www.openssh.com/`) en la máquina **ServidorLinuxXX** para permitir su administración remota.

1. **Instalación**

   1.1. Inicia una sesión en **ServidorLinuxXX** con un usuario con privilegios de administración.

   1.2. Instala el servidor desde los repositorios oficiales de *Ubuntu*.

   ```
   sudo apt-get update
   sudo apt-get install openssh-server
   ```

   Al instalar el servidor:

   - Se crean los ficheros de configuración.
   - Se generan las parejas de claves RSA, DSA y ECDSA que se almacenan en el directorio **/etc/ssh**.

   1.3. Comprueba que el servidor está iniciado y escuchando peticiones en el puerto 22/TCP.

   ```
   ps -ef | grep ssh
   netstat -ltn
   ```

   1.4. Consulta las claves públicas (*.pub) y privadas dentro del directorio **/etc/ssh**, Figura 6.39.

```
alumno@ServidorLinux01:/etc/ssh$ ls -l
total 284
-rw-r--r-- 1 root root 242091 may 12  2014 moduli
-rw-r--r-- 1 root root   1690 may 12  2014 ssh_config
-rw-r--r-- 1 root root   2541 jun  8 16:36 sshd_config
-rw------- 1 root root    668 jun  8 16:36 ssh_host_dsa_key
-rw-r--r-- 1 root root    620 jun  8 16:36 ssh_host_dsa_key.pub
-rw------- 1 root root    227 jun  8 16:36 ssh_host_ecdsa_key
-rw-r--r-- 1 root root    192 jun  8 16:36 ssh_host_ecdsa_key.pub
-rw------- 1 root root    432 jun  8 16:36 ssh_host_ed25519_key
-rw-r--r-- 1 root root    112 jun  8 16:36 ssh_host_ed25519_key.pub
-rw------- 1 root root   1679 jun  8 16:36 ssh_host_rsa_key
-rw-r--r-- 1 root root    412 jun  8 16:36 ssh_host_rsa_key.pub
-rw-r--r-- 1 root root    338 jun  8 16:36 ssh_import_id
alumno@ServidorLinux01:/etc/ssh$
```

Figura 6.39: Claves del servidor SSH

2. **Configuración por defecto**

   2.1. Consulta el fichero de configuración de servidor **/etc/ssh/sshd_config** y analiza las directivas habilitadas.

   2.2. Observa, por ejemplo, que el servidor escucha peticiones en el puerto 22 (directiva `Port`) y que se permite el acceso al usuario root, pero utilizando autenticación por clace pública (no con password) (directiva `PermitRootLogin`).

3. Conexión al servidor

3.1. En **DesarrolloW7XX** inicia el cliente *Putty* y establece una conexión SSH al servidor, Figura 6.40.

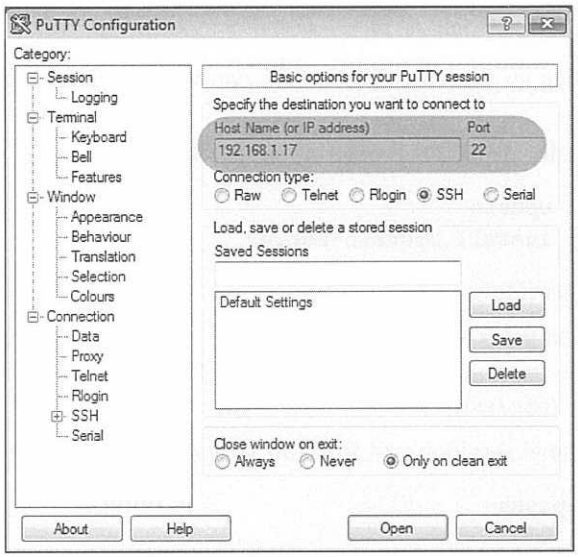


Figura 6.40: Conexión SSH

3.2. En servidor envía un resumen (*fingerprint*) de su clave pública RSA, Figura 6.41.

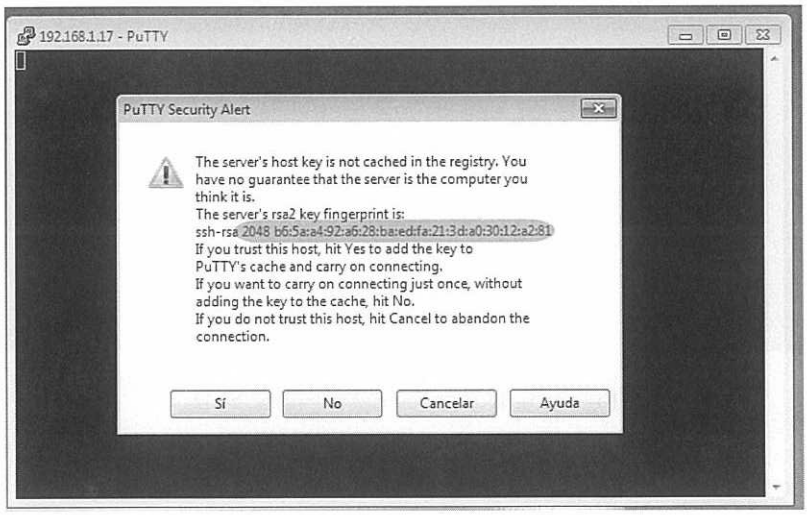


Figura 6.41: *Fingerprint* de la clave pública RSA enviada por el servidor SSH

En este punto debemos comprobar que es realmente el resumen de la clave del servidor para evitar una suplantación de identidad (podemos ir al servidor y ejecutar el comando `ssh-keygen -l -f /etc/ssh/ssh_host_rsa_key`) para obtener el *fingerprint* de la clave), Figura 6.42.

```
alumno@ServidorLinux01:~$ ssh-keygen -l -f /etc/ssh/ssh_host_rsa_key.pub
2048 b6:5a:a4:92:a6:28:ba:ed:fa:21:3d:a0:30:12:a2:81  root@ServidorLinux01.daw01
.net (RSA)
alumno@ServidorLinux01:~$ _
```

Figura 6.42: *Fingerprint* de la clave pública RSA del servidor SSH

El cliente SSH almacena el *fingerprint* de la clave del servidor. En las próximas conexiones ya no pide la aceptación por parte del usuario. Si en una conexión el *fingerprint* enviado por el servidor no coincide con el almacenado por el cliente se avisará al usuario.

3.3. Inicia sesión como usuario alumno, Figura 6.43.

```
alumno@ServidorLinux01: ~
login as: alumno
alumno@192.168.1.17's password:
Welcome to Ubuntu 14.04.2 LTS (GNU/Linux 3.16.0-30-generic x86_64)

 * Documentation:  https://help.ubuntu.com/

  System information as of Mon Jun  8 16:46:33 CEST 2015

  System load:  0.0               Processes:           80
  Usage of /:   12.4% of 9.22GB   Users logged in:     1
  Memory usage: 15%               IP address for eth0: 192.168.1.17
  Swap usage:   0%

  Graph this data and manage this system at:
    https://landscape.canonical.com/

Last login: Mon Jun  8 16:46:34 2015 from 192.168.1.4
alumno@ServidorLinux01:~$
```

Figura 6.43: Conexión SSH como usuario alumno

6.6. Despliegue de aplicaciones en un servidor remoto

Despliega la aplicación **películas** sobre el servidor LAMP de **ServidorLinuxXX**.

- El acceso a la aplicación se realizará a través de la URL `http://<servidor>/peliculas`.
- El código de la aplicación se almacenará en el directorio **/var/www**.

1. **Obtener el código de la aplicación**

   1.1. Inicia sesión con un usuario con privilegios de administrador en **DesarrolloW7XX**.

   1.2. Obtén el código de la aplicación siguiendo las pasos que indique el profesor.

   1.3. Descomprime el fichero obtenido, Figura 6.44.

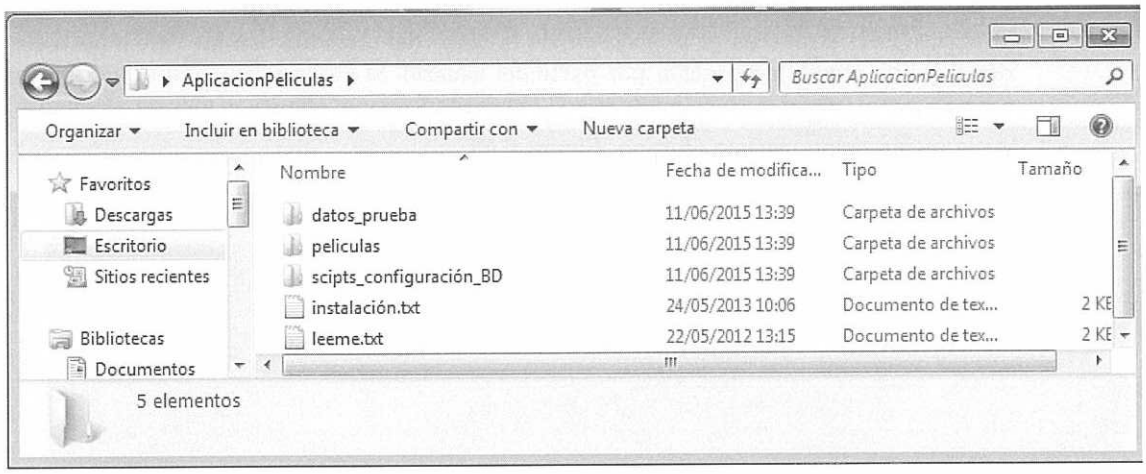


Figura 6.44: Aplicación

   1.4. Lee el contenido del fichero **leeme.txt** donde se explican los ficheros disponibles.

   1.5. Lee el contenido del fichero **instalación.txt** donde se explica cómo instalar la aplicación.

2. **Configuración de la base de datos**

   2.1. Inicia una navegador y accede a `http://192.168.1.X7/phpmyadmin` o `http://servidorlinuxXX.daw01.net/phpmyadmin`.

   2.2. Inicia sesión con el usuario **root**.

   2.3. Accede a **Privilegios**.

   2.4. Pincha en **Agregar un nuevo usuario**.

   2.5. Introduce **peliculas** como nombre de usuario, **localhost** como servidor y **peliculas** como clave del usuario. Marca la opción **Crear base de datos con el mismo nombre y otorgar todos los privilegios** y pincha en **Crear Usuario**, Figura 6.45.

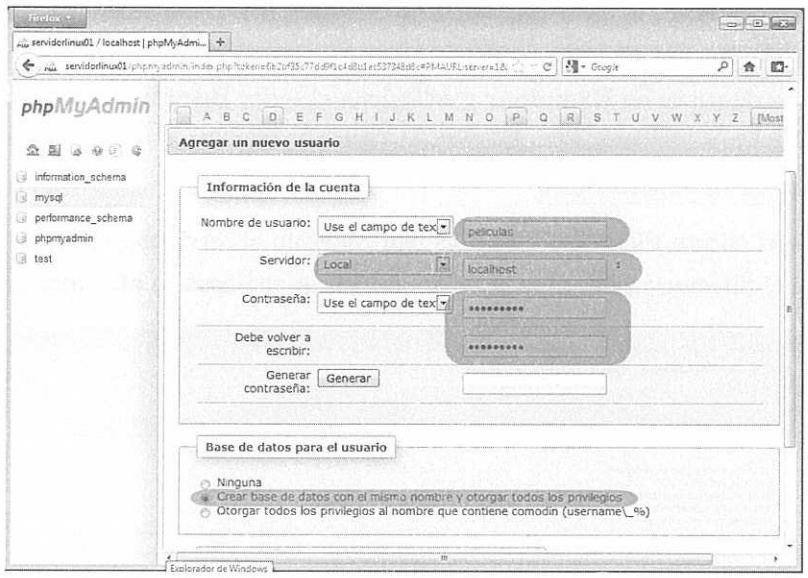


Figura 6.45: Crear el usuario y la base de datos peliculas

2.6. Se han creado el usuario **peliculas**, la base de datos **peliculas** y se han otorgado todos los privilegios al usuario sobre la base de datos, Figura 6.46.

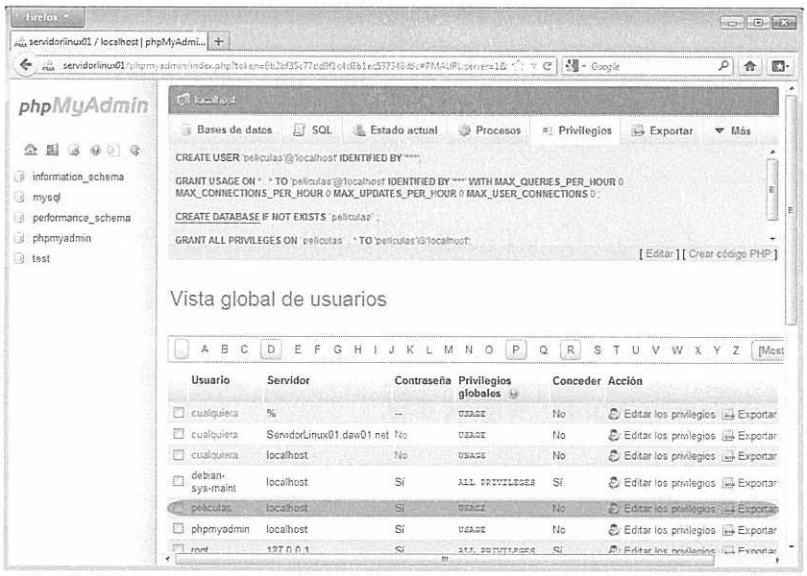


Figura 6.46: Usuario y la base de datos creados

2.7. Cierra la sesión de *phpmyadmin* del usuario **root.**

2.8. Inicia sesión en *phpmyadmin* con el usuario **peliculas.**

2.9. En la parte izquierda selecciona la base de datos **peliculas.**

2.10. Pincha en **Importar.**

2.11. Pincha en **Examinar** y selecciona el script **crear_tablas_peliculas.sql**. Pincha en **Continuar** para ejecutar el script.

2.12. Pincha de nuevo en **Examinar** y selecciona el script **insercion_admin_peliculas.sql**. Pincha en **Continuar** para ejecutar el script.

3. ***Desplegar la aplicación web***

3.1. Inicia el cliente *Filezilla* y establece una conexión al servidor como usuario **alumno**.

3.2. Sube el directorio **peliculas** al directorio **home** del usuario **alumno**, Figura 6.47.

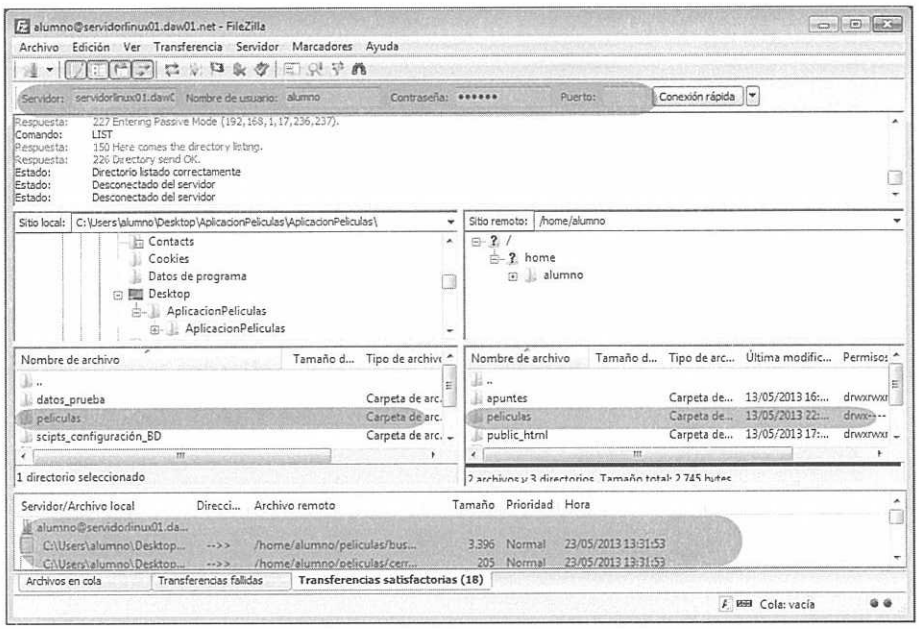


Figura 6.47: Conexión FTP como usuario alumno

3.3. Inicia el cliente SSH *Putty* y establece una conexión SSH al servidor como usuario alumno.

3.4. Mueve el directorio **peliculas** desde del directorio **home** del usuario **alumno** al directorio **/var/www/html/.** , Figura 6.48.

```
sudo mv /home/alumno/peliculas /var/www/html
```

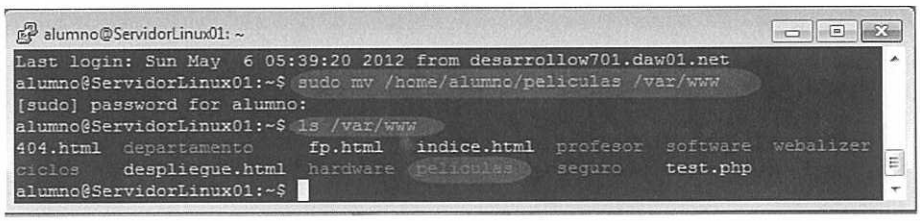


Figura 6.48: Conexión SSH

3.5. Cambia el propietario y el grupo del directorio **/var/www/html/peliculas** y todo su contendido al usuario **root** y grupo **root**.

```
sudo chown root:root -R /var/www/html/peliculas
```

3.6. Cambia los permisos del directorio **/var/www/html/peliculas** para que *Apache* pueda leer los ficheros.

```
sudo chmod 755 /var/www/html/peliculas
sudo chmod 644 -R /var/www/html/peliculas/*
sudo chmod 755 /var/www/html/peliculas/css
sudo chmod 755 /var/www/html/peliculas/img
```

3.7. Edita el fichero **/etc/apache2/sites-availables/000-default.conf** y añade las siguientes directivas (muy importante la directiva `DirectoryIndex index.php`), Figura 6.49.

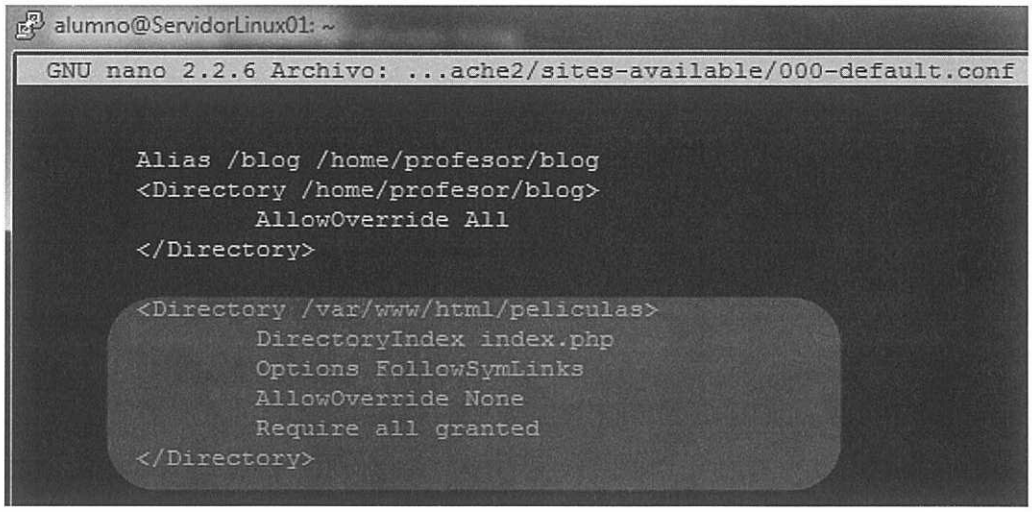


Figura 6.49: Fichero/**etc/apache2/sites-availables/000-default.conf**

3.8. Reinicia *Apache*.

4. **Acceso a la aplicación**

4.1. Abre un navegador y accede a `http://servidorlinuxXX.dawXX.net/peliculas`, Figura 6.50.

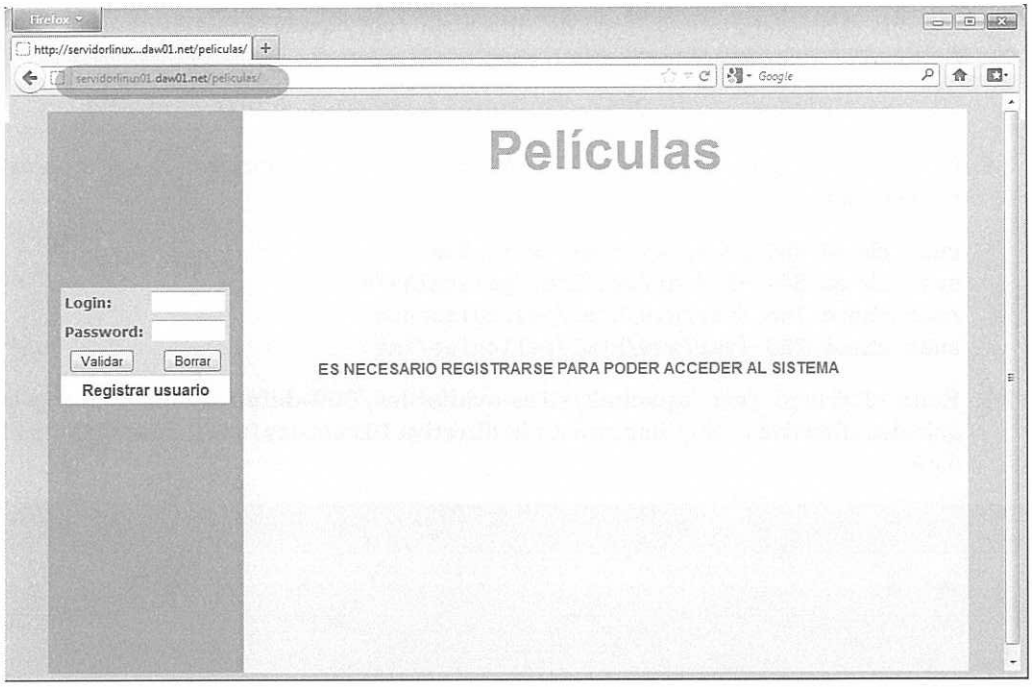


Figura 6.50: Conexión a **peliculas**

6.7. Despliegue en un servidor de *hosting*

- Busca información sobre alguna empresa/web que ofrezca servicios de alojamiento web (*web hosting*) con *Apache* y con soporte PHP y MySQL.
- Crea una cuenta en el servidor de *hosting* y despliega la aplicación **peliculas**.

1. **Crear una cuenta en el servidor de alojamiento**

   Existen múltiples empresas que ofrecen servicios de *hosting*. Para la resolución de la práctica hemos elegido `http://www.000webhost.com/` pero podrías elegir otra, el proceso sería similar.

   1.1. Inicia sesión con un usuario con privilegios de administrador en **DesarrolloW7XX**.

   1.2. Abre un navegador y accede a `http://www.000webhost.com/`.

   1.3. Observa los planes de alojamiento que se ofrecen y analiza las características del plan gratuito (*Free hosting*), Figura 6.51.

   1.4. Pincha en ***Order Now*** para crear una cuenta.

   1.5. Como no disponemos de un nombre de dominio propio usaremos el subdominio gratis que nos ofrecen, Figura 6.52.

   1.6. Una vez confirmada la petición accede a tu cuenta de correo para verificar el registro.

   1.7. Cuando confirmes la petición podrás acceder a tu panel de control, Figuras 6.53 y 6.54.

	» Free Hosting	» Premium Hosting
Price	$0.00	$4.84 / month
Disk Space	1500 MB	Unlimited Disk Space!
Data Transfer	100 GB / month	Unlimited Data Transfer!
Add-on Domains	5	Unlimited
Sub-domains	5	Unlimited
E-mail Addresses	5	Unlimited
MySQL Databases	2	Unlimited
Free domain yourname.COM, .NET, .ORG, .INFO, .CO.UK	✗	✓
Control Panel	Custom Panel	cPanel Pro, see demo
Reseller Hosting Feature	✗	✓
	Order Now	Order Now
Simple Site Builder	✓	✓
Advanced Site Builder	✗	✓
Support by Phone, Live Chat	✗	✓
Assistance in Installing Scripts	✗	✓
Help in Developing Your Website	✗	✓
Backups	✓	✓
Automated Weekly Backups	Limited	✓
Uptime	99%	99.9%
FTP Accounts	1	Unlimited
Anonymous FTP Access	✗	✓
CGI Scripts	✗	✓
PHP	✓	✓
Ruby On Rails	✗	✓
SSH Access	✗	✓

Figura 6.51: *Free hosting*

or, I will choose your free subdomain (recommended)

www. [despliegue] .web44.net

Figura 6.52: Subdominio

List of your domains

» Domain	» Status	» Action
despliegue.web44.net	Active	Go to CPanel

Figura 6.53: Acceder al panel de control

Figura 6.54: Panel de control

1.8. Abre otra ventana del navegador y accede a tu dominio (en el ejemplo `http://despliegue.web44.net/`) para ver lo que hay por defecto en el directorio raíz del servidor web, Figura 6.55.

1.9. En el panel de control accede a ***File Manager***, Figura 6.56.

1.10. Inicia sesión en ***File Manager*** (usa la misma clave que usaste en tu registro) y accede a **public_html** (la carpeta raíz de tu servidor web), Figura 6.57.

1.11. Consulta tu cuenta de correo y observa que has recibido varios mails con información de los servicios (ftp, mysql, ...) y los usuarios y contraseñas que se han activado en cuenta, Figura 6.58.

1.12. Inicia el cliente *Filezilla* y establece una conexión al servidor FTP (en el ejemplo `ftp.despliegue.web44.net`) con el usuario y password que te han proporcionado, Figura 6.59.

1.13. En el panel de control accede a ***phpMyadmin***, Figura 6.60.

1.14. Observa que nos informa de que no hay ninguna base de datos creada, Figura 6.61.

1.15. En el panel de control accede a ***MySQL***, Figura 6.62.

1.16. Desde aquí podremos crear nuestras bases de datos, Figura 6.63.

2. <u>**Configuración de la base de datos**</u>

2.1. Crea una base de datos para la aplicación películas, Figura 6.64 y observa los datos de conexión.

2.2. Pincha en ***Import***.

2.3. Pincha en **Seleccionar un archivo** y selecciona el script **crear_tablas_peliculas.sql**. Pincha en ***Go*** para ejecutar el script.

2.4. Pincha de nuevo en **Seleccionar un archivo** y selecciona el script **insercion_admin_peliculas.sql**. Pincha en ***Go*** para ejecutar el script.

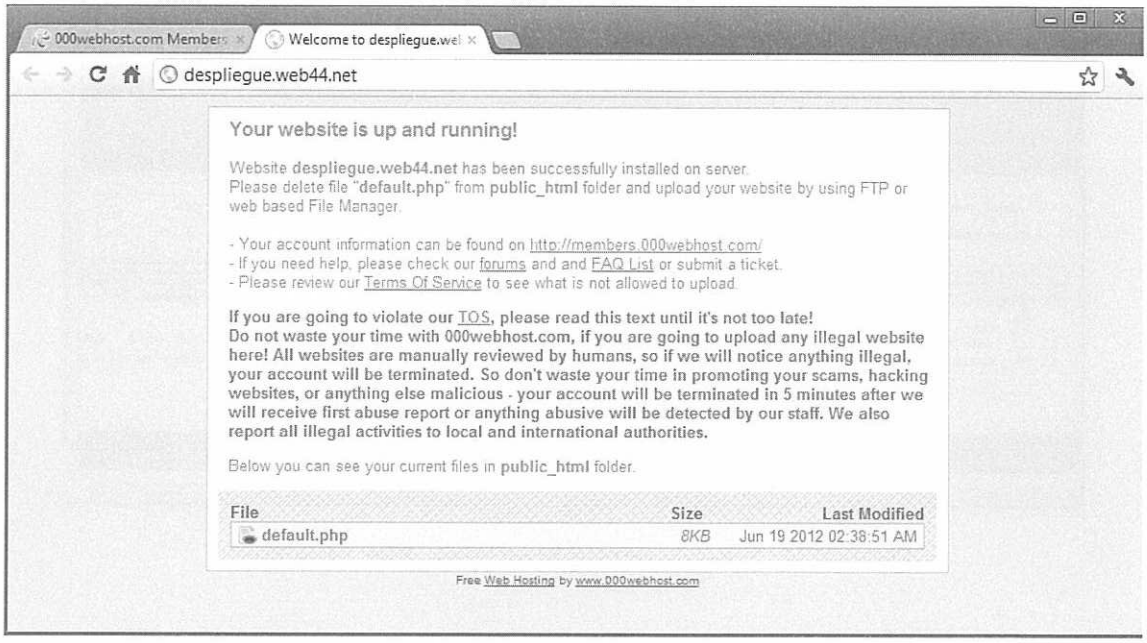


Figura 6.55: Acceso web

Figura 6.56: Acceso a ***File Manager***

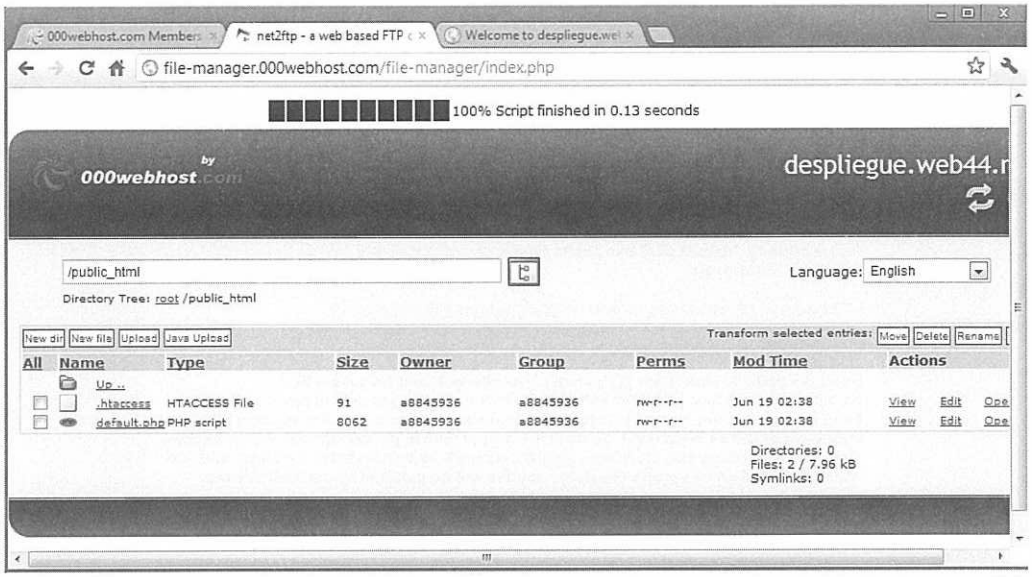


Figura 6.57: Acceso a **public_html** desde ***File Manager***

We have just setup 'despliegue.web44.net' account.

Note: If you have chosen to host your own domain you must update your nameservers to ns01.000webhost.com, ns02.000webhost.com
You will see message 'Your website is up and running!' after visiting your domain once it will start working (usually in 24 hours after nameserver update).

Visit your website: http://despliegue.web44.net
Access control panel at: http://members.000webhost.com/ (login with your email and password)

*** General Account Details ***
Domain: despliegue.web44.net
IP Address: 31.170.160.74
Username: a8845936
Password: ******

*** File Upload Details ***
FTP Hostname: ftp.despliegue.web44.net or 31.170.160.74
FTP Username: a8845936
FTP Passsword: ******

Files must be uploaded to 'public_html' folder.
If you are not using FTP, you can access web based file manager from control panel.

Figura 6.58: Correo con información de los servicios activados

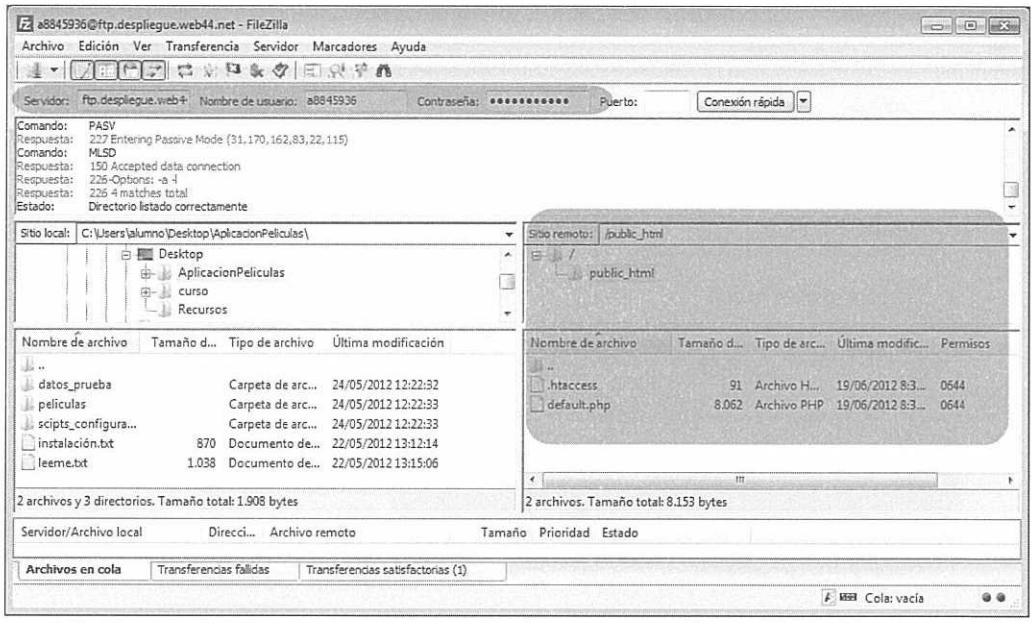


Figura 6.59: Conexión FTP

Figura 6.60: Acceso a ***phpMyadmin***

Figura 6.61: ***phpMyadmin***

Figura 6.62: Acceso a ***MySQL***

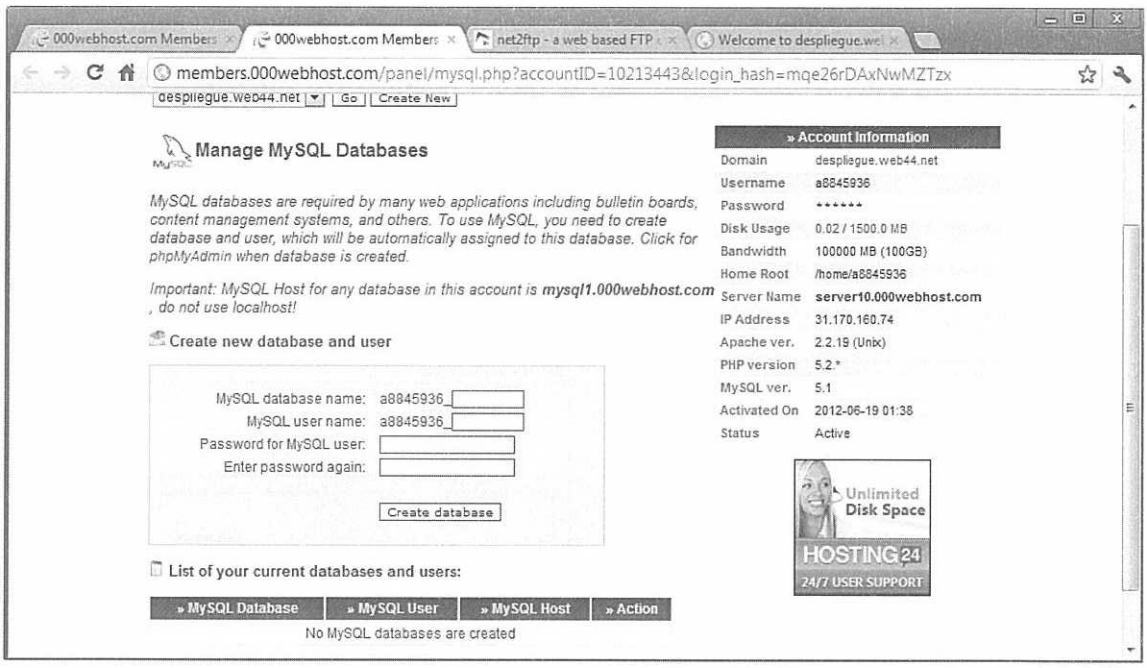


Figura 6.63: ***MySQL***

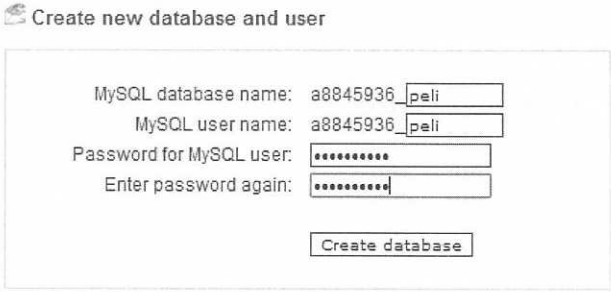


Figura 6.64: Crear la base de datos

3. ***Desplegar la aplicación web***

   3.1. Inicia el cliente *Filezilla* y establece una conexión al servidor FTP con el usuario y password que te han proporcionado.

   3.2. Borra el fichero *default.php* del directorio **public_html**.

   3.3. Sube el contenido del directorio **peliculas** al directorio **public_html**, Figura 6.65.

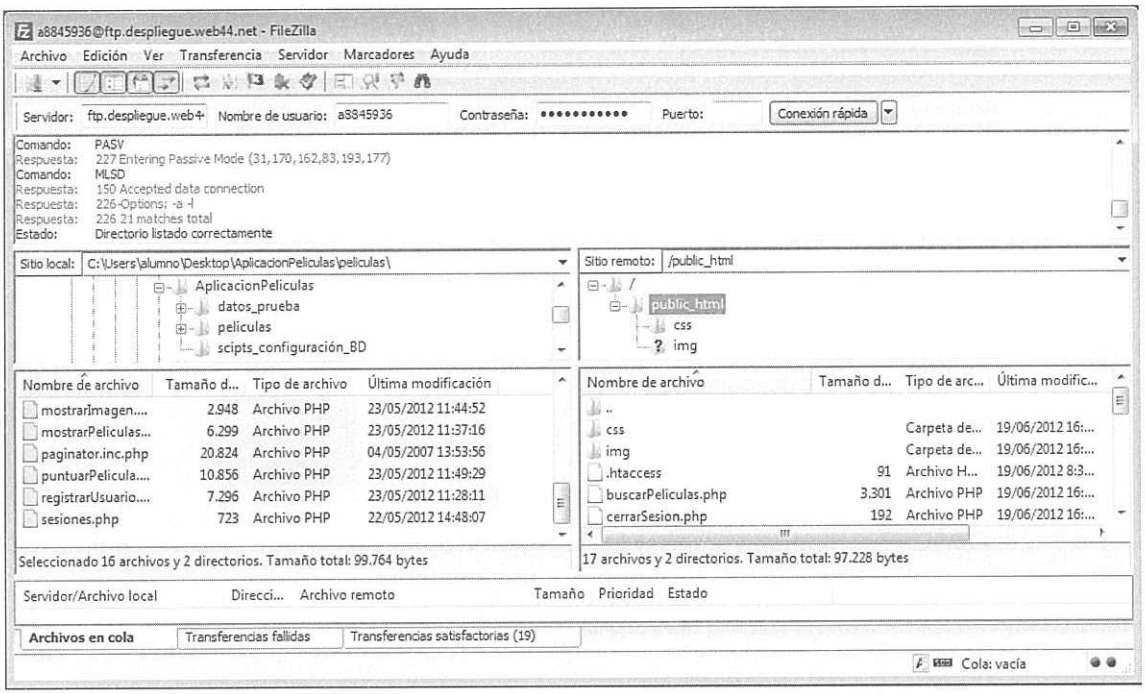


Figura 6.65: Conexión FTP

   3.4. En el panel de control accede a ***File Manager***, inicia sesión.

   3.5. Edita el fichero **configuración.php** y modifica los parámetros de conexión a la base de datos, Figura 6.66.

4. **Acceso a la aplicación**

   4.1. Accede a la aplicación, Figura 6.67.

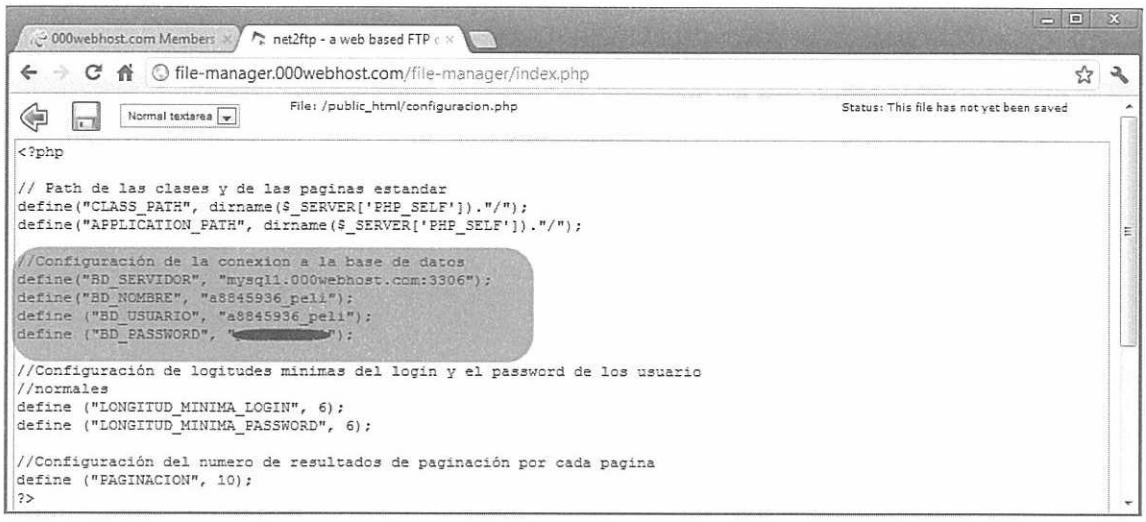


Figura 6.66: Fichero **configuración.php**

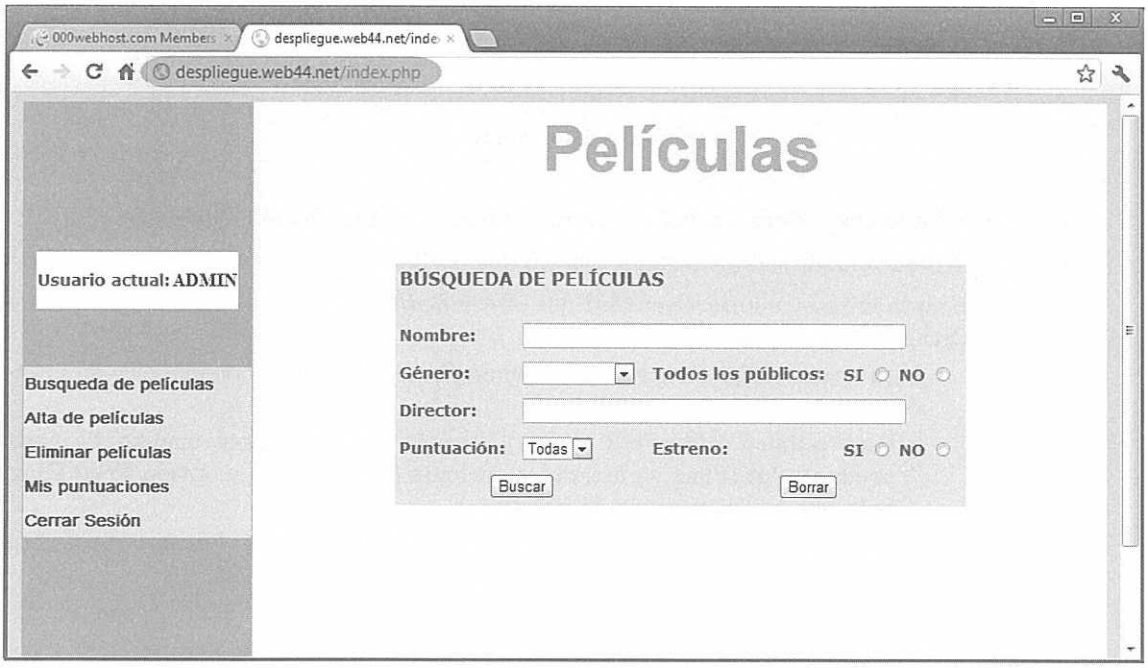


Figura 6.67: Conexión a **películas**

6.8. Despliegue en la nube (OpenShift) - 1

En esta práctica utilizaremos el servicio que ofrece *OpenShift* (`https://www.openshift.com/`) para desplegar una aplicación PHP en la "nube"(*cloud*).

1. **Introducción a *OpenShift***

   OpenShift es una plataforma *PaaS (Platform as a Service)* desarrollada por *Red Hat*, como proyecto *Open Source* (todo el codigo del *OpenShift* esta disponible en *GitHub*, bajo la licencia *Apache 2.0*). Los desarrolladores pueden construir, probar y ejecutar sus aplicaciones, Figura 6.68.

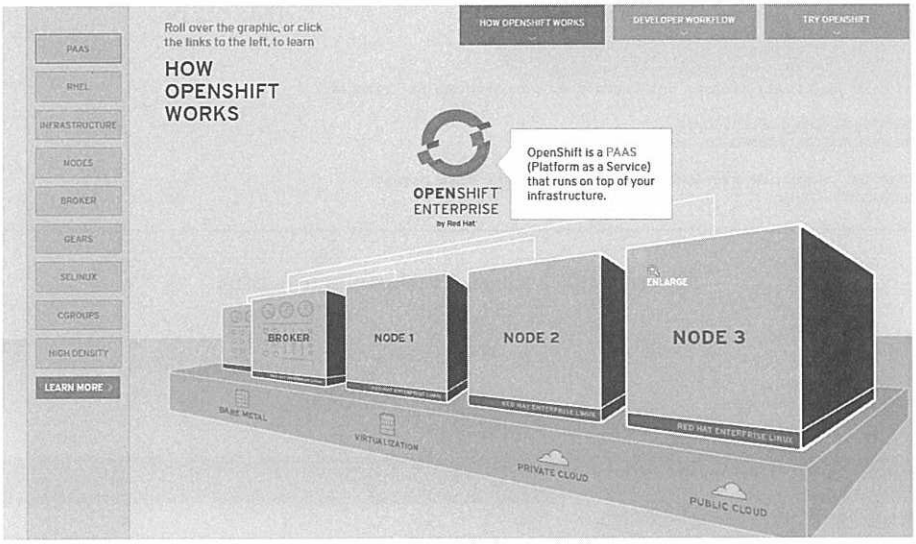


Figura 6.68: *Stack*

1.1. Accede a `https://www.openshift.com/` y observa la información disponible.

1.2. Como desarrollador/administrador existen dos posibilidades.

- Usar la infraestructura *OpenShift* que ofrece *RedHat* para desplegar las aplicaciones (`https://www.openshift.com`).
  - Para usar la infraestructura de *OpenShift*, *RedHat* ofrece dos modelos de negocio.
    - PaaS público (*OpenShift Online*). Accede a `https://www.openshift.com/products/pricing` y observa las opciones disponibles. La opción **Free Plan** es la que utilizaremos en la práctica.
    - PaaS privado (*OpenShift Enterprise*). Accede a `https://www.openshift.com/products/enterprise` para obtener mas información.
- Montar un PaaS privado propio utilizando ***OpenShift Origin*** (`http://openshift.github.io`), el proyecto *OpenSource* que engloba los distintos componentes del *PaaS OpenShift*.
  - Accede a `https://www.openshift.com/products/origin` o a `http://openshift.github.io` y observa cómo puedes descargar el código de *OpenShift Origin* para montar tu propio PaaS privado.

Ofrece una amplia selección de lenguajes de programación y *frameworks* incluyendo *Java, PHP, Ruby, Python y Node.js*. También proporciona herramientas de desarrollo integrado para soportar el ciclo de vida de las aplicaciones incluyendo integración con *Eclipse, JBoss Developer Studio, Jenkins, Maven y GIT*.

2. **Arquitectura y componentes**

OpenShift proporciona espacio en disco, CPU, memoria, conexión de red y servidores (*Apache, JBoss, Tomcat, ...*). Dependiendo del tipo de aplicaciones a construir/desplegar se encuentran disponibles plantillas de recursos y software.

A continuación se describen brevemente los principales componentes de *OpenShif*. Para ampliar lo aquí explicado se recomienda consultar el manual de usuario `https://access.redhat.com/site/documentation/en-US/OpenShift/2.0/html/User_Guide/`.

- **Cuenta de usuario**. Para poder utilizar la infraestructura de PaaS público (*OpenShift Online*) es necesario resgitrarse y crear una cuenta de usuario.
- ***Namespace* (Espacio de nombres)**. Cada cuenta de usuario tiene asociado un espacio de nombres que se añadirá a un un dominio DNS (**rhcloud.com** o un nombre de dominio propio si lo tenemos registrado) para generar un nombre FQDN que permita el acceso las aplicaciones que se creen con esa cuenta de usuario. Por ejemplo, si nos registramos con el usuario **alumno** y creamos el *Namespace* **daw01**, el nombre DNS que se utilizará como base para nombrar a las aplicaciones será **daw01.rhcloud.com**.
- **Aplicaciones**. *OpenShif* permite desplegar y ejecutar aplicaciones desarrolladas en diferentes lenguajes y tecnologías.
  - **Nombre DNS de la aplicación**. Cuando se crea una aplicación, se registra un nombre de dominio DNS con la combinación del nombre de la aplicación y el espacio de nombres (*namespace*). Por ejemplo, si creamos la aplicación **peliculas** el nombre DNS que se utilizará será **peliculas-daw01.rhcloud.com**.
  - **Repositorio Git**. Por cada aplicación se crea un repositorio Git al que solo puede acceder el propietario de la aplicación. *Openshift* accede al repositorio de Git para obtener el código de la aplicación y desplegarlo en el servidor correspondiente. Los desarrolladores tienen que subir el código de sus aplicaciones al repositorio.
- ***Cartridges***. Software/*frameworks* que proporcionan el entorno y la funcionalidad necesarias para ejecutar aplicaciones. Existen númerosos *cartridges* disponibles para soportar diferentes tecnologías. De forma genérica es posible clasificar los *cartridges* en:
  - ***Web***. Reciben y gestionan las peticione a las aplicaciones web. Algunos ejemplos son ***PHP 5.3, JBoss Application Server 7.1, Python 2.6, Tomcat 7 (JBoss EWS 2.0), ...***
  - ***Addon***
    - ***Database***. Proporcionan un SGBD. Algunos ejemplos son ***MySQL Database 5.1, MongoDB NoSQL Database 2.2, PostgreSQL Database 8.4, ...***
    - ***Database management***. Proporcionan software para gestionar los SGBD. Algunos ejemplos son ***phpMyAdmin, RockMongo, ...***
    - ***Monitoring and Management*** . Proporciona software para gestionar y monitorizar las aplicaciones. Algunos ejemplos son ***Cron task scheduler, Jenkins Client, ...***

Por ejemplo, para la aplicación **peliculas** necesitaríamos los *cartridges* de **PHP** y **MySQL**

- ***Gears***. Contenedores de recursos donde se pueden ejecutar uno o varios *cartridges*. Limitan la cantidad de memoria y espacio de disco disponible para los *cartridges*. Actualmente se proporcionan tres tipos de *gear*:
  - ***Small***. Ofrece 512MB de RAM, 100MB de swap, y 1GB de espacio de disco.
  - ***Medium***. Ofrece 1GB de RAM, 100MB de swap, y 1GB de espacio de disco.
  - ***Large***. Ofrece 2GB de RAM, 100MB de swap y 1GB de espacio de disco.

  Por defecto, con una cuenta gratuita, se pueden usar 3 *gears small* (un total 1.5GB de RAM y 3GB de espacio de disco). Es posible asignar los 3 *gears* una solo aplicación, distribuir varias aplicaciones en cada *gear* e incluso crear aplicaciones escalables que utilicen los recursos de los *gears* según las necesidades. Podemos ver a los *gears* como servidores virtuales.
- ***Nodes***. Máquinas físicas o virtuales donde se ejecutan uno o múltiples *gears*.

3. <u>Interfaces de usuario</u>

   Es posible gestionar los servicios (para desarrollar y desplegar aplicaciones) que ofrece *Openshift* usando diferentes interfaces (`https://www.openshift.com/get-started/`, Figura 6.69.

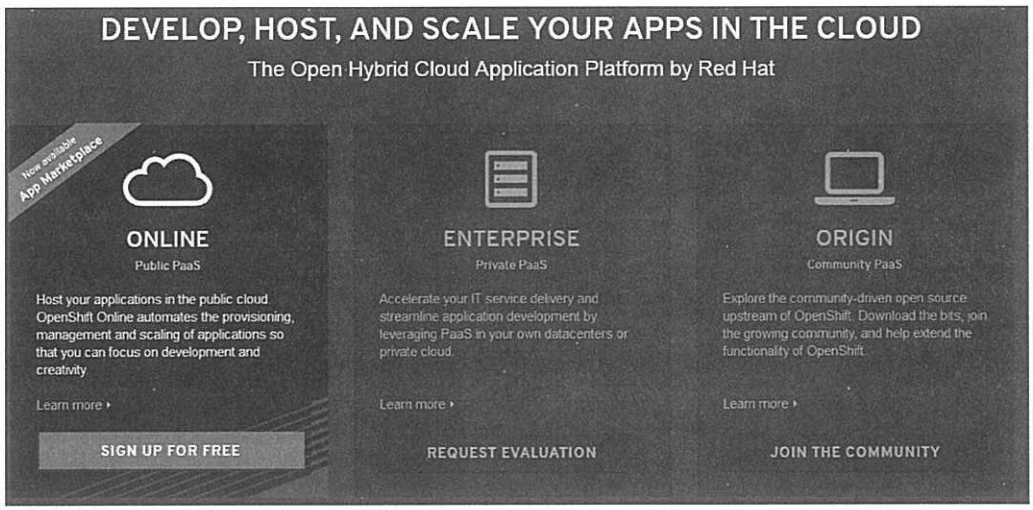


Figura 6.69: Interfaces de usuario

- ***OpenShift Web Interface***. Las aplicaciones pueden ser creadas y gestionadas usando *OpenShift Management Console*, una interfaz gráfica de usuario accesible usando un navegador. Tambien permite gestionar las opciones de tu cuenta de usuario *OpenShift* y enlaces a documentación y recursos.
- ***OpenShift Command Line Interface***. Utilizando un conjunto de herramientas de en línea de comandos (`https://www.openshift.com/developers/rhc-client-tools-install`) (escritas en el lenguaje de programación *Ruby*) que se comunican con los servidores de *OpenShift*.
- ***IDE***. Utilizando *JBoss Developer Studio* (también existen un conjunto de herramientas (*JBoss Tools*) para poder utilizar *Eclipse*) o *Zend Studio*.

  En esta práctica usaremos la interfaz web y las herramientas en línea de comandos.

4. **Instalación de Git**

   Recuerda que por cada aplicación *Openshift* se crea un repositorio Git al que solo puede acceder el propietario de la aplicación. *Openshift* accede al repositorio de Git para obtener el código de la aplicación y desplegarlo en el servidor correspondiente. Los desarrolladores tienen que subir el código de sus aplicaciones al repositorio. Por lo tanto es necesario tener instalado Git en la máquina desde la que queremos desarrollar/desplegar aplicaciones.

   4.1. Inicia sesión en **DesarrolloW7XX** con un usuario con privilegios de administrador.

   4.2. Accede la web oficial de Git (`http://git-scm.com/`) descarga la última versión disponible para *Windows* (el momento de desarrollar este documento la versión **1.9.5**).

   4.3. Instala Git con las opciones por defecto excepto para la opción ***Adjusting your Path enviroment*** en la que seleccionaremos ***Run Git from the Windows Command Prompt***, Figura 6.70.

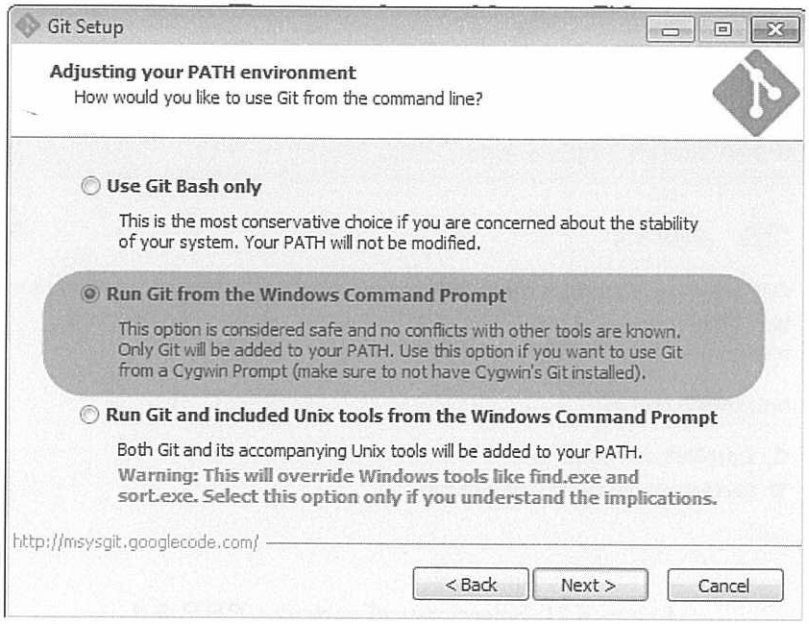


Figura 6.70: Instalación de Git

5. Accede a **Inicio, Todos los programas, Git, Git Bash** para abrir una consola de Git.

6. Ejecuta los siguientes comando para defenir un nombre de usuario y una dirección de correo con los que almacenarán tus cambios en Git.

```
git config --global user.name "Tu nombre"
git config --global user.email tu@example.com
```

7. **Registro en *OpenShift***

   7.1. Inicia sesión en **DesarrolloW7XX** con un usuario con privilegios de administrador.

   7.2. Accede a `https://www.openshift.com/`.

7.3. Pincha en ***Sign up for free***.

7.4. Regístrarte.

7.5. Acederas a tu cuenta y serás redirigido a una página de documentación.

8. **Despliegue de una aplicación usando el interfaz web**

En primer lugar vamos a crear una aplicación en PHP utilizando la interfaz web que ofrece *OpenShift*. Vamos a seguir los pasos indicados en **Get started in the cloud** (`https://www.openshift.com/get-started`).

8.1. Una vez iniciada sesión en *OpenShift* con tu usuario pincha en ***Create your first application now***.

8.2. Selecciona el *cartridge* **PHP 5.4** que necesitamos para crear la aplicación, Figura 6.71.

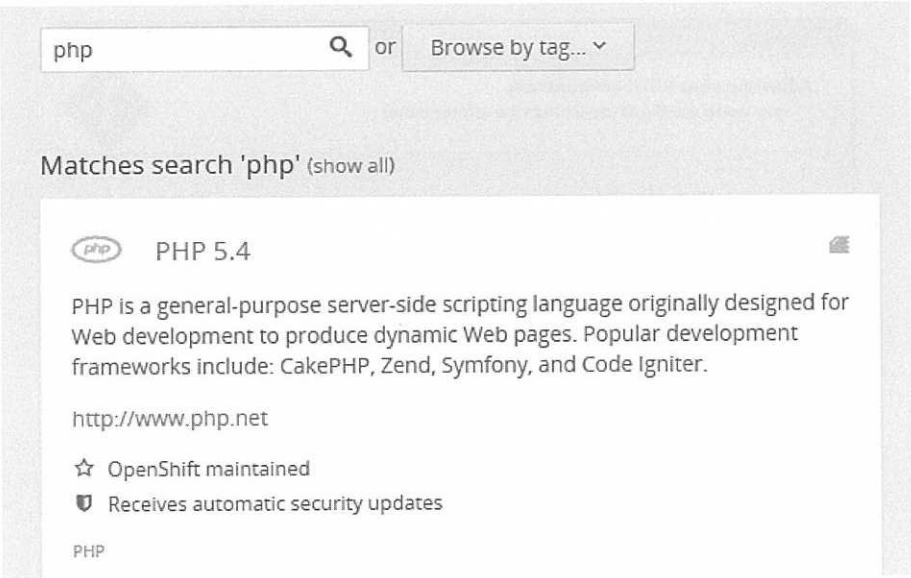


Figura 6.71: Seleccionar el *cartridge* **PHP 5.3**

8.3. Introduce el nombre de tu aplicación (por ejemplo **cursodespliegue**) el ***namespace*** de tu cuenta (por ejemplo **dawXX**), Figura 6.72.

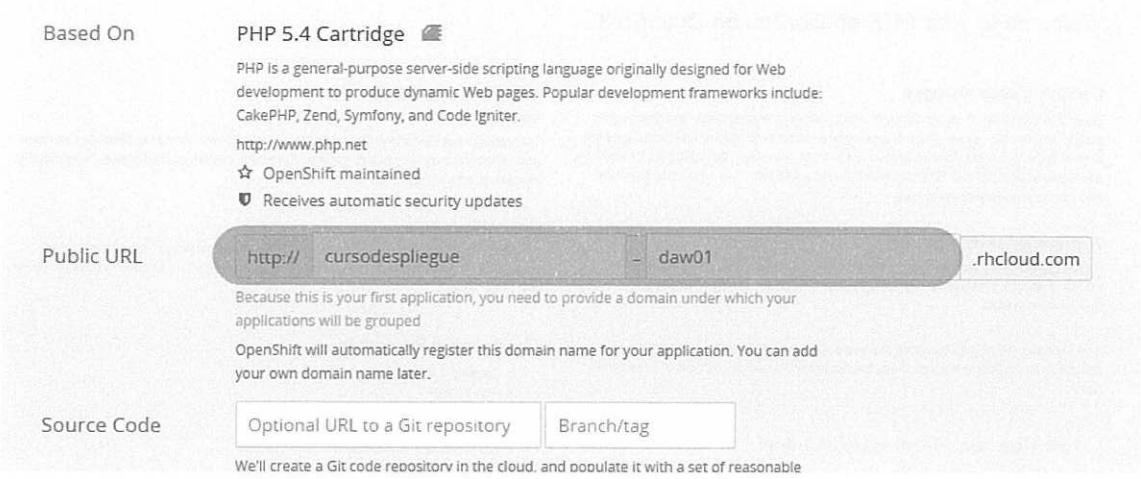


Figura 6.72: Nombre de la aplicación

8.4. Observa el resto de opciones que se pueden configurar y déjalas por defecto.

8.5. Pincha en ***Create Application*** y espera a que se cree la aplicación.

8.6. Observa las opciones que nos ofrecen una vez creada la aplicación, Figura 6.73.

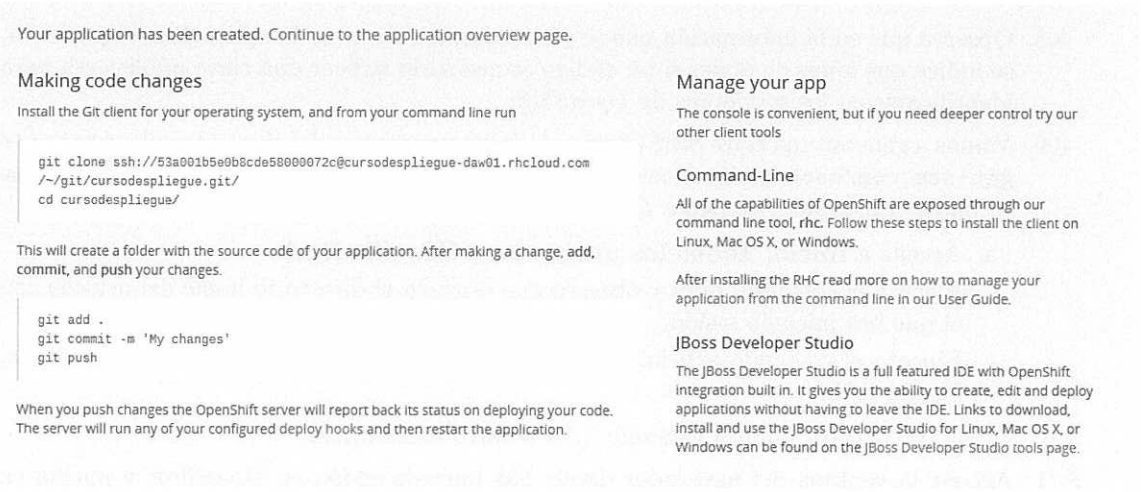


Figura 6.73: Aplicación creada

8.7. Pincha con el botón derecho del ratón sobre **visit app in the browser** y abre la aplicación en otra ventana del navegador. Observa que la aplicación consiste en una página html con información sobre cómo desplegar el código de nuestra aplicación, Figura 6.74.

Welcome to your PHP application on OpenShift

Deploying code changes

OpenShift uses the Git version control system for your source code, and grants you access to it via the Secure Shell (SSH) protocol. In order to upload and download code to your application you need to give us your public SSH key. You can upload it within the web console or install the RHC command line tool and run `rhc setup` to generate and upload your key automatically.

Working in your local Git repository

If you created your application from the command line and uploaded your SSH key, rhc will automatically download a copy of that source code repository (Git calls this 'cloning') to your local system.

If you created the application from the web console, you'll need to manually clone the repository to your local system. Copy the application's source code Git URL and then run:

```
$ git clone <git_url> <directory_to_create>

# Within your project directory
# Commit your changes and push to OpenShift

$ git commit -a -m 'Some commit message'
$ git push
```

- Learn more about deploying and building your application
- Read more details about Repository Layout, Action Hooks and Markers

Managing your application

Web Console

You can use the OpenShift web console to enable additional capabilities via cartridges, add collaborator access authorizations, designate custom domain aliases, and manage domain memberships.

Command Line Tools

Installing the OpenShift RHC client tools allows you complete control of your cloud environment. Read more on how to manage your application from the command line in our User Guide.

Development Resources

- Developer Center
- User Guide
- OpenShift Support
- Stack Overflow questions for OpenShift
- IRC channel at #openshift on freenode.net
- Git documentation

Figura 6.74: Acceso a la aplicación

8.8. Observa que en la información que se ofrece después de crear la aplicación, Figura 6.74, se indica que antes de poder subir código es necesario indicar una clave pública ssh para identificarse en los servidores de *OpenShift*.

8.9. Vamos a generar una clave SSH (siguiendo las instrucciones del siguiente enlace `http://git-scm.com/book/es/Git-en-un-servidor-Generando-tu-clave-pblica-SSH`) para poder usarla subir código a *OpenShift*.

   a. Accede a **Inicio, Todos los programas, Git, Git Bash**.

   b. Ejecuta el comando **pwd** y observa que estás en el directorio home del usuario con el que has iniciado sesión.

   c. Ejecuta el comando **ssh-keygen** para generar un par de clave pública y privada, Figura 6.75.

8.10. Copia del la clave pública generada (del archivo **id_rsa.pub**.

8.11. Accede la ventana del navegador donde has iniciado sesión en OpenShift y pincha en **Settings**.

8.12. Copia del la clave pública generada (del archivo **id_rsa.pub** y cópiala en el cuadro de texto de *Public Keys*, Figura 6.76 (cuidado, elimina los posibles espacios en blanco al final), y pincha en ***Save***.

```
MINGW32:/c/Users/alumno
Welcome to Git (version 1.9.4-preview20140611)


Run 'git help git' to display the help index.
Run 'git help <command>' to display help for specific commands.

alumno@DESARROLLOW701 ~
$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/c/Users/alumno/.ssh/id_rsa):
Created directory '/c/Users/alumno/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /c/Users/alumno/.ssh/id_rsa.
Your public key has been saved in /c/Users/alumno/.ssh/id_rsa.pub.
The key fingerprint is:
32:61:85:8c:66:ff:de:bf:96:10:da:04:06:81:67:e0 alumno@DESARROLLOW701

alumno@DESARROLLOW701 ~
$ _
```

Figura 6.75: Generar claves SSH

Public Keys

OpenShift uses a public key to securely encrypt the connection between your local machine and your application and to authorize you to upload code. You must create a private and public key on your local machine and then upload the public key before you can connect to your applications' Git repositories or remotely access your applications.

Learn more about SSH keys.

Paste the contents of your public key file (.pub)

```
+GbPTGSAVZTsXg0poSQNeZ
/aoKFq+D3kMUSi5JelKJixwJwmWwCr2
/wVCQ9ViiL5uwQRSmnBarWC05OskPEKdeTLkc0p
szL50
/ushRapXSyc1BIH89VLz7FPQFxvwSWwcD9w==
alumno@DESARROLLOW701
```

Save

Figura 6.76: Añadir clave pública SSH

8.13. La clave se añadirá, Figura 6.77.

Settings

Your public key has been created

Public Keys

OpenShift uses a public key to securely encrypt the connection between your local machine and your application and to authorize you to upload code. Learn more about SSH keys.

Key name	Type	Contents	
default	ssh-rsa	AAAB3Nza..WwcD9w==	Delete

Add a new key...

Figura 6.77: Información sobre Git

8.14. Accede a **Applications** y tu aplicación. Observa la URL GIT con el código de tu aplicación, Figura 6.78.

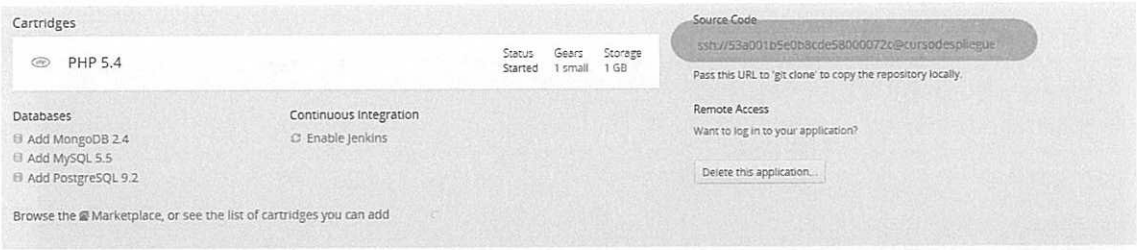


Figura 6.78: URL Git

8.15. Accede a **Inicio, Todos los programas, Git, Git Bash** para abrir una consola de Git.

8.16. Clona el repositorio de tu aplicación usando el comando **git clone**, Figura 6.79.

```
alumno@DESARROLLOW701 ~
$ git clone ssh://53a001b5e0b8cde58000072c@cursodespliegue-daw01.rhcloud.com/~/
git/cursodespliegue.git/
Cloning into 'cursodespliegue'...
Receiving objects:  94% (18/19)   . eceiving objects:   5% (1/19)
remote: Compressing objects: 100% (14/14), done.
remote: Total 19 (delta 1), reused 19 (delta 1)
Receiving objects: 100% (19/19), 18.21 KiB | 0 bytes/s, done.
Resolving deltas: 100% (1/1), done.
Checking connectivity... done.

alumno@DESARROLLOW701 ~
$
```

Figura 6.79: Clonar el repositorio Git

8.17. Accede a la carpeta de la aplicación (**cd cursodespliegue**).

8.18. Edita el fichero **index.php** con el contenido que quieras.

8.19. Sobre el directorio raíz de la aplicación (cursodespliegue) ejecuta los siguientes comandos para subir los cambios al repositorio git de *OpenShift*.

```
git add -A
git commit -m "Aplicación solo con index.php"
git push
```

8.20. Accede a la url de la aplicación y verifica los cambios que has realizado, Figura 6.80.

Figura 6.80: Acceso a la aplicación

9. <u>**Acceso SSH**</u>

   OpenShift permite acceder por ssh a la máquina virtual donde se está ejecutando la aplicación.

   9.1. Utiliza el navegador para acceder a tus aplicaciones de *OpenShift* y selecciona la aplicación **cursodespliegue**.

   9.2. Pincha en **Remote Access** y observa la cadena de conexión que hay que utilizar para establecer una conexión ssh, Figura 6.81.

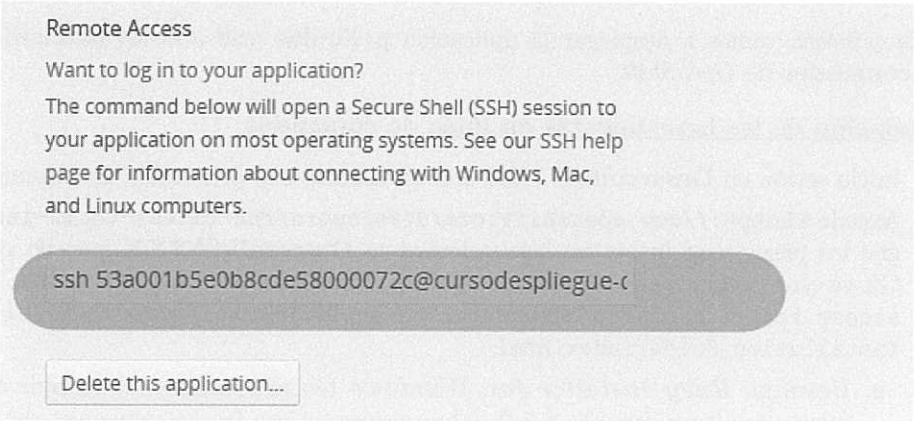


Figura 6.81: Cadena de conexión ssh

9.3. Utiliza el cliente ssh integrado en Git para establecer una conexión SSH, Figura 6.82.

```
alumno@DESARROLLOW701 ~/cursodespliegue (master)
$ ssh 53a001b5e0b8cde58000072c@cursodespliegue-daw01.rhcloud.com

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

    You are accessing a service that is for use only by authorized users.
    If you do not have authorization, discontinue use at once.
    Any use of the services is subject to the applicable terms of the
    agreement which can be found at:
    https://www.openshift.com/legal

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

    Welcome to OpenShift shell

    This shell will assist you in managing OpenShift applications.

    !!! IMPORTANT !!! IMPORTANT !!! IMPORTANT !!!
    Shell access is quite powerful and it is possible for you to
    accidentally damage your application.  Proceed with care!
    If worse comes to worst, destroy your application with "rhc app delete"
    and recreate it
    !!! IMPORTANT !!! IMPORTANT !!! IMPORTANT !!!

    Type "help" for more info.

[cursodespliegue-daw01.rhcloud.com 53a001b5e0b8cde58000072c]\>
```

Figura 6.82: Conexión SSH

9.4. Navega por la estructura de directorios e investiga donde está el código de tu aplicación.

6.9. Despliegue en la nube (OpenShift) - 2

En esta práctica vamos a desplegar la aplicación **peliculas** utilizado las **herramientas en línea de comandos** de *OpenShift*.

1. <u>**Instalación de las herramientas en línea de comandos**</u>

   1.1. Inicia sesión en **DesarrolloW7XX** con un usuario con privilegios de administrador.

   1.2. Accede a `https://www.openshift.com/developers/rhc-client-tools-install` y sigue los pasos para instar las herramientas en **DesarrolloW7XX** (ten en cuenta que Git ya está instalado). En la siguiente guía se ofrece información mas detallada `https://access.redhat.com/site/documentation/en-US/OpenShift/2.0/html/Client_Tools_Installation_Guide/index.html`.

      a. Descarga ***Ruby Installer for Windows*** (en el momento de escribir este documento la última versión **2.2.2** daba problemas con las herramientas rhc, descarga la versión **2.0.0**).

      b. Instala el software marcando la opción ***Add Ruby Executables to Path***.

      c. Abre un terminal y ejecuta el siguiente comando para utilizar ***RubyGems package manager*** y descargar e instalar las ***OpenShift client tools***.

      ```
      gem install rhc
      ```

2. **Configuración inicial**

   2.1. Ejecuta el siguiente comando para realizar una configuración inicial y sigue los pasos indicados.

   ```
   rhc setup
   ```

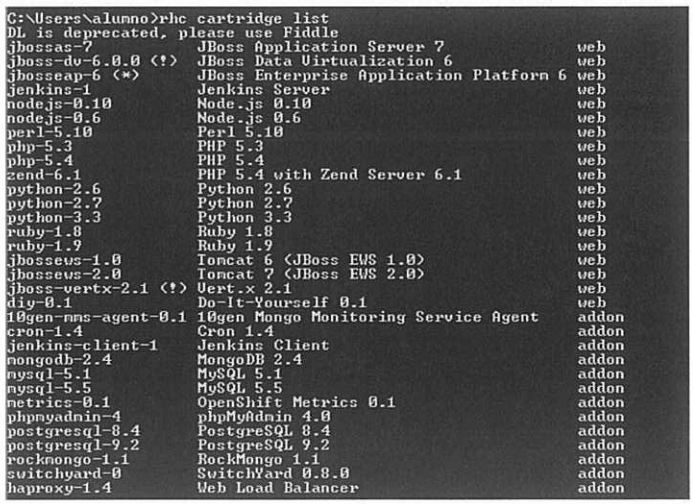
   El asistente realiza las siguientes acciones de configuración:

   - Pregunta por las credenciales de acceso a tu cuenta en *OpenShift* (usuario y contraseña).
   - Genera un *token* de autenticación. Si se acepta su almacenamiento se usará en la siguientes conexiones (hasta que expire no será necesario introducir de nuevo la contraseña).
   - Genera el archivo **express.conf** que se almacena el directorio home de tu usuario con información sobre la conexión.
   - Crea un par de claves SSH para la autenticación en el servidor (en nuestro caso como ya se crearon en la práctica anterior, usa las que ya existen) y sube la clave pública al servidor (nosotros lo hicimos manualmente a través de la consola web).
   - Verifica si Git está instalado (recuerda que Git es necesario para crear y desplegar aplicaciones).
   - Verifica si ya has creado un *Namespace* (en nuestro caso ya lo hemos creado en la práctica anterior) y si no lo has creado te pregunta su nombre para crearlo.
   - Finalmente, verifica las aplicaciones existentes en tu cuenta.

3. **Crear la aplicación**

   3.1. Ejecuta el siguiente comando para mostrar los ***cartridges*** disponibles, Figura 6.83.

   ```
   rhc cartridge list
   ```



```
C:\Users\alumno>rhc cartridge list
DL is deprecated, please use Fiddle
jbossas-7              JBoss Application Server 7                web
jboss-dv-6.0.0 (!)     JBoss Data Virtualization 6               web
jbosseap-6 (*)         JBoss Enterprise Application Platform 6   web
jenkins-1              Jenkins Server                            web
nodejs-0.10            Node.js 0.10                              web
nodejs-0.6             Node.js 0.6                               web
perl-5.10              Perl 5.10                                 web
php-5.3                PHP 5.3                                   web
php-5.4                PHP 5.4                                   web
zend-6.1               PHP 5.4 with Zend Server 6.1              web
python-2.6             Python 2.6                                web
python-2.7             Python 2.7                                web
python-3.3             Python 3.3                                web
ruby-1.8               Ruby 1.8                                  web
ruby-1.9               Ruby 1.9                                  web
jbossews-1.0           Tomcat 6 (JBoss EWS 1.0)                  web
jbossews-2.0           Tomcat 7 (JBoss EWS 2.0)                  web
jboss-vertx-2.1 (!)    Vert.x 2.1                                web
diy-0.1                Do-It-Yourself 0.1                        web
10gen-mms-agent-0.1    10gen Mongo Monitoring Service Agent      addon
cron-1.4               Cron 1.4                                  addon
jenkins-client-1       Jenkins Client                            addon
mongodb-2.4            MongoDB 2.4                               addon
mysql-5.1              MySQL 5.1                                 addon
mysql-5.5              MySQL 5.5                                 addon
metrics-0.1            OpenShift Metrics 0.1                     addon
phpmyadmin-4           phpMyAdmin 4.0                            addon
postgresql-8.4         PostgreSQL 8.4                            addon
postgresql-9.2         PostgreSQL 9.2                            addon
rockmongo-1.1          RockMongo 1.1                             addon
switchyard-0           SwitchYard 0.8.0                          addon
haproxy-1.4            Web Load Balancer                         addon
```

Figura 6.83: *Cartridges* disponibles

3.2. Ejecuta el siguiente comando para crear la aplicación **peliculas** con el *cartridge* PHP. Figura 6.84.

```
rhc app create peliculas php-5.3
```

Observa la información que ofrece la salida del comando. Puedes observar que se ha creado la aplicación y se ha clonado el repositorio git (recuerda que la práctica anterior lo hicimos nosotros con el comando **git clone**).

```
C:\Users\alumno>rhc app create peliculas php-5.3
DL is deprecated, please use Fiddle
Application Options
-------------------
Domain:     daw01
Cartridges: php-5.3
Gear Size:  default
Scaling:    no

Creating application 'peliculas' ... done


Waiting for your DNS name to be available ... done

Cloning into 'peliculas'...
Warning: Permanently added the RSA host key for IP address '54.196.46.200' to th
e list of known hosts.
remote: Counting objects: 19, done.
remote: Compressing objects: 100% (14/14), done. eceiving objects:   5% (1/19)
Receiving objects:  94% (18/19)
remote: Total 19 (delta 1), reused 19 (delta 1)
Receiving objects: 100% (19/19), 18.21 KiB | 0 bytes/s, done.
Resolving deltas: 100% (1/1), done.
Checking connectivity... done.

Your application 'peliculas' is now available.

  URL:        http://peliculas-daw01.rhcloud.com/
  SSH to:     53a00c885973ca655a00001c@peliculas-daw01.rhcloud.com
  Git remote: ssh://53a00c885973ca655a00001c@peliculas-daw01.rhcloud.com/~/git/p
eliculas.git/
  Cloned to:  C:/Users/alumno/peliculas

Run 'rhc show-app peliculas' for more details about your app.

C:\Users\alumno>
```

Figura 6.84: Creación de la aplicación **peliculas** con el *cartrige* PHP

3.3. Como la aplicación **peliculas** necesita una base de datos *MySQL* vamos a añadir el *cartridge MySQL* a la aplicación, observa la información que ofrece la salida del comando, Figura 6.85.

```
rhc cartridge add mysql-5.5 -a peliculas
```

3.4. Ahora añadiremos el el *cartridge* de *PHPMyadmin* para administrar la base de datos fácilmente. Observa la información que ofrece la salida del comando, Figura 6.86.

```
rhc cartridge add phpmyadmin-4 -a peliculas
```

3.5. Si accedes a tu cuenta con la consola de administración web, puedes observar la información de tu nueva aplicación, Figuras 6.87 y 6.88.

3.6. Creación de la base de datos

a. Inicia una navegador y accede a **phpmyadmin**

```
C:\Users\alumno>rhc cartridge add mysql-5.5 -a peliculas
DL is deprecated, please use Fiddle
Adding mysql-5.5 to application 'peliculas' ... done

mysql-5.5 (MySQL 5.5)
---------------------
  Gears:          Located with php-5.3
  Connection URL: mysql://$OPENSHIFT_MYSQL_DB_HOST:$OPENSHIFT_MYSQL_DB_PORT/
  Database Name:  peliculas
  Password:       VLi81kajldwy
  Username:       adminTDAMS1d

MySQL 5.5 database added.  Please make note of these credentials:

       Root User: adminTDAMS1d
   Root Password: VLi81kajldwy
   Database Name: peliculas

Connection URL: mysql://$OPENSHIFT_MYSQL_DB_HOST:$OPENSHIFT_MYSQL_DB_PORT/

You can manage your new MySQL database by also embedding phpmyadmin.
The phpmyadmin username and password will be the same as the MySQL credentials
above.

C:\Users\alumno>
```

Figura 6.85: Añadir el cartridge *mysql*

```
C:\Users\alumno>rhc cartridge add phpmyadmin-4 -a peliculas
DL is deprecated, please use Fiddle
Adding phpmyadmin-4 to application 'peliculas' ... done

phpmyadmin-4 (phpMyAdmin 4.0)
-----------------------------
  Gears:          Located with php-5.3, mysql-5.5
  Connection URL: https://peliculas-daw01.rhcloud.com/phpmyadmin/

Please make note of these MySQL credentials again:
  Root User: adminTDAMS1d
  Root Password: VLi81kajldwy
URL: https://peliculas-daw01.rhcloud.com/phpmyadmin/

C:\Users\alumno>_
```

Figura 6.86: Añadir el cartridge *phpmyadmin*

Figura 6.87: Aplicaciones

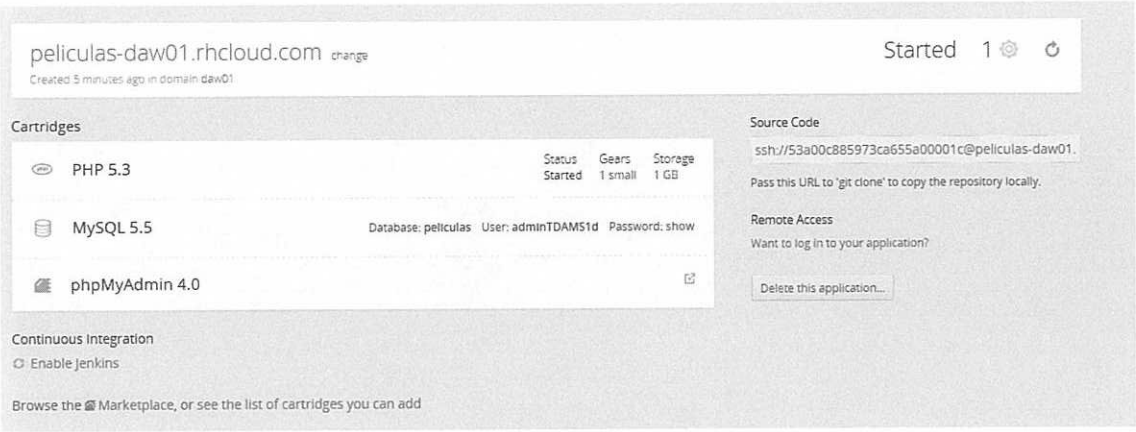


Figura 6.88: Aplicación **peliculas**

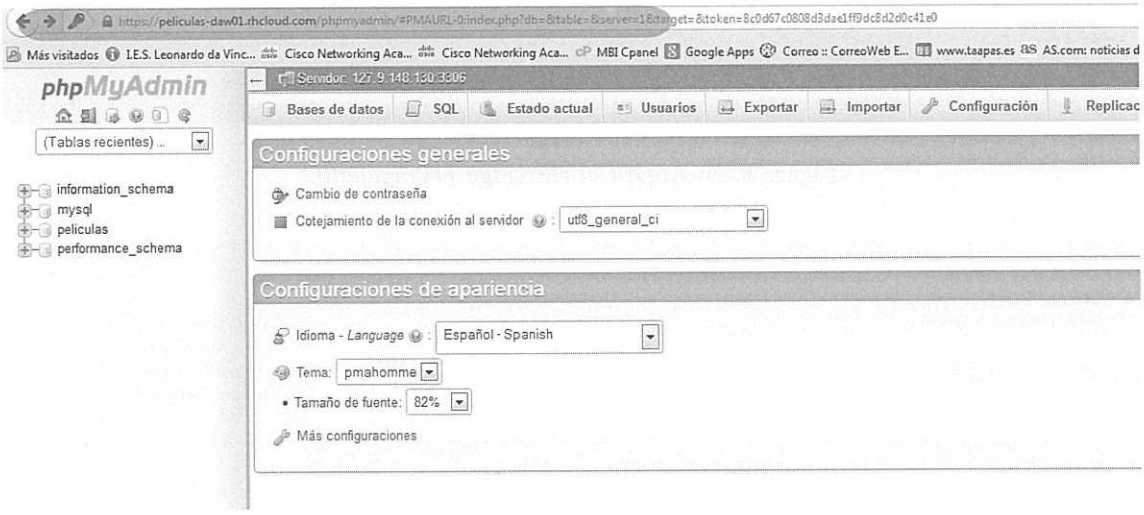


Figura 6.89: Acceso a phpmyadmin

b. Inicia sesión con el usuario y contraseña que has obtenido en la salida del comando anterior, Figura 6.89.
c. En la parte izquierda selecciona la base de datos **peliculas**.
d. Pincha en **Importar**.
e. Pincha en **Examinar** y selecciona el script **crear_tablas_peliculas.sql**. Pincha en **Continuar** para ejecutar el script.
f. Pincha de nuevo en **Examinar** y selecciona el script **insercion_admin_peliculas.sql**. Pincha en **Continuar** para ejecutar el script.

4. **Desplegar la aplicación**

4.1. Accede al directorio de la aplicación **peliculas** en el directorio home de tu usuario.

4.2. Accede al directorio **php** y borra su contenido.

4.3. Copia el código de la aplicación **peliculas** en directorio **php**.

4.4. Configuración de la conexión a la base de datos.

a. Hay que configurar la aplicación para que acceda la base de datos de mysql creada en *OpenShift* (recuerda la cadena de conexión, usuario y contraseña que se crearon al añadir el *cartrige* MySQL).
b. Edita el fichero **configuracion.php** y configura las variables que definen la conexión a la base de datos, Figura 6.90. En el siguiente enlace `https://github.com/openshift/origin-server/tree/master/cartridges/openshift-origin-cartridge-php/README.md` puedes encontrar información sobre las variables de entorno que almacenan los parámetros de conexión a la base de datos **MySQL** en **Openshift**. También seria posible poner directamente los valores.

```
<?php

// Path de las clases y de las paginas estandar
define("CLASS_PATH", dirname($_SERVER['PHP_SELF']).."/");
define("APPLICATION_PATH", dirname($_SERVER['PHP_SELF']).."/");

//Configuración de la conexion a la base de datos
define("BD_SERVIDOR", getenv('OPENSHIFT_MYSQL_DB_HOST').":".getenv('OPENSHIFT_MYSQL_DB_PORT'));
define("BD_NOMBRE", "peliculas");
define ("BD_USUARIO", getenv('OPENSHIFT_MYSQL_DB_USERNAME'));
define ("BD_PASSWORD", getenv('OPENSHIFT_MYSQL_DB_PASSWORD'));

//Configuración de logitudes minimas del login y el password de los usuario
//normales
define ("LONGITUD_MINIMA_LOGIN", 6);
define ("LONGITUD_MINIMA_PASSWORD", 6);

//Configuración del numero de resultados de paginación por cada pagina
define ("PAGINACION", 10);

?>
```

Figura 6.90: Fichero **configuracion.php**

4.5. Configuración de la conexión a la base de datos.

4.6. Accede a **Inicio, Todos los programas, Git, Git Bash** para abrir una consola de Git.

4.7. Accede a la carpeta de la aplicación (**cd peliculas**).

4.8. Sobre el directorio raíz de la aplicación (peliculas) ejecuta los siguientes comandos para subir los cambios al repositorio git de *OpenShift*, Figura 6.91.

```
git add -A
git commit -m "Primer despliegue de peliculas"
git push
```

```
alumno@DESARROLLOW701 ~/peliculas (master)
$ git push
warning: push.default is unset; its implicit value is changing in
Git 2.0 from 'matching' to 'simple'. To squelch this message
and maintain the current behavior after the default changes, use:

  git config --global push.default matching

To squelch this message and adopt the new behavior now, use:

  git config --global push.default simple

See 'git help config' and search for 'push.default' for further information.
(the 'simple' mode was introduced in Git 1.7.11. Use the similar mode
'current' instead of 'simple' if you sometimes use older versions of Git)

Enter passphrase for key '/c/Users/alumno/.ssh/id_rsa':
Counting objects: 24, done.
Compressing objects: 100% (20/20), done.
Writing objects: 100% (21/21), 23.00 KiB | 0 bytes/s, done.
Total 21 (delta 6), reused 0 (delta 0)
remote: CLIENT_MESSAGE: Stopping Apache+mod_php HTTPD server
remote: Waiting for stop to finish
remote: [Thu Jun 06 06:49:42 2013] [warn] module php5_module is already loaded,
skipping
remote: Waiting for stop to finish
remote: CLIENT_MESSAGE: `pre-repo-archive' control action not implemented
remote: CLIENT_MESSAGE: `update-configuration' control action not implemented
remote: CLIENT_MESSAGE: `pre-build' control action not implemented
remote: [Thu Jun 06 06:49:59 2013] [warn] module php5_module is already loaded,
skipping
remote: CLIENT_MESSAGE: Starting Apache+mod_php HTTPD server
remote: CLIENT_MESSAGE: `post-deploy' control action not implemented
To ssh://51af1ddc5973ca11a6000132@peliculas-daw01.rhcloud.com/~/git/peliculas.gi
t/
   98bf24b..e8f95e9  master -> master

alumno@DESARROLLOW701 ~/peliculas (master)
$
```

Figura 6.91: Despliegue de la aplicación **peliculas**

4.9. Accede a la url de la aplicación **peliculas** para probar su funcionamiento, Figura 6.92.

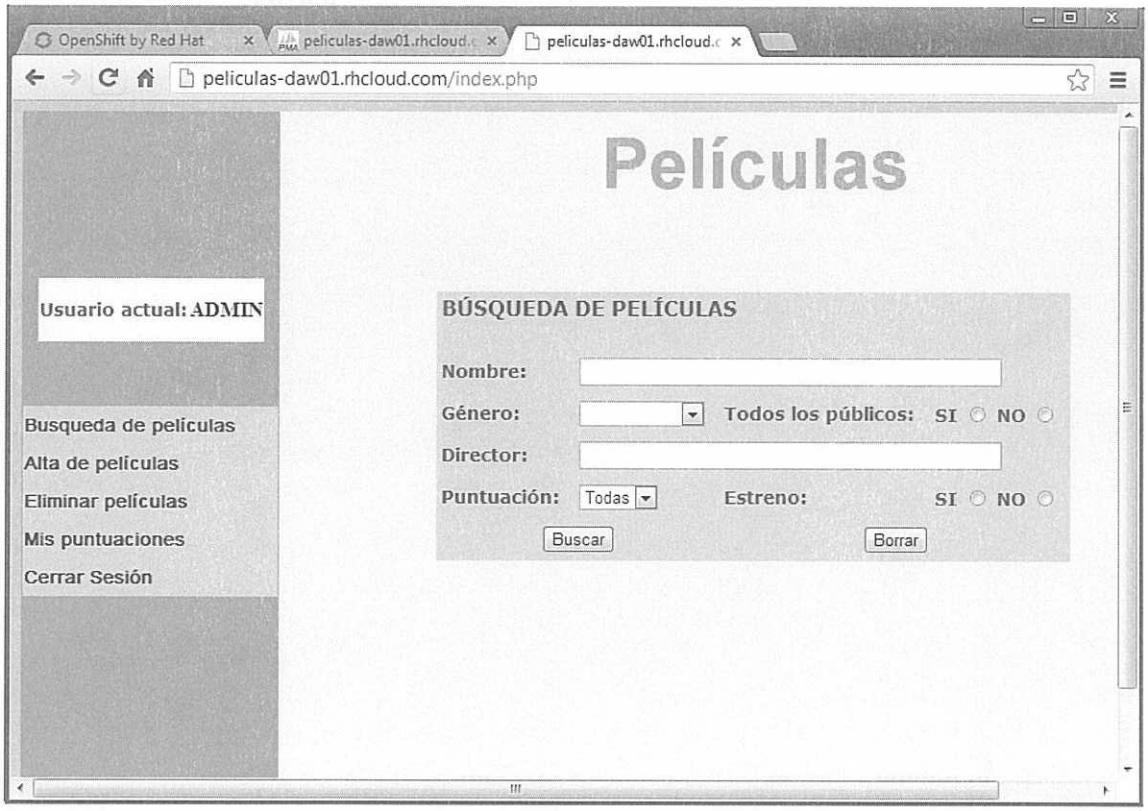


Figura 6.92: Acceso a la aplicación **peliculas**

Capítulo 7

Aplicaciones JavaEE

7.1. Instalación de *Tomcat 7.0* en *Windows*

En esta práctica se instalará el servidor de aplicaciones *Apache Tomcat*, en su versión 7.0, en la máquina **DesarrolloW7**.

Apache Tomcat 7.0 requiere para su funcionanmiento tener instalada la máquina virtual de *Java* (*Java Standard Edition Runtime Environment (JRE)* en su versión 6.0 o posterior.

1. **Instalación de *Java Standard Edition Development Kit (JDK)*.**

   1.1. Inicia sesión en el equipo **DesarrolloW7XX** con el usuario **alumno**.

   1.2. Descarga *Java SE Development Kit* de `http://www.oracle.com/technetwork/java/javase/downloads/index.html`. En el momento de realizar este manual estaba disponible la versión 7.
   Elige la versión de 32 bits de instalación offline, *Windows x86 Offline*, y descarga el fichero **jdk-7u21-windows-i586.exe**.

   1.3. Instala la versión de *JDK* descargada previamente.

   A la hora de instalar *Tomcat* tenemos dos opciones, utilizar un instalador de *Windows* que realiza de forma automática la instalación creando un servicio asociado a *Tomcat* o realizar manualmente la instalación. Vamos a ver ambas aunque la segunda nos dará más posibilidades de personalización.

2. **Instalación de *Tomcat* utilizando el instalador de *Windows*.**

   2.1. Descarga de `http://tomcat.apache.org/` la versión **32-bit/64-bit Windows Service Installer**.

   2.2. Instala la versión descargada anteriormente, marcando la instalación completa y dejando el resto de opciones por defecto, Figuras 7.1, 7.2:

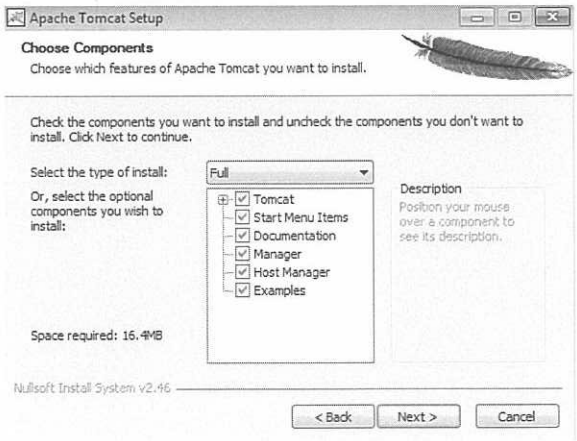


Figura 7.1: Instalación automática

   2.3. Esta instalación automatizada ha creado un servicio asociado a *Tomcat* que se iniciará al arrancar *Windows*.

   2.4. Comprueba que el servidor está iniciado, en la barra de tareas abajo a la izquierda hay un icono que muestra una herramienta para monitorizar el servidor, Figura 7.3.

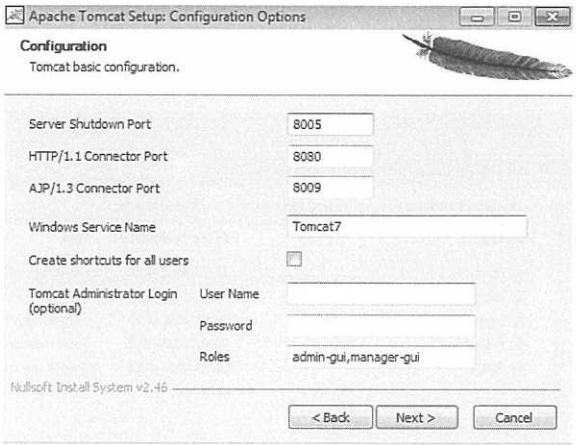


Figura 7.2: Instalación automática

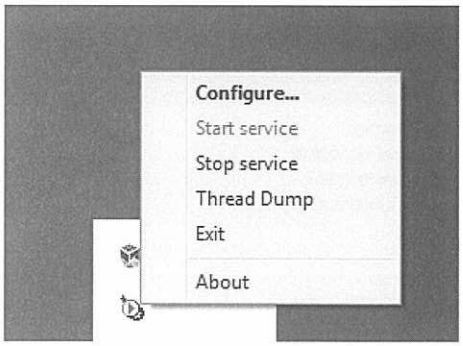


Figura 7.3: Consola de administración

2.5. Comprueba que el servidor está escuchando en el puerto 8080/TCP.

```
netstat -a -p TCP -n
```

2.6. Consulta el contenido del directorio **C:\Program Files\Apache Software Foundation\Tomcat 7.0\webapps**, Figura 7.4.

2.7. Consulta el contenido del directorio **C:\Program Files\Apache Software Foundation\Tomcat 7.0\webapps\ROOT**.

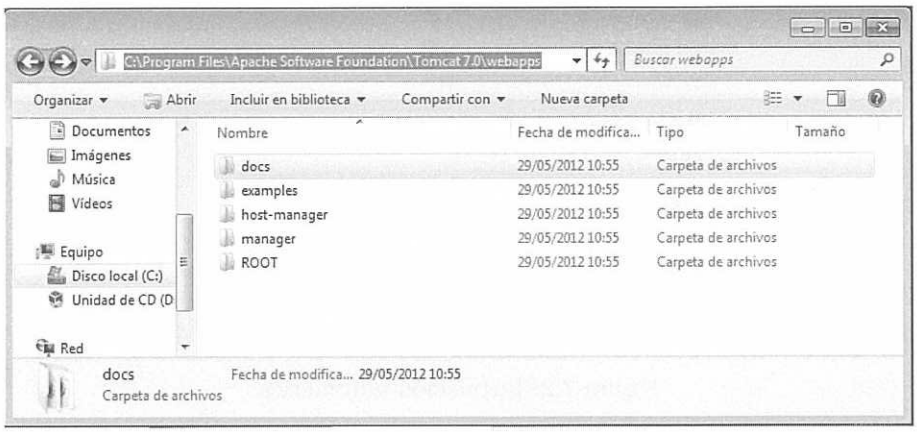


Figura 7.4: Directorio **C:\Program Files\Apache Software Foundation\Tomcat 7.0\webapps**

2.8. Para comprobar el correcto funcionamiento de *Tomcat* accede con el navegador a la dirección `http://localhost:8080`, Figura 7.5.

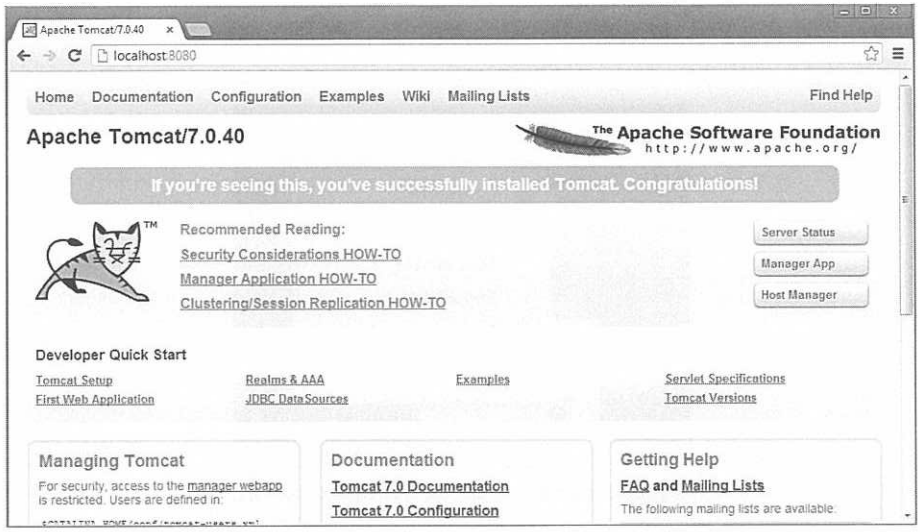


Figura 7.5: Comprobación del funcionamiento

3. **Ejecución de *Servlets* en *Tomcat*.**

   3.1. Para terminar de verificar el correcto funcionamiento de la instalación vamos a ejecutar algún *servlet* de los que vienen incluidos con la instalación de *Tomcat*.

   3.2. Arranca el servidor y accede al mismo. En la página de inicio selecciona **Examples**, Figura 7.6.

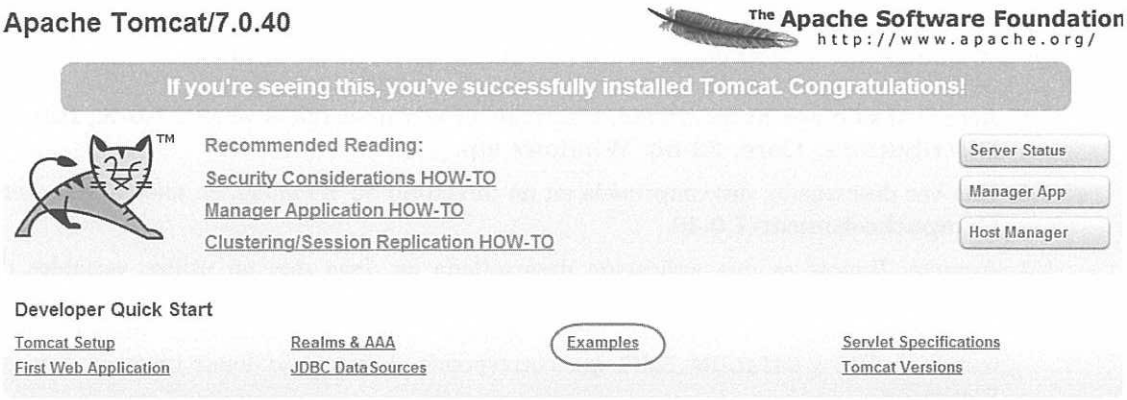


Figura 7.6: Ejemplos *servlets*

   3.3. Ejecuta el *servlet HelloWorld*, Figura 7.7.

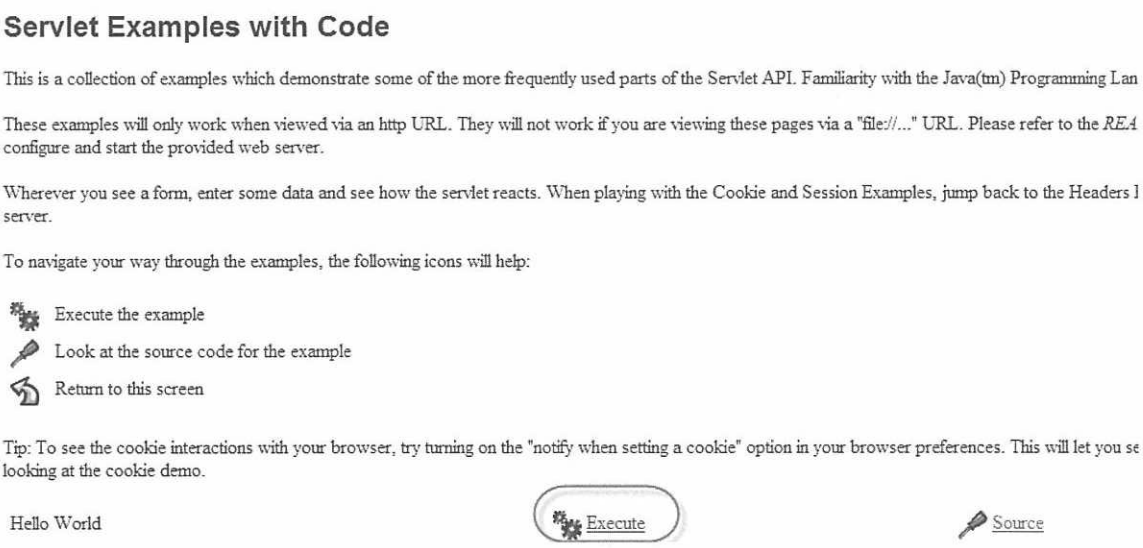


Figura 7.7: Ejemplos *servlets*

4. **Desinstalar *Tomcat***

   4.1. Para el servicio asociado a *Tomcat*.

   4.2. Accede a **Inicio, Todos los programas, *Apache Tomcat 7.0 Tomcat 7, Unistall Tomcat 7.0*** y sigue los pasos del asistente.

5. **Instalación manual de *Tomcat*.**

   Para gestionar todos los aspectos de la instalación de Tomcat, gestionar las variables de entorno y administrar *Tomcat* desde línea de comandos vamos a instalar *Tomcat* manualmente.

   5.1. Accede al sitio web `http://tomcat.apache.org/` y descarga la versión **7.0.X, Binary Distributions, Core, 32-bit Windows zip**.

   5.2. Una vez descargada, descomprímela en un directorio de *Windows*, en nuestro caso será **C:\apache-tomcat-7.0.40**

   5.3. Aunque *Tomcat* es una aplicación desarrollada en *Java* que no utiliza variables de entorno, para ejecutar los *scripts* de administración necesitamos crear dos variables de entorno de sistema en *Windows*, `JAVA_HOME` que apuntará al directorio donde tenemos instalado *JRE* y `CATALINA_HOME` que corresponde al directorio donde tenemos *Tomcat*, Figura 7.8.

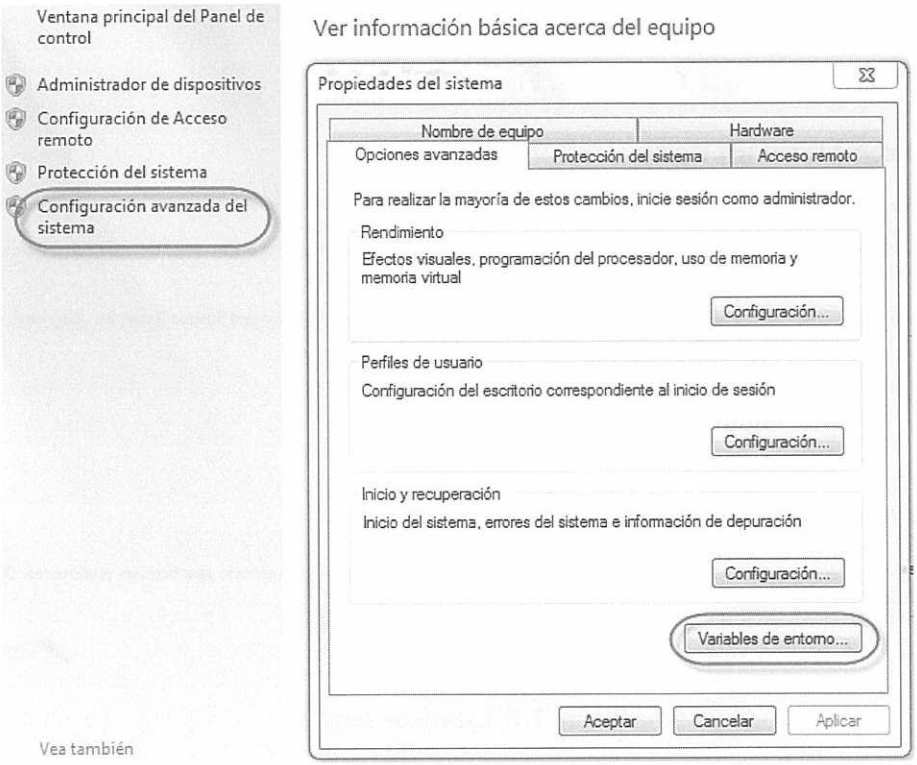


Figura 7.8: Creación de variables de entorno

   5.4. Para comprobar que las variables de entorno han sido creadas correctamente ejecuta los comandos:

```
echo %CATALINA_HOME%
echo  %JAVA_HOME%
```

5.5. Si no aparecieran reinicia la máquina.

5.6. Para comprobar el funcionamiento de *Tomcat*, inicia el servidor ejecutando el *script*:

```
%CATALINA_HOME%\bin\startup.bat (C:\apache-tomcat-7.0.40\bin\startup.bat)
```

5.7. Comprueba que el servidor está escuchado en el puerto **TCP 8080**:

```
netstat -a -p TCP -n
```

5.8. Accede a `http://localhost:8080`.

5.9. Puedes ver los ficheros *logs* de *Tomcat* en el directorio *CATALINA_HOME/logs*. En concreto los ficheros **catalina.yyyy-mm-dd.log** contendrán los mensajes relativos al arranque de *Tomcat*.

5.10. Para parar *Tomcat* ejecuta el script:

```
%CATALINA_HOME%\bin\shutdown.bat (C:\apache-tomcat-7.0.40\bin\shutdown.bat)
```

5.11. Finalmente, vamos a acceder a la documentación de *Tomcat* ubicada en el directorio **CATALINA_HOME\webapps\docs\index.html**, Figura 7.9.

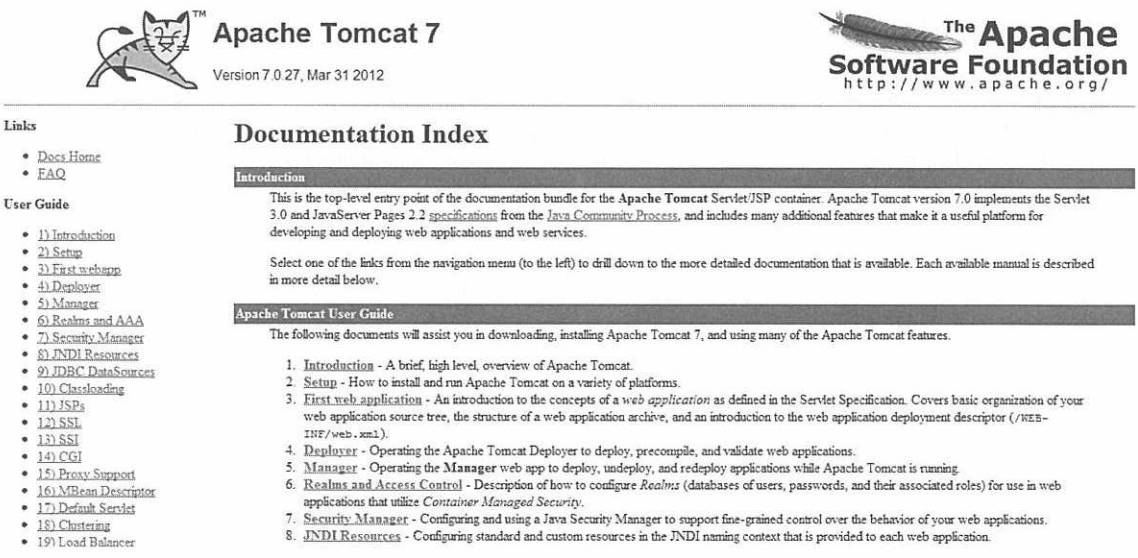


Figura 7.9: Documentación *Tomcat*

6. Ejecución de *Servlets* en *Tomcat*.

6.1. Para terminar de verificar el correcto funcionamiento de la instalación vamos a ejecutar algún *servlet* de los que vienen incluidos con la instalación de *Tomcat*.

6.2. Arranca el servidor y accede al mismo. En la página de inicio selecciona **Examples**.

6.3. Ejecuta el *servlet HelloWorld*.

7. **Errores típicos en la instalación de *Tomcat*.**

   7.1. El error más común viene dado por tener instalado en la misma máquina un servidor web que esté escuchando en el puerto **8080**. Este es el puerto por defecto que *Tomcat* utiliza para escuchar las peticiones y lo hace en este puerto para diferenciarlo del puerto **80**, que es el habitualmente empleado por los servidores Web. Para cambiar el puerto por el que escucha peticiones *Tomcat* hay que editar el fichero de configuración **CATALINA_HOME\conf\server.xml**, cambiarlo por un puerto que no esté en uso y reiniciar *Tomcat*.

   7.2. Otros errores que se puede producir pueden ser debidos a la presencia de un servidor *proxy*. Si estamos accediendo de forma local, debemos indicar en la configuración del navegador que no utilice el servidor *proxy* para las direcciones locales.

   7.3. Si la ejecución del *servlet* no funcionara deberíamos comprobar que la variable de entorno *JAVA_HOME* apunta al directorio donde está instalado *JRE*.

   7.4. Si realizamos la instalación automática de *Tomcat* al reiniciar el equipo puede que obtengamos el mensaje de error: "acceso denegado unable to open the service tomcat 7". Para solucionarlo debemos ir al fichero **C:\Program Files\Apache Software Foundation\Tomcat 7.0\bin\Tomcat7w** y marcar la casilla **Ejecutar este programa como administrador**.

7.2. Instalación y configuración básica de *Eclipse*

En esta práctica se instalará la herramienta de desarrollo *Eclipse* en la máquina virtual **DesarrolloW7XX**. Como en dicho equipo tenemos instalado el servidor de aplicaciones *Tomcat 7*, resultará sencillo probar los diferentes desarrollos que vayamos realizando.

Eclipse es un entorno de desarrollo de *software* compuesto por un *IDE* y por un conjunto de *plugins* que permiten extender su funcionalidad. Es un proyecto *opensource* controlado por la *Eclipse fundation*, `eclipse.org`. Aunque inicialmente fue lanzado por *IBM* en la actualidad es patrocinado por diferentes empresas de *software*. *Eclipse* provee herramientas que facilitan el desarrollo de aplicaciones en *Java*, lo que constituye su principal utilidad aunque no la única.

Como *Eclipse* es una aplicación basada en *Java*, para su funcionamiento es necesario tener instalado la máquina virtual de *Java* (*JRE*).

Además, para desarrollar *JSPs* y *Servlets* usaremos las librerias respectivas, contenidas en la carpeta **lib** de la instalación de *Tomcat*.

1. **Instalación de *Eclipse*.**

   1.1. Inicia sesión en el equipo **DesarrolloW7XX**, accede a la página de descargas de *Eclipse*, `http://www.eclipse.org/downloads/`, y elige *Eclipse IDE for Java EE Developers*, Figura 7.10. Esta versión de *Eclipse* nos facilitará el despliegue de aplicaciones en el servidor *Tomcat*. Frente a otras versiones, la denominada *Eclipse IDE for Java EE Developers* incluye integración con los servidores de aplicaciones más frecuentes y diferentes herramientas para el desarrollo de aplicaciones *Java Enterprise Edition*, en el siguiente enlace puedes ver una comparativa de las funcionalidades de cada versión: `http://www.eclipse.org/downloads/compare.php`.

Figura 7.10: *Eclipse IDE for Java EE Developers*

1.2. Descarga el fichero **.zip** en tu equipo y descomprímelo. Como *Eclipse* es una aplicación portable que no necesita instalación, simplemente crea un acceso directo en el escritorio al fichero **eclipse.exe**.

2. **Configuración básica de *Eclipse*.**

2.1. Al arrancar *Eclipse*, **eclipse.exe**, deberemos elegir un directorio como espacio de trabajo, *workspace*, Figura 7.11. En este directorio se guardarán opciones de configuración de *Eclipse* y los diferentes proyectos que vayamos desarrollando.

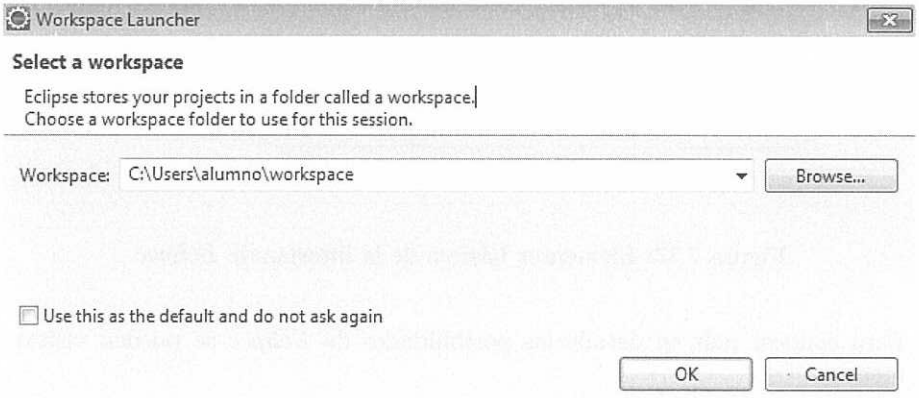


Figura 7.11: *Workspace*

2.2. Una vez iniciada la aplicación obtendremos información sobre diferentes tutoriales relativos a la herramienta, Figura 7.12.

2.3. Una vez cerrada la ayuda accedemos al entorno de desarrollo, dividido en diferentes partes según la perspectiva que elijamos. Por defecto está activada la perspectiva *Java EE*, Figura 7.13.

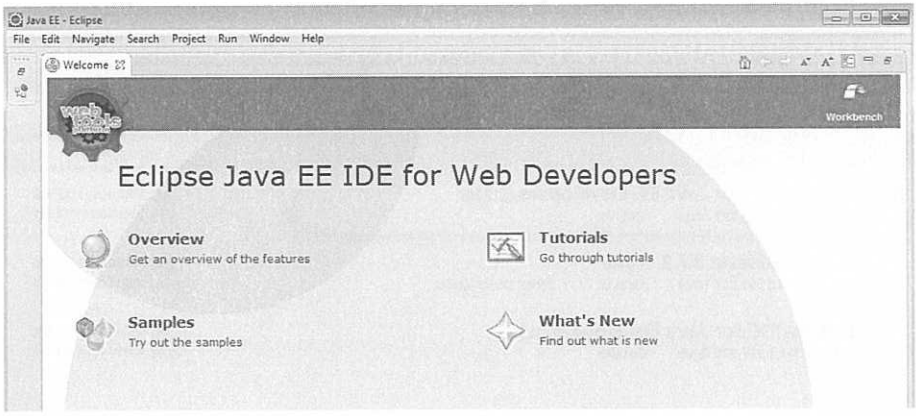


Figura 7.12: Inicio de *Eclipse*

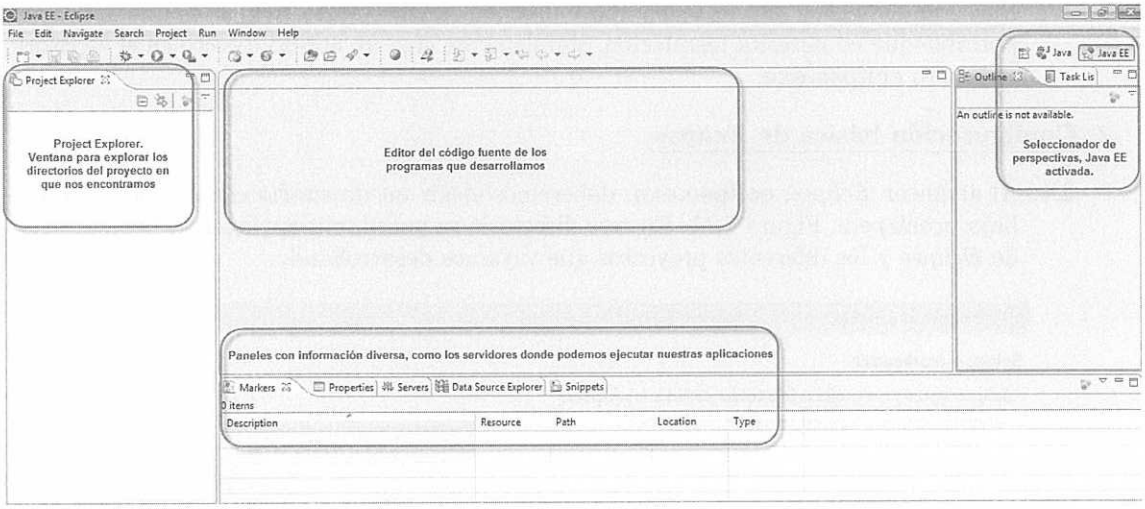


Figura 7.13: Elementos básicos de la interfaz de *Eclipse*

2.4. Para conocer más en detalle las posibilidades de *Eclipse* se pueden visitar diferentes recursos:

- `http://www.eclipse.org/resources/`
- `http://www.eclipse.org/documentation/`
- `http://eclipsetutorial.sourceforge.net/`

7.3. Integración de *Eclipse* con *Tomcat*

En esta práctica integraremos *Eclipse* con *Tomcat* de forma que podamos probar nuestros desarrollos directamente en el servidor de aplicaciones.

1. **Registro del servidor *Tomcat* en *Eclipse*.**

   1.1. Inicia sesión en el equipo **DesarrolloW7XX** con el usuario **alumno** e inicia *Eclipse*.

Estará configurado con la perspectiva *Java EE*. Si no fuera así, puedes cambiarla en la opción de menú **Windows, Open Perspective**.

1.2. En el menú **Windows, Show View**, accede a la opción **Servers**, Figura 7.14.

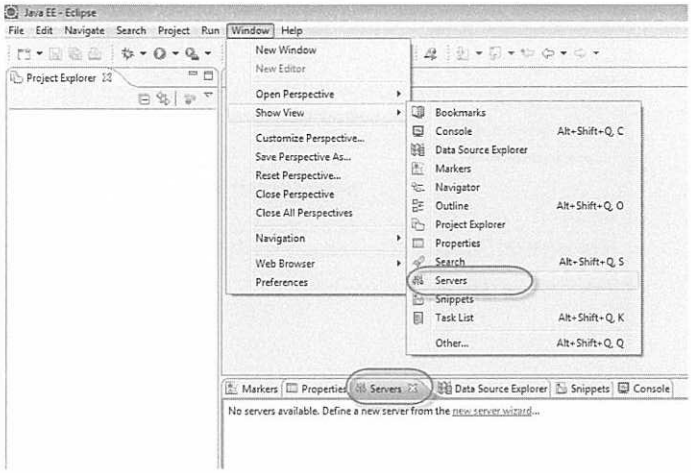


Figura 7.14: Añadir servidor de aplicaciones a *Eclipse*

1.3. Pulsa en **new server wizard** o **botón derecho del ratón, new, server** y elige el servidor de aplicaciones que tenemos instalado, **Apache Tomcat 7**, Figura 7.15. Observa cómo *Eclipse* puede integrarse con otros servidores de aplicaciones como *WebSphere*, *JBOSS*, etc.

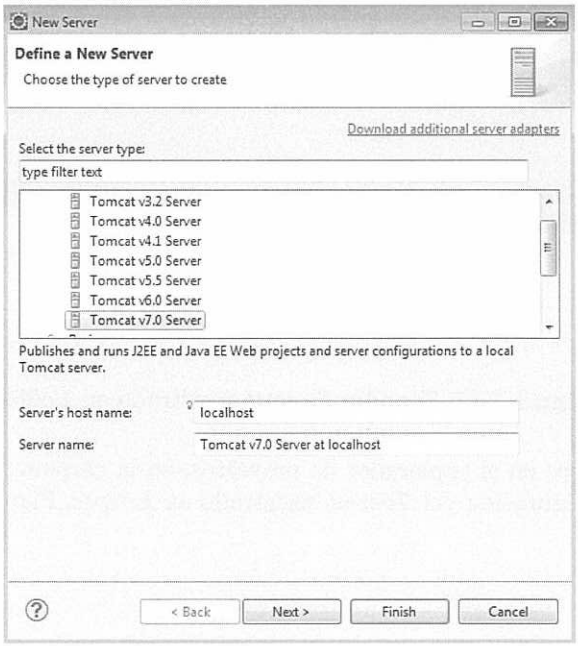


Figura 7.15: Añadir servidor de aplicaciones a *Eclipse*

1.4. Especifica el nombre que quieres dar al servidor, el directorio donde está instalado, la máquina virtual de *Java* y pulsa en **Finalizar**, Figura 7.16.

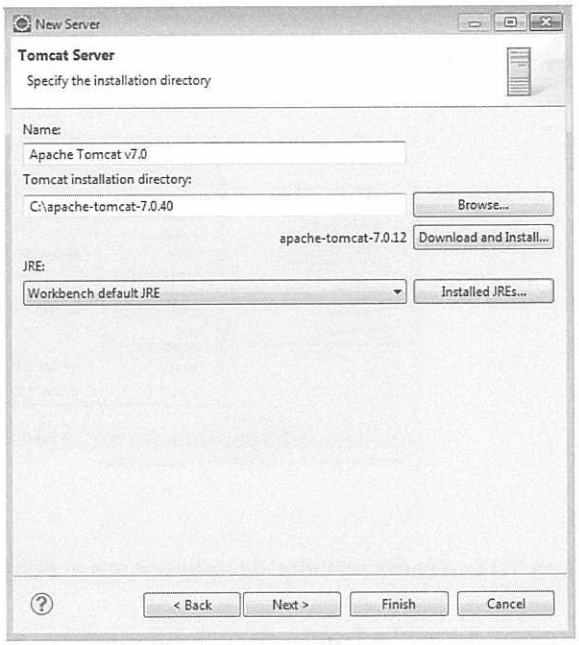


Figura 7.16: Añadir servidor de aplicaciones a *Eclipse*

2. **Relación entre el Servidor *Tomcat* registrado en *Eclipse* y la instalación de *Tomcat*.**

Una vez registrado el servidor *Tomcat* en *Eclipse* veremos que aparece en la pestaña *Servers*, Figura 7.17.

Figura 7.17: Servidor *Tomcat* registrado en *Eclipse*

También nos aparece en el explorador de proyectos, en la carpeta *Servers*, una carpeta con los ficheros de configuración del *Tomcat* registrado en *Eclipse*, Figura 7.18.

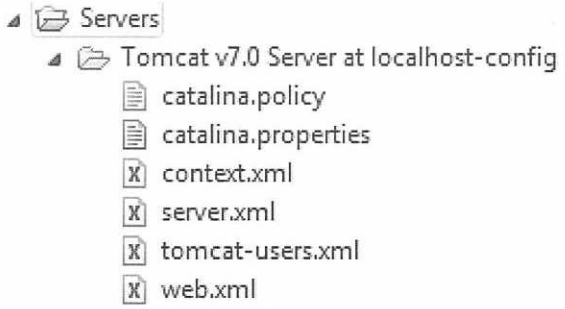


Figura 7.18: Ficheros de configuración del servidor *Tomcat* registrado en *Eclipse*

Esto quiere decir que el servidor *Tomcat* instalado en *Eclipse* tiene sus propios ficheros de configuración diferentes de los de la instación de *Tomcat*. Si hacemos doble clic sobre el servidor *Tomcat* en la pestaña *Servers* podemos comprobar cómo la configuración se guarda en la *metadata* del *workspace* de *Eclipse* y no se modifica la instalación original de *Eclipse*, Figura 7.19.

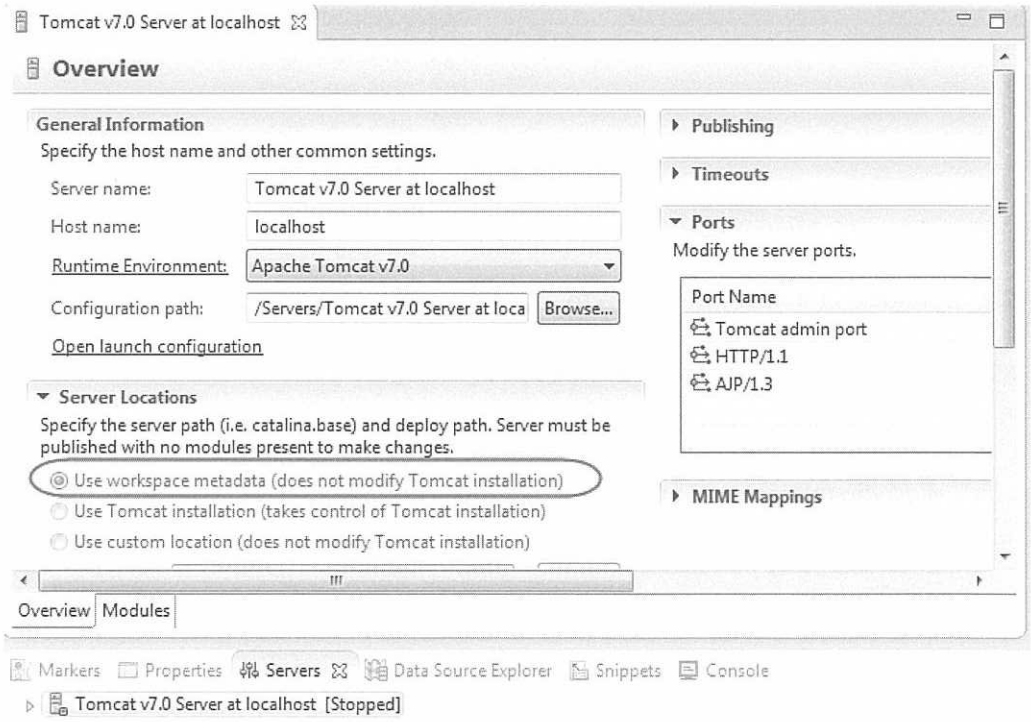


Figura 7.19: Configuración del servidor *Tomcat* registrado en *Eclipse*

Se puede comprobar también cómo *Tomcat* está preparado para republicar automáticamente los proyectos cuando se realicen cambios en los ficheros fuente, Figura 7.20.

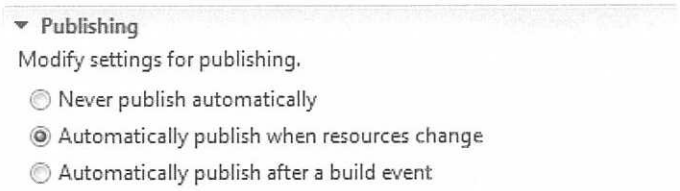


Figura 7.20: Republicación automática

3. **Prueba de la integración entre *Eclipse* y *Tomcat*.**

Para comprobar la integración entre *Eclipse* y *Tomcat* vamos a realizar un primer proyecto.

3.1. Crea un proyecto web dinámico pulsando en **File, New, Dynamic Web Project**, Figura 7.21.

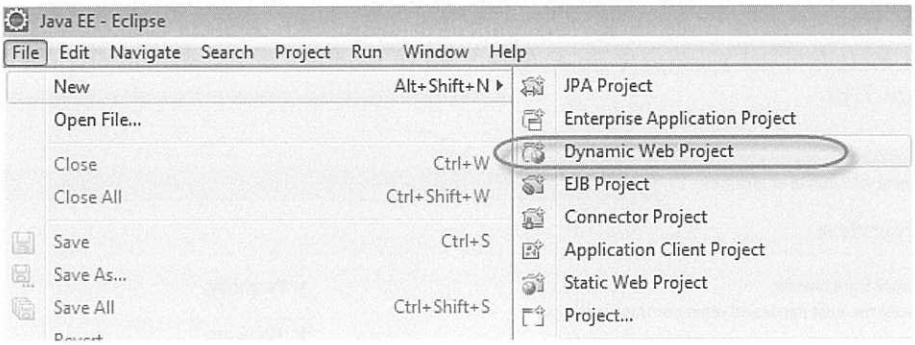


Figura 7.21: Creación de un proyecto web dinámico

3.2. Introduce el nombre del proyecto, en nuestro caso **PrimerProyecto**, y pulsa en **Siguiente**, Figura 7.22.

3.3. Las dos siguientes ventanas, Figuras 7.23 y 7.24, nos permiten establecer:

- El directorio donde estarán las fuentes de *Java* (**src**).
- El directorio donde se guardarán los archivos precompilados **build\classes**.
- El directorio por el que empezarán todas las URLs de los recursos de nuestro proyecto (**Context root**), que por defecto *Eclipse* utiliza el nombre del proyecto.
- El directorio de contenidos (**WebContent**) donde se guardarán los archivos **html** y **jsp**.
- Si queremos crear un descriptor de despliegue (**web.xml**) para la aplicación.

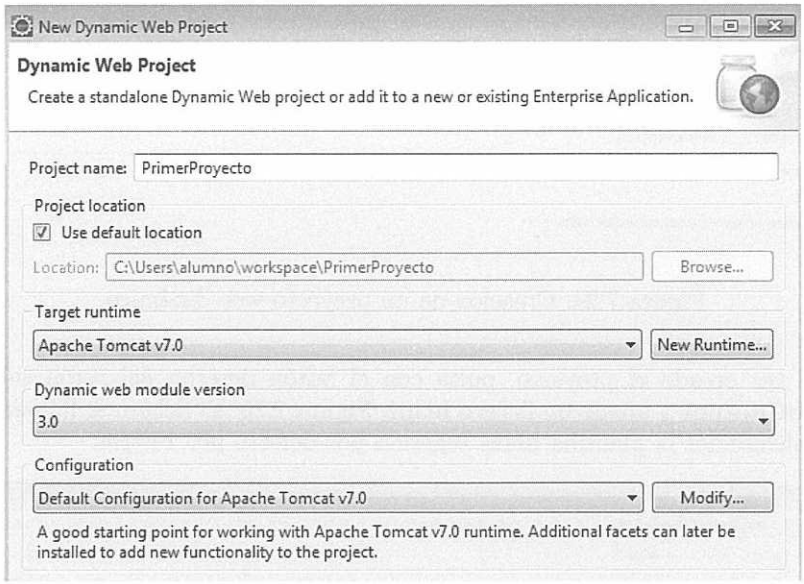


Figura 7.22: Creación de un proyecto web dinámico

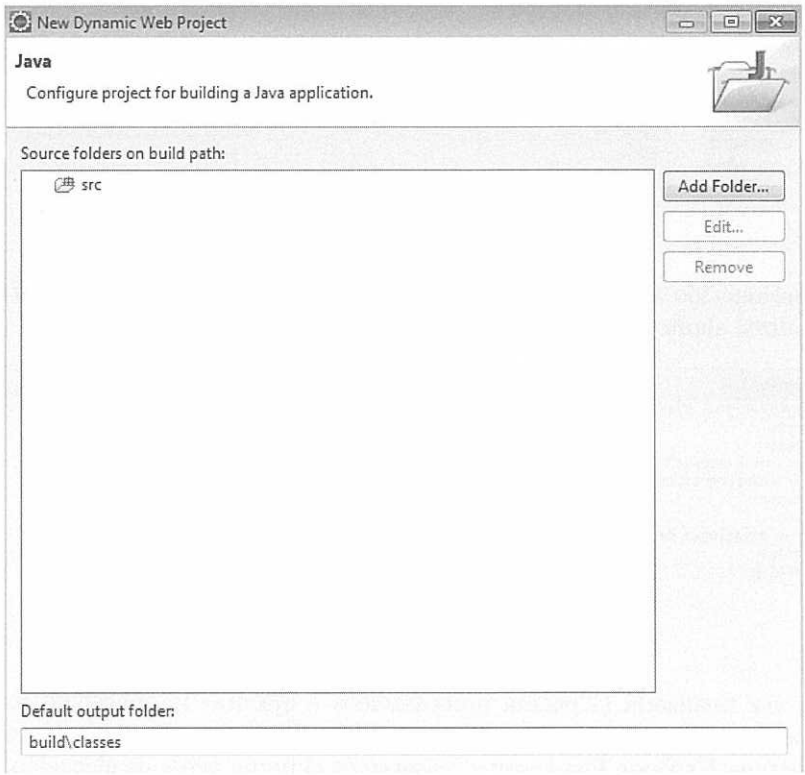


Figura 7.23: Creación de un proyecto web dinámico

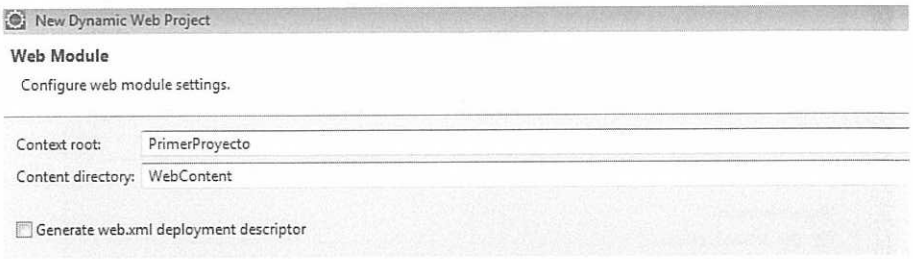


Figura 7.24: Creación de un proyecto web dinámico

3.4. Una vez creado el proyecto, pulsa con el botón derecho del ratón sobre la carpeta **WebContent** y añade un fichero **html**, Figura 7.25. El fichero se llamará **index.html** y utilizaremos la plantilla **html** sugerida por defecto por *Eclipse*.

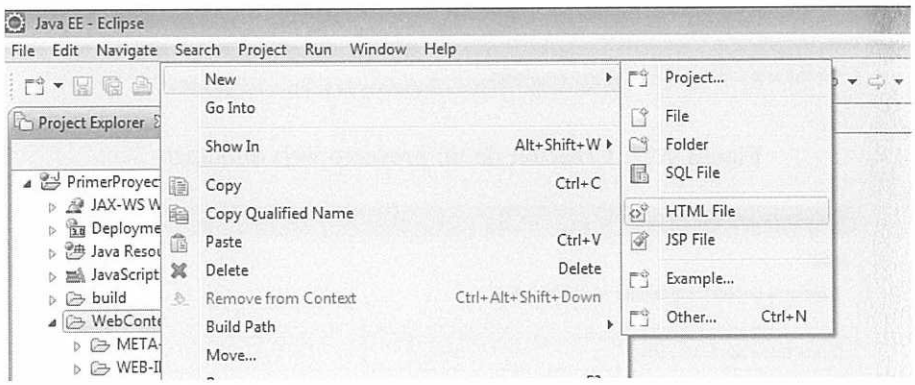


Figura 7.25: Añadir fichero **index.html**

3.5. A continuación añade algún texto dentro del cuerpo del fichero **index.html** para poder visualizar algún contenido en la página, Figura 7.26.

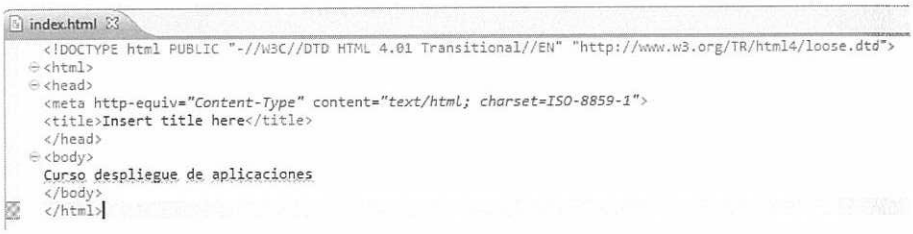


Figura 7.26: Edición del fichero **index.html**

3.6. Una vez finalizada la página procederemos a ejecutar la aplicación sobre el servidor Tomcat que definimos en el primer apartado. Para ello seleccionaremos el proyecto en la ventana **Project Explorer** y pulsaremos el botón verde de ejecución. Nos aparecerá un asistente para elegir el servidor donde se ejecutará la aplicación, elige el definido en el apartado inicial de la práctica, Figura 7.27.

3.7. Es importante que antes de iniciar *Tomcat* comprobemos que el servidor no haya sido arrancado previamente.

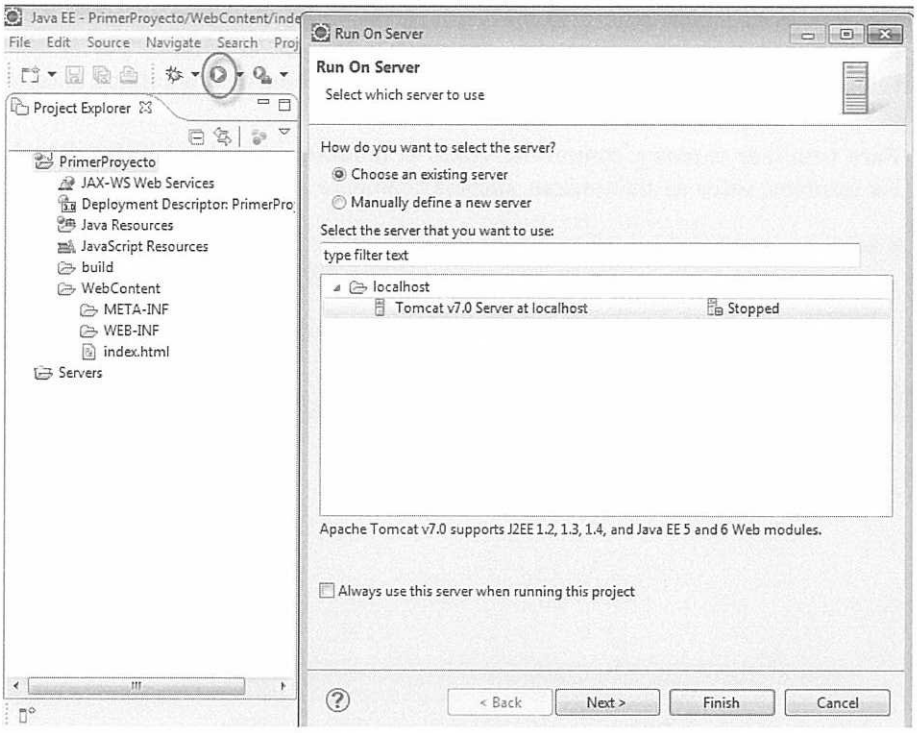


Figura 7.27: Ejecución del proyecto

3.8. Al terminar el asistente y pulsar en **Finalizar** nuestro proyecto se ejecutará en el servidor *Tomcat*, Figura 7.28.

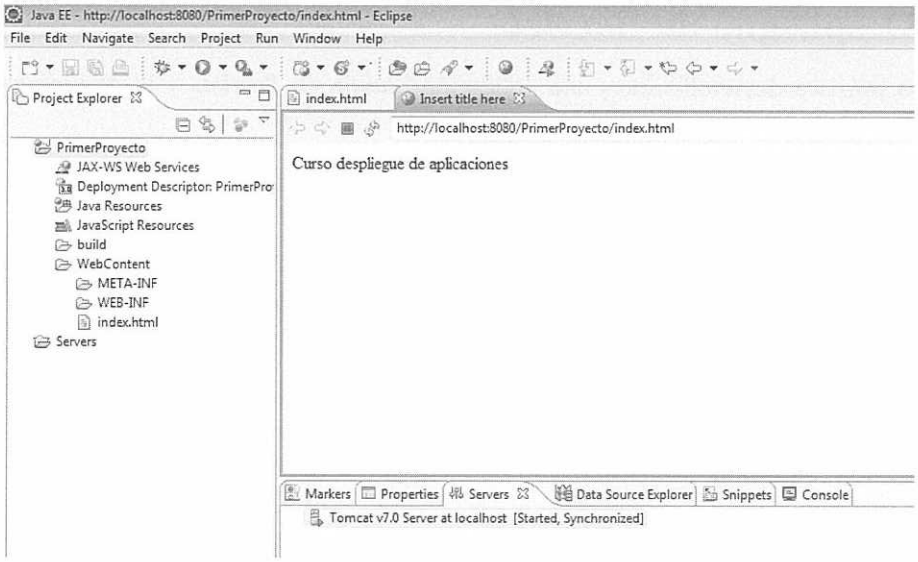


Figura 7.28: Ejecución del proyecto

3.9. Observa la dirección `http://localhost:8080/PrimerProyecto/index.html`, aparece el nombre del equipo y el puerto que utiliza *Tomcat*, el nombre por el que empiezan todos los recursos de nuestro proyecto (el *Context root* que definimos anteriormente) y el fichero al que estamos accediendo.

3.10. Para terminar vamos a comprobar cómo al modificar el fichero **index.html** y guardar los cambios, estos se sincronizan automáticamente con el servidor *Tomcat*, Figura 7.29.

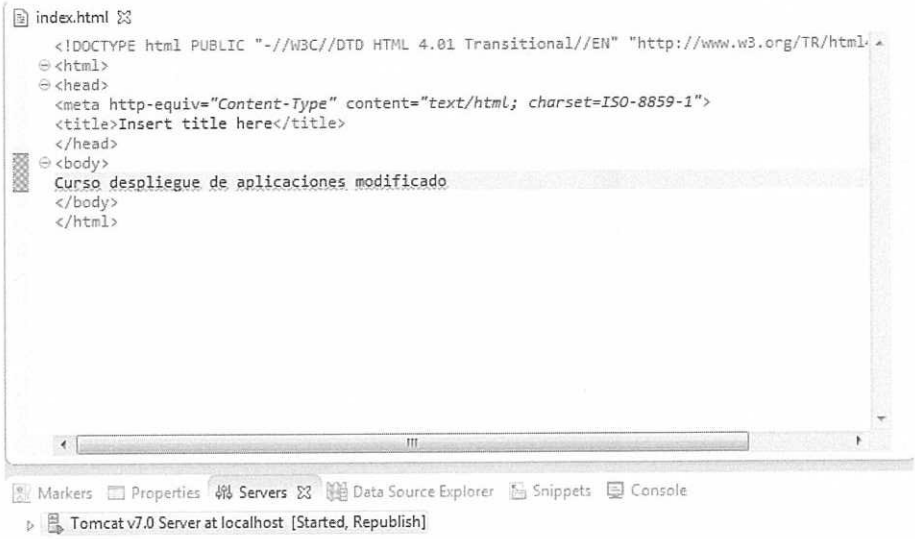


Figura 7.29: Republicación del proyecto

7.4. Despliegue de *Servlets*

En esta práctica desarrollaremos una aplicación basada en *Servlets* y la desplegaremos en el servidor *Tomcat*.

1. **Creación de una aplicación HolaMundo**.

   1.1. Inicia sesión en el equipo **DesarrolloW7XX** y arranca *Eclipse*.

   1.2. En *Eclipse*, crea un nuevo *Dynamic Web Project* de nombre **HolaMundo**. Marca la opción de crear un descriptor de despliegue de la aplicación, Figura 7.30.

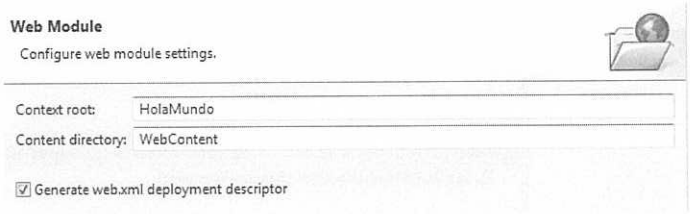


Figura 7.30: Crear descriptor de despliegue de la aplicación

1.3. Si observas la estructura de carpetas del proyecto creado, Figura 7.31.

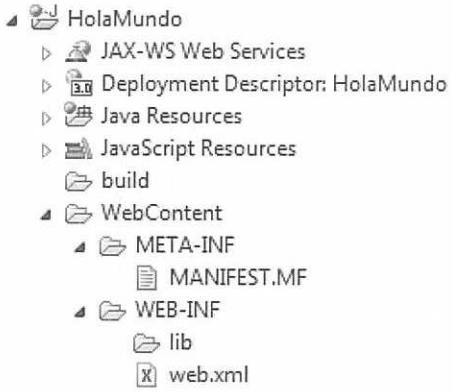


Figura 7.31: Estructura carpetas proyecto

Podrás ver los siguientes elementos:

- **Deployment Descriptor**. Acceso al descriptor de despliegue guardado en el fichero **WEB-INF\web.xml**.
- Carpeta **WebContent**. Donde colocaremos todos los recursos Web, como ficheros *HTML*, *JSPs*, imágenes, etc. Los ficheros no colocados en este directorio, o en algún subdirectorio del mismo, no estarán disponibles cuando la aplicación se ejecute en el servidor. Todos los ficheros fuera de este directorio se consideran archivos de desarrollo, no de producción. Por ejemplo ficheros **.java**, **.sql**, etc.
- Carpeta **WebContent\META-INF**. Contiene el fichero **MANIFEST.MF**, utilizado para mapear clases de ficheros JAR existentes en otros proyectos pertenecientes al mismo *Enterprise Application project*.
- Carpeta **WebContent\WEB-INF**. Este directorio contiene el descriptor de despliegue de la aplicación, **web.xml**.
- Carpeta **Java Resources**. Donde colocaremos nuestros *Servlets* y nuestros ficheros **.java**.

1.4. Ahora añade una página **HolaMundo.html** a nuestro proyecto y edítala para incluir el mensaje Hola Mundo.

1.5. En vez de ejecutar directamente el proyecto en el servidor *Tomcat* que tenemos registrado en *Eclipse*, vamos a exportarlo como un archivo **war**, *Web archives*, Figuras 7.32 y 7.33. Puedes guardar el fichero exportado en cualquier ubicación.

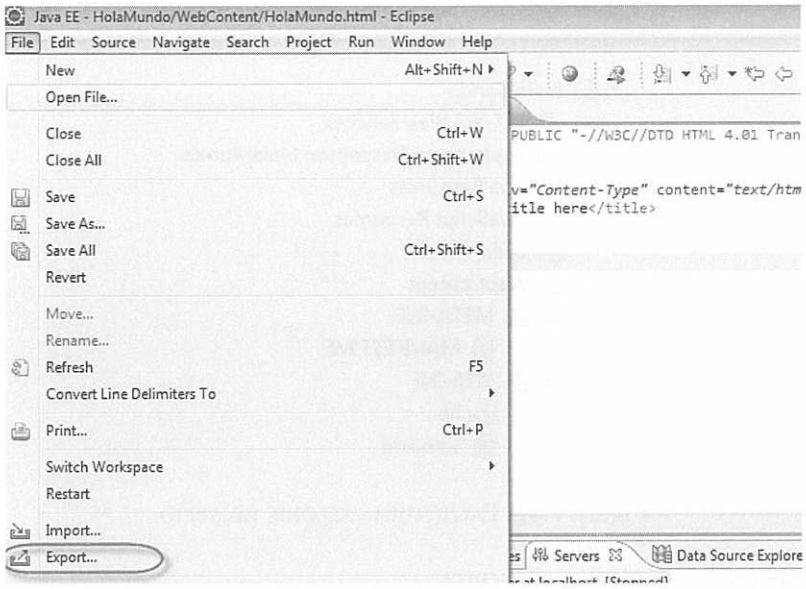


Figura 7.32: Exportar proyecto

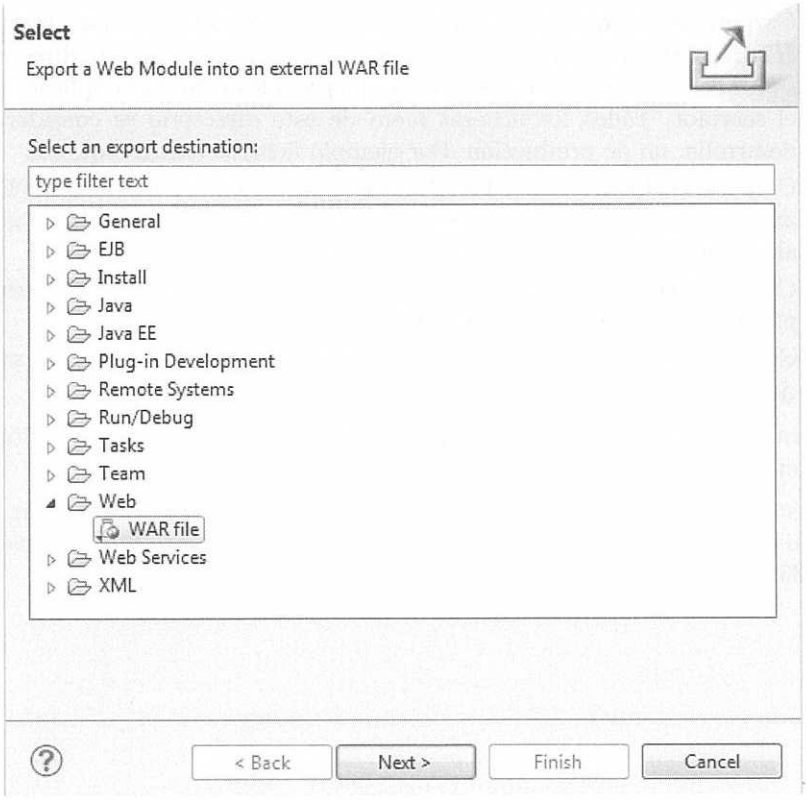


Figura 7.33: Exportar proyecto

1.6. Inicia ahora el servidor *Tomcat*, ejecutando el *script startup*, situado en **CATALINA_HOME\bin**.

1.7. Para desplegar el proyecto en *Tomcat*, copia el fichero **HolaMundo.war** en la carpeta **CATALINA_HOME\webapps**. Automáticamente se desplegará el proyecto.

1.8. A continuación accede a la dirección `http://localhost:8080/HolaMundo/HolaMundo.html` desde el navegador.

1.9. En *Eclipse*, renombra el fichero **HolaMundo.html**, llamándolo **index.html** y vuelve a exportar y desplegar el proyecto. Comprueba cómo ahora puedes acceder también con las *urls*:

- `http://localhost:8080/HolaMundo/index.html`
- `http://localhost:8080/HolaMundo`

1.10. Esto es debido a que el **index.html** es uno de los nombres especificados en el descriptor de despliegue como página de inicio.

1.11. Para finalizar observa cómo en el directorio **CATALINA_HOME\webapps** se ha creado automáticamente un directorio **HolaMundo** con todos los componentes del proyecto, Figura 7.34

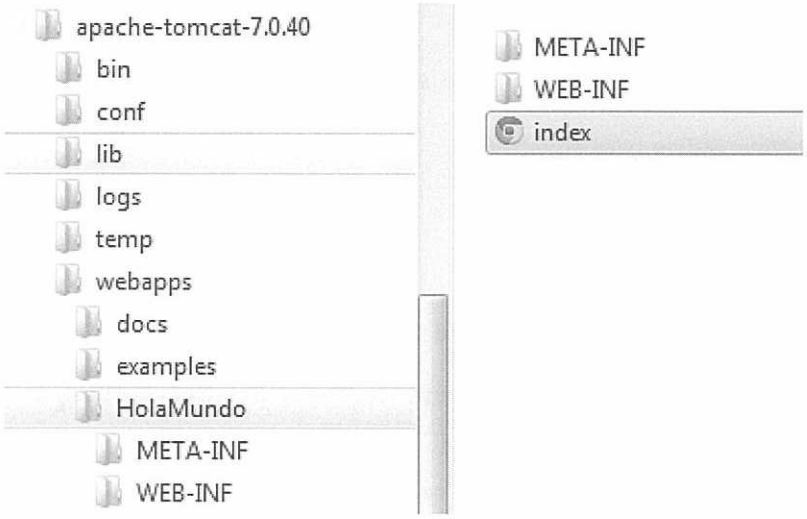


Figura 7.34: Proyecto desplegado

2. **Creación de un *servlet* HolaMundo.**

2.1. En *Eclipse*, en el mismo proyecto, colócate sobre el directorio **Java Resources\src** y añade un nuevo *servlet* llamado *HolaMundoServlet*, Figura 7.35. No crearemos ningún paquete, simplemente se creará en el paquete por defecto.

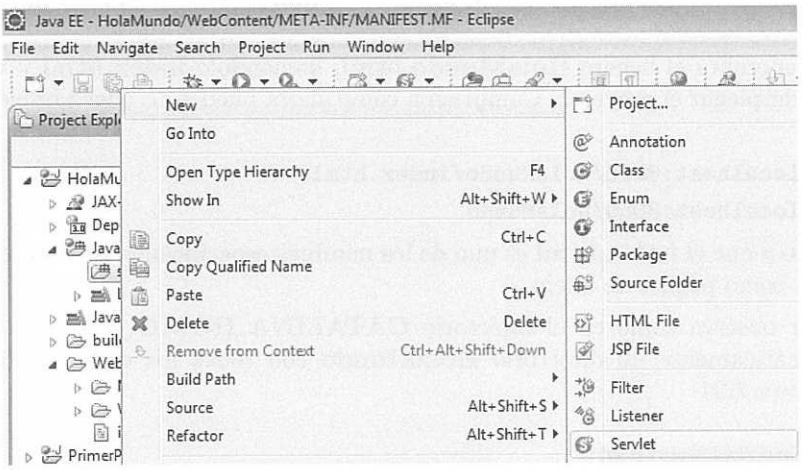


Figura 7.35: Nuevo *servlet*

Utiliza el siguiente código para el *servlet*:

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class HolaMundoServlet extends HttpServlet {
   @Override
   public void doGet(HttpServletRequest request, HttpServletResponse response)
         throws IOException, ServletException {

      // Establece el tipo MIME del mensaje de respuesta
      response.setContentType("text/html");
      // Crea un flujo de salida para escribir la respuesta a la petición del cliente
      PrintWriter out = response.getWriter();

      // Escribe el mensaje de respuesta en una página html
      try {
         out.println("<html>");
         out.println("<head><title>Hola Mundo</title></head>");
         out.println("<body>");
         out.println("<h1>Hola Mundo</h1>");  // dice hola
         // Muestra información de la petición del cliente
         out.println("<p>Request URI: " + request.getRequestURI() + "</p>");
         out.println("<p>Protocolo: " + request.getProtocol() + "</p>");
         out.println("<p>Dirección remota: " + request.getRemoteAddr() + "</p>");
         // Genera un número aleatorio para cada petición
```

```
            out.println("<p>Número aleatorio: <strong>" + Math.random() + "</strong></p>");
            out.println("</body></html>");
        } finally {
            out.close();  // Cierra el flujo de salida
        }
    }
}
```

2.2. Desde *Eclipse* vamos a editar el descriptor de despliegue, **web.xml**, de la aplicación introduciendo el siguiente contenido.

```
<!-- Indicamos que la clase HolaMundoServlet se corresponde
con el nombre de servlet HolaMundo -->
<servlet>
   <servlet-name>HolaMundo</servlet-name>
   <servlet-class>HolaMundoServlet</servlet-class>
</servlet>

<!-- Indicamos que el nombre de servlet HolaMundo corresponde a la url /DiHola -->
<servlet-mapping>
   <servlet-name>HolaMundo</servlet-name>
   <url-pattern>/DiHola</url-pattern>
</servlet-mapping>
```

Para editar el descriptor de despliegue **web.xml** seleccionaremos la pestaña **Source**, Figura 7.36. Ten en cuenta que para cada *servlet* tendrás que indicar un elemento *servlet* y un elemento *servlet-mapping* que compartan el *servlet-name*.

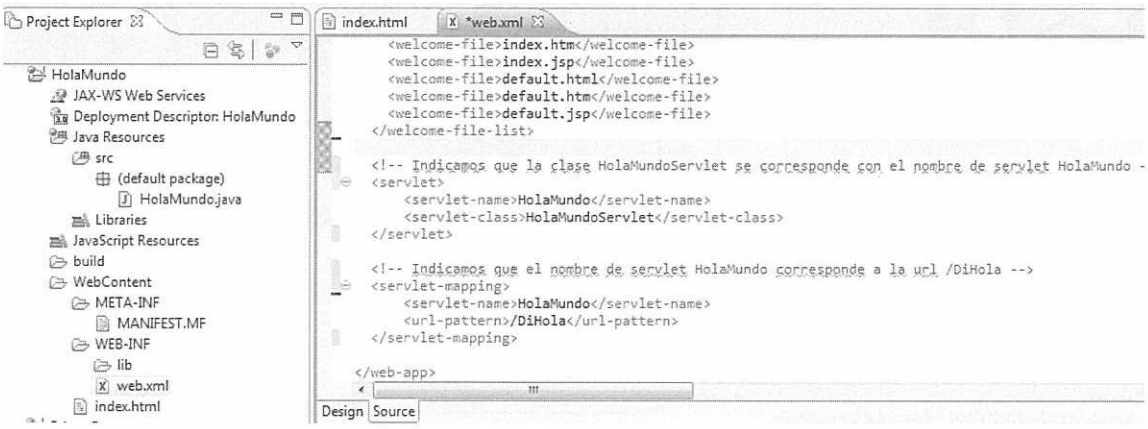


Figura 7.36: Editar descriptor de despliegue **web.xml**

2.3. Para todas las prácticas, es aconsejable copiar el código del fichero **pdf** a un editor de texto plano, como el bloc de notas, y de aquí a *Eclipse*. Así evitaremos errores debidos a caracteres presentes en el **pdf** que pueden provocar errores.

2.4. Para que el código copiado y pegado aparezca bien formateado, utiliza la opción **Format** del menú **Source** de *Eclipse*.

2.5. A continuación exportamos el proyecto a un archivo **.war** y lo desplegamos en el servidor.

2.6. Para acceder al *Servlet* utilizaremos la dirección `http://localhost:8080/HolaMundo/DiHola` y veremos el resultado mostrado en la Figura 7.37.

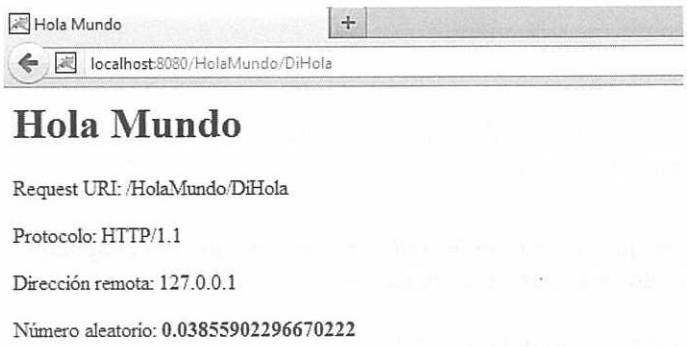


Figura 7.37: *Servlet* HolaMundo

2.7. En la siguiente figura podemos ver el resumen de funcionamiento desde la petición por parte del cliente a la ejecución del *servlet*, Figura 7.38.

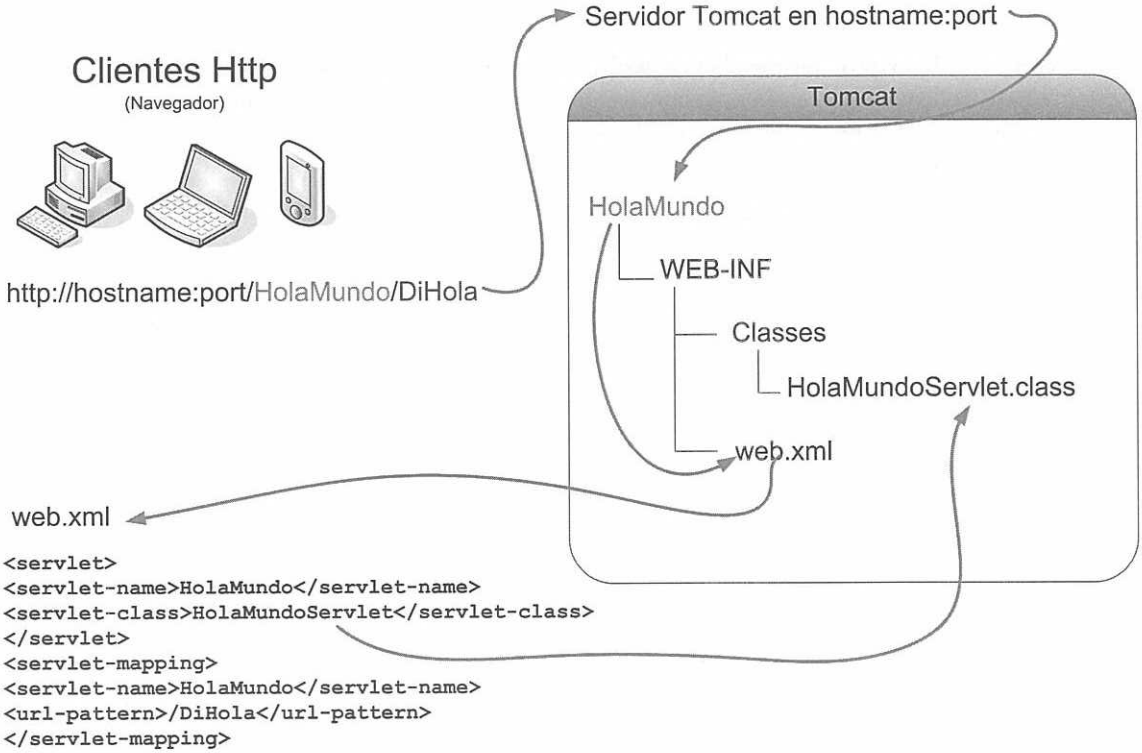


Figura 7.38: *Servlet* HolaMundo

2.8. Para finalizar comprueba cómo en la carpeta: **CATALINA_HOME\webapps\HolaMundo\WEB-INF\classes** aparece el fichero **HolaMundoServlet.class** correspondiente al fichero **HolaMundoServlet.java** que creamos en *Eclipse*, Figura 7.39.

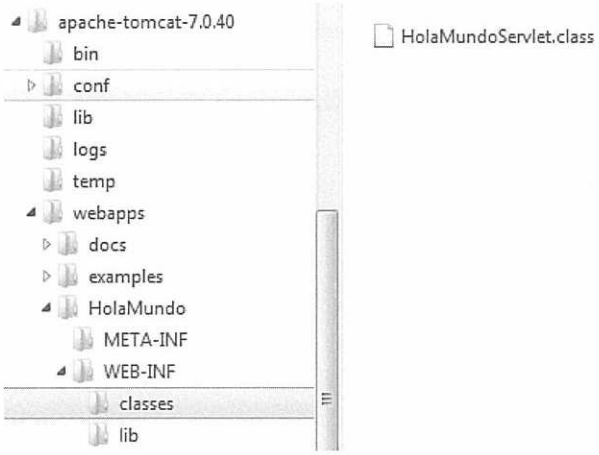


Figura 7.39: *Servlet* HolaMundo

7.5. *Servlet* con acceso a base de datos con *JDBC*

En esta práctica desarrollaremos una aplicación basada en un *Servlet* que accederá a una base de datos *MySQL* mediante un conector *JDBC*.

1. **Instalación de *MySQL*.**

   1.1. Inicia sesión en el equipo **DesarrolloW7XX**.

   1.2. Para simplificar la instalación descarga el instalador para *Windows* de *MySQL* de la siguiente dirección `http://dev.mysql.com/downloads/installer/`.

   1.3. También necesitarás el instalador independiente de *Microsoft .NET Framework 4*, que puedes descargar de la dirección: `http://www.microsoft.com/en-us/download/details.aspx?id=17718`.

   1.4. Ejecuta el instalador de *Microsoft .NET Framework 4* y reinicia el equipo.

   1.5. A continuación ejecuta el instalador de *MySQL*, eligiendo las siguientes opciones:

   - Tipo de instalación: *Developer Default.*
   - *Instalation Path*: **C:\MySQL**.
   - *MYSQL Server Configuration Type*: *Development Machine.*
   - *Enable TCP/IP Networking*: *Port Number 3306.*
   - *MySQL Root Password*: despliegue.
   - *Create Windows Service, Windows Service Name: MySQL56.*

   1.6. Una vez concluida la instalación comprueba en **Panel de control, Servicios**, que el servicio *MySQL56* está iniciado.

2. **Creación de la base de datos TiendaLibros.**

   2.1. Inicia sesión en *MySQL* con el comando:

   ```
   C:\MySQL\bin> mysql -u root -p
   ```

   e introduce el *password* que establecimos en la instalación. También puedes iniciar el **MySQL 5.6 Command Line Client**.

   2.2. Para crear la base de datos **TiendaLibros** y la tabla **libros**, crea el siguiente *script* en el fichero **C:\MySQL\TiendaLibros.sql**:

   ```
   create database TiendaLibros;

   use TiendaLibros;

   create table libros (
      id      int,
      titulo  varchar(50),
      autor varchar(50),
      precio  float,
      cantidad    int,
      primary key (id));

   insert into libros values
   (1001, 'Servicios de Red e Internet', 'Alvaro Garcia', 25, 100);
   insert into libros values
   (1002, 'Apache Tomcat 7', 'Aleksa Vukotic', 22.22, 22);
   insert into libros values
   (1003, 'Beginning JSP, JSF,and Tomcat Web Development', 'Giulio Zambon', 33.33, 33);
   insert into libros values
   (1004, 'Tomcat, the definitive guide', 'Jason Brittain', 55.55, 55);
   insert into libros values
   (1005, 'Profesional Apache Tomcat', 'Vivek Chopra', 66.66, 66);
   insert into libros values
   (1006, 'Despliegue de aplicaciones Web', 'Alvaro Garcia', 22, 100);

   select * from libros;
   ```

   2.3. Ejecuta el *script* con el comando:

   ```
   source C:\MySQL\TiendaLibros.sql
   ```

3. **Descarga del conector *JDBC*.**

JDBC es un *API* de *Java* para ejecutar sentencias *SQL*. Está formado por un conjunto de clases e interfaces programadas con el propio *Java*. Permite interactuar con bases de datos, de forma transparente al tipo de la misma. Es decir, es una forma única de programar el acceso a bases de datos desde *Java*, independiente del tipo de la base de datos.

   3.1. En nuestro caso vamos a utilizar un *driver* nativo *JDBC* que permite la conectividad entre *Java* y *MySQL*. Puedes descargar este conector desde la propia página de *MySQL*: `http://www.mysql.com/products/connector/`, Figura 7.40.

MySQL Connectors

MySQL provides standards-based drivers for JDBC, ODBC, and .Net enabling developers t
library allows developers to embed MySQL directly into their applications.

Developed by MySQL	
ADO.NET Driver for MySQL (Connector/NET)	Download
ODBC Driver for MySQL (Connector/ODBC)	Download
JDBC Driver for MySQL (Connector/J)	Download
C++ Driver for MySQL (Connector/C++)	Download
C Driver for MySQL (Connector/C)	Download
C API for MySQL (mysqlclient)	Download
MySQL Connector for OpenOffice.org	

Figura 7.40: *Driver JDBC para MySQL*

3.2. En concreto, descarga el archivo **.zip** con el conector. En el momento de hacer esta documentación era **mysql-connector-java-5.1.25.zip**. Una vez descargado descomprímelo y comprueba que está el archivo **mysql-connector-java-5.1.25-bin.jar** que contiene el conector.

4. ***Servlet* de acceso a *MySQL* con *JDBC***.

4.1. Inicia *Eclipse* y crea un nuevo proyecto *Dynamic Web Project* llamado **AccesoDatos**. Recuerda marcar la opción para que se cree un descriptor de despliegue **web.xml**.

4.2. Para empezar a desarrollar la aplicación vamos a crear una página *HTML* que será la interfaz de entrada con la que interactuará el usuario. Añade una página **ConsultaLibros.html** a la carpeta *WebContent* de tu proyecto **AccesoDatos**. Su código será el siguiente:

```
<html>
<head>
  <title>TiendaLibros</title>
</head>
<body>
  <h2>Tienda Libros</h2>
  <form method="get" action="http://localhost:8080/AccesoDatos/consulta">
    <b>Elige un autor:</b>
    <input type="checkbox" name="autor" value="Alvaro Garcia">Alvaro Garcia
    <input type="checkbox" name="autor" value="Aleksa Vukotic">Aleksa Vukotic
    <input type="checkbox" name="autor" value="Giulio Zambon">Giulio Zambon
    <input type="submit" value="Buscar">
  </form>
</body>
</html>
```

4.3. Ejecuta el proyecto desde *Eclipse* para acceder a la página `http://localhost:8080/AccesoDatos/ConsultaLibros.html`. Para ello pulsa con el botón derecho del ratón sobre el fichero **ConsultaLibros.html**, elige la opción **Run as, Run on server**, Figuras 7.41, 7.42. Esto hará que se inicie el servidor *Tomcat* que tenemos registrado en *Eclipse* y que se ejecute la página seleccionada.

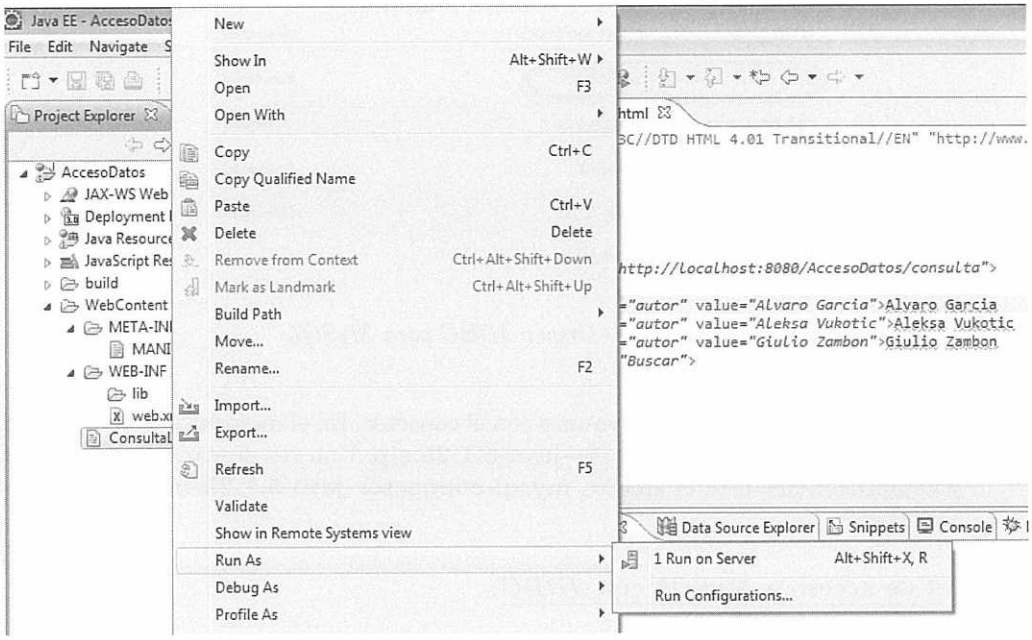


Figura 7.41: Ejecutar aplicación desde *Eclipse*

Figura 7.42: Página *HTML* de acceso a la aplicación

4.4. Si pulsamos el botón **Buscar** no funcionará, porque hemos establecido que al enviar la página se ejecute la *URL* `http://localhost:8080/AccesoDatos/consulta` que corresponderá al *Servlet* que todavía no hemos desarrollado.

```
<form method="get" action="http://localhost:8080/AccesoDatos/consulta">
```

4.5. Ahora vamos a desarrollar el *Servlet* de acceso a la base de datos. Lo primero que tenemos que realizar, para que el servidor *Tomcat* pueda ejecutar una aplicación utilizando el *driver JDBC*, es colocar el fichero **mysql-connector-java-5.1.20-bin.jar** en el directorio **CATALINA_HOME\lib**.

4.6. Reinicia el servidor desde *Eclipse* para que sea posible utilizar el conector.

4.7. Las clases que conforman el *API JDBC* del conector se encuentran agrupadas en el paquete `java.sql`. Este paquete contiene clases para cargar los *drivers*, realizar las conexiones a las bases de datos, consultar los datos y manejar un conjunto de registros. También posee las clases para el manejo de excepciones que se produzcan en el acceso a bases de datos.

4.8. A continuación veremos los pasos que se realizan en una aplicación con acceso a datos mediante *JDBC*:

1 Cargar el *driver JDBC*.

2 Conectarse a la Base de Datos utilizando la clase *Connection*.

3 Crear sentencias *SQL*, utilizando objetos de tipo *Statement*.

4 Ejecutar las sentencias *SQL* a través de los objetos de tipo *Statement*.

5 En caso de que sea necesario, procesar el conjunto de registros resultante utilizando la clase *ResultSet*.

4.9. Ahora crearemos el *Servlet* que se encargará de hacer la consulta a la base de datos. Añade el fichero **ConsultaServlet.java** a la carpeta **Java Resources\src** del proyecto **AccesoDatos** e introduce el siguiente código **Java**.

```
import java.io.*;
import java.sql.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class ConsultaServlet extends HttpServlet {

 // El método doGet() se ejecuta una vez por cada petición HTTP GET.
 @Override
 public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
  // Establecemos el tipo MIME del mensaje de respuesta
  response.setContentType("text/html");
  // Creamos un objeto para poder escribir la respuesta
  PrintWriter out = response.getWriter();

  Connection conn = null;
  Statement stmt = null;
  try {
```

```
//Paso 1: Cargar el driver JDBC.
Class.forName("com.mysql.jdbc.Driver").newInstance();

// Paso 2: Conectarse a la Base de Datos utilizando la clase Connection
String userName="root";
String password="despliegue";
//URL de la base de datos(equipo, puerto, base de datos)
String url="jdbc:mysql://localhost/TiendaLibros";
conn = DriverManager.getConnection(url, userName, password);

  // Paso 3: Crear sentencias SQL, utilizando objetos de tipo Statement
  stmt = conn.createStatement();

  // Paso 4: Ejecutar las sentencias SQL a través de los objetos Statement
  String sqlStr = "select * from libros where autor = "
  + "'" + request.getParameter("autor") + "'";

  // Generar una página HTML como resultado de la consulta
  out.println("<html><head><title>Resultado de la consulta</title></head><body>");
  out.println("<h3>Gracias por tu consulta.</h3>");
  out.println("<p>Tu consulta es: " + sqlStr + "</p>");
  ResultSet rset = stmt.executeQuery(sqlStr);

  // Paso 5: Procesar el conjunto de registros resultante utilizando ResultSet
  int count = 0;
  while(rset.next()) {
   out.println("<p>" + rset.getString("autor")
              + ", " + rset.getString("titulo")
              + ", " + rset.getDouble("precio") + "</p>");
   count++;
  }
  out.println("<p>==== " + count + " registros encontrados =====</p>");
  out.println("</body></html>");
 } catch (Exception ex) {
    ex.printStackTrace();
   } finally {
     out.close();  // Cerramos el flujo de escritura
     try {
          // Cerramos el resto de recursos
          if (stmt != null) stmt.close();
          if (conn != null) conn.close();
       } catch (SQLException ex) {
          ex.printStackTrace();
       }
  }
 }
}
```

4.10. Finalmente y para que este *Servlet* sea accesible edita el descriptor de despligue añadiendo la siguiente información dentro de la etiqueta **web-app**:

```
    ...
    <servlet>
       <servlet-name>ConsultaUsuario</servlet-name>
       <servlet-class>ConsultaServlet</servlet-class>
    </servlet>

    <servlet-mapping>
       <servlet-name>ConsultaUsuario</servlet-name>
       <url-pattern>/consulta</url-pattern>
    </servlet-mapping>
</web-app>
```

4.11. Prueba a ejecutar el proyecto desde *Eclipse* para comprobar su correcto funcionamiento. Recuerda que la página `http://localhost:8080/AccesoDatos/ConsultaLibros.html` será el punto de entrada a la aplicación.

4.12. Marca la opción **Alvaro Garcia** y pulsa sobre el botón **Buscar**. Una petición *HTTP Get* es enviada a la *URL* especificada en el atributo *action* de la etiqueta *form*:

```
<form method="get" action="http://localhost:8080/AccesoDatos/consulta">
```

Observa la *URL* de la petición *HTTP GET*:

`http://localhost:8080/AccesoDatos/consulta?autor=Alvaro+Garcia`

La petición consiste en dos partes, una *URL* correspondiente al atributo *action* y separado por una "?" aparece la pareja "name=value" extraída de la etiqueta *input*. Observa cómo los blancos son reemplazados por "+" porque los espacios en blanco no estan permitidos en las *URLs*.

4.13. Si ocurriera algún error en la ejecución realizada desde *Eclipse* se mostraría un mensaje en la consola, Figura 7.43.

Figura 7.43: La consola mostrará los errores de ejecución

4.14. Ten también presente que no es necesario volver a iniciar el servidor cada vez que realicemos un cambio en el *Servlet*. Cualquier cambio en el código es automáticamente reflejado en el servidor *Tomcat* registrado en *Eclipse*.

4.15. **Despliegue de la aplicación de Acceso a Datos con *JDBC*.**

Una vez que hemos comprobado en el entorno de desarrollo que la aplicación funciona, dentendremos el servidor *Tomcat* desde *Eclipse*, Figura 7.44.

Figura 7.44: Para el servidor *Tomcat* desde *Eclipse*

4.16. Es interesante indicar que la ejecución del servidor *Tomcat* que realizamos desde *Eclipse* utiliza sus propios ficheros de configuración, Figura 7.45, diferentes de los ubicados **CATALINA_HOME\conf**

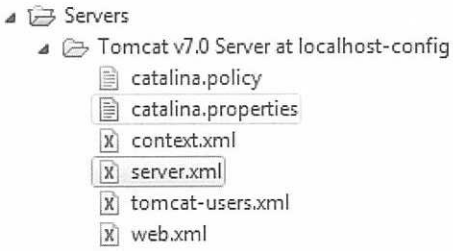


Figura 7.45: Ficheros de configuración de *Tomcat* en *Eclipse*

4.17. Exporta el proyecto **AccesoDatos** a un fichero **AccesoDatos.war**, marcando la opción sobreescribir si el fichero ya existe.

4.18. Inicia ahora el servidor *Tomcat*, ejecutando el *script* de arranque *startup*, situado en **CATALINA_HOME\bin**

4.19. Para desplegar el proyecto en *Tomcat*, copia el fichero **AccesoDatos.war** en la carpeta **CATALINA_HOME\webapps**.

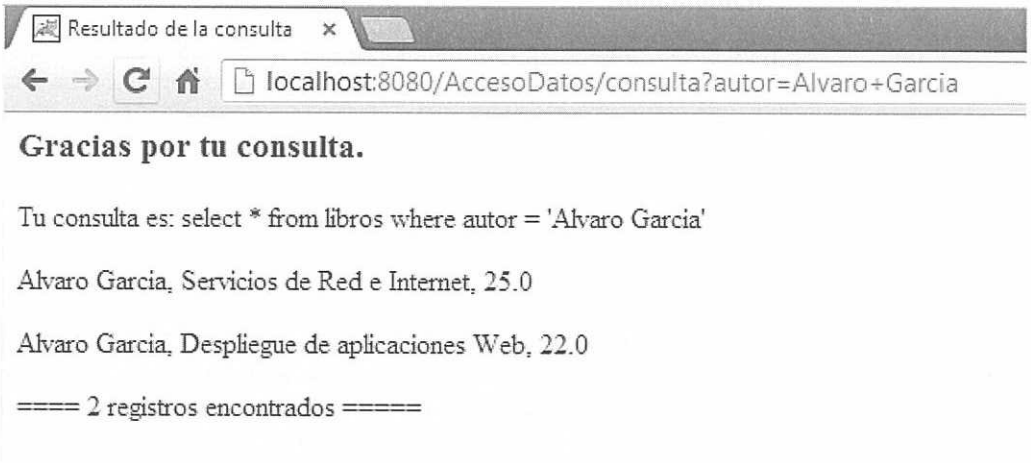


Figura 7.46: Funcionamiento de la aplicación una vez desplegada

4.20. Puedes comprobar cómo funciona desde el navegador, Figura 7.46

5. **Peticiones POST**.

HTTP define los métodos para realizar peticiones: *GET* y *POST*. En este ejemplo hemos usado el método *GET*, como especificamos en la página **ConsultaLibros.html**:

```
<form method="get" action="http://localhost:8080/AccesoDatos/consulta">
```

La petición *GET* es procesada por el método *doGet* del *Servlet ConsultaServlet*.

¿Cuál es la diferencia entre los dos métodos?. En las peticiones *POST* los parámetros se envían en el cuerpo de mensaje, por lo que no son visibles en la *URL* como pasa con las peticiones *GET*. Las ventajas radican en que:

- El cliente no ve una extraña cadena, *query string*, en la *URL*.
- Existe una limitación en la longitud de la *query string* de la petición *GET*, mientras que la petición *POST* carece de esa limitación.

En la práctica es habitual utilizar el mismo método para gestionar tanto las peticiones *GET* y *POST*. Vamos a aplicar esta idea a nuestro ejemplo para que el *servlet* sea capaz de atender tanto las peticiones *GET* como las *POST*.

6. Añade una nueva página llamada página **ConsultaLibrosPost.html** con el siguiente código:

```
<html>
<head>
  <title>TiendaLibros</title>
</head>
<body>
  <h2>Tienda Libros</h2>
  <form method="post" action="consulta">
```

```
    <b>Elige un autor:</b>
    <input type="checkbox" name="autor" value="Alvaro Garcia">Alvaro Garcia
    <input type="checkbox" name="autor" value="Aleksa Vukotic">Aleksa Vukotic
    <input type="checkbox" name="autor" value="Giulio Zambon">Giulio Zambon
    <input type="submit" value="Buscar">
  </form>
</body>
</html>
```

Observa cómo hemos cambiado el método por el cual se enviará la petición *method="post"*. También hemos cambiado la forma de indicar qué recurso será el destinatario de la acción. Ahora indicamos *action="consulta"*, sin indicar su ruta absoluta `http://localhost:8080/AccesoDatos/consulta`, como hicimos en **ConsultaLibros.html**.

7. Respecto al *servlet* que procesará la petición *POST* no es necesario escribir uno nuevo, simplemente invocaremos al método *doGet* desde el método *doPost* que tratará la petición *POST*. Añade el siguiente código al *servlet ConsultaServlet*:

```
public void doPost (HttpServletRequest request, HttpServletResponse response)
                    throws ServletException, IOException {
   doGet(request, response);  // Redirecciona la petición POST al método doGet()
}
```

8. Observa cómo al utilizar la página *ConsultaLibrosPost.html* no aparecen los parámetros en la *URL*, Figura 7.47:

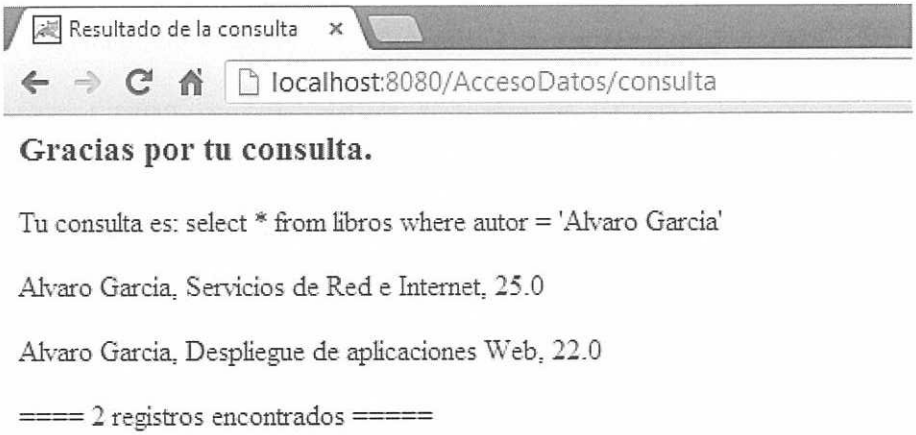


Figura 7.47: *Funcionamiento de la petición POST*

7.6. Sesiones con *Servlets*

En esta práctica utilizaremos *Servlets* para gestionar sesiones en nuestra aplicación.

HTTP es un protocolo sin estado y no orientado a conexión, esto quiere decir que cada petición no conoce qué se ha hecho en las peticiones anteriores. Sin embargo la mayoría de las aplicaciones comerciales necesitan:

- Sesión, es decir, el servidor debe ser capaz de diferenciar distintas solicitudes de un mismo cliente.
- Estado, es decir, el servidor debe ser capaz de almacenar/recordar información de anteriores solicitudes de un cliente.

Existen tres formas de gestionar las sesiones:

- *Cookies*. Pequeño fichero de texto almacenado en el cliente que se envia al servidor en cada petición. Se puede colocar la información de la sesión en una *cookie* de sesión. El mayor problema se produce porque los clientes pueden deshabilitar las *cookies*.
- Reescritura de *URLs*. Se añade un texto al final de cada *URL* con la información de la sesión, por ejemplo `http://localhost:8080/Sesiones/inicio.html;jsessionid=131908643`. Es necesario reescribir todas las *URLs* para incluir esta información.
- Campos ocultos en un formulario. Se pasa información de la sesión en campos ocultos del formulario. Hay que incluir esta información en todas las páginas.

Programar la gestión de la sesión mediante estos mecanismos puede resultar tedioso y complejo. Para mejorar esta situación el *API Servlet* proporciona un mecanismo de gestión de sesiones mediante la interfaz *HttpSession* que permite:

- Gestionar la sesión.
- Visualizar información de la sesión como su identificador o fecha de creación.
- Enlazar objetos a la sesión permitiendo que la información del usuario se mantenga a lo largo de múltiples peticiones.

La especificación del *API Servlet* realiza el registro de sesión utilizando el mecanismo de *cookies*, debido a que el servidor configura automáticamente una *cookie* "*jsessionid*" para el control de la sesión. Si el navegador del cliente no admite *cookies* automáticamente se emplea reescritura de *URLs*.

1. Para realizar un ejemplo de sesiones inicia sesión en el equipo **DesarrolloW7XX**.
2. Vamos a probar, crear y recuperar objetos de sesión con los métodos *session.setAttribute* y *session.getAttribute*. También veremos cómo obtener otra informaicón de la sesión como su identificador o su tiempo máximo permitido de inactividad.
3. Crea un nuevo *Dynamic Web Proyect* llamado **Sesiones** y añade un *Servlet* llamado **SessionServlet.java** con el siguiente código:

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
```

```
import java.util.Date;

public class SessionServlet extends HttpServlet {
 @Override
 public void doGet(HttpServletRequest request, HttpServletResponse response)
            throws IOException, ServletException {
   // Establece el tipo MIME del mensaje de respuesta
   response.setContentType("text/html");
   // Crea un flujo de salida para escribir la respuesta a la petición del cliente
   PrintWriter out = response.getWriter();

   // Recoge la sesión actual si existe, en otro caso crea una nueva
   HttpSession session = request.getSession();
   Integer contadorAccesos;
   synchronized(session) {
      contadorAccesos = (Integer)session.getAttribute("contadorAccesos");
      if (contadorAccesos == null) {
         contadorAccesos = 0;
      } else {
         contadorAccesos = new Integer(contadorAccesos + 1);
      }
      session.setAttribute("contadorAccesos", contadorAccesos);
   }

   // Escribe el mensaje de respuesta en una página html
   try {
     out.println("<!DOCTYPE html>");
     out.println("<html>");
     out.println("<meta http-equiv='Content-Type' content='text/html; charset=UTF-8'>");
     out.println("<title>Servlet de prueba de sesión</title></head><body>");
     out.println("<h2>Accesos: " +  contadorAccesos + " en esta sesión.</h2>");
     out.println("<p>(Identificador de sesión: " + session.getId() + ")</p>");
     out.println("<p>(Fecha de creación de la sesión: " +
            new Date(session.getCreationTime()) + ")</p>");
     out.println("<p>(Fecha de último acceso a la sesión " +
            new Date(session.getLastAccessedTime()) + ")</p>");
     out.println("<p>(Máximo tiempo inactivo de la sesión: " +
            session.getMaxInactiveInterval() + " seconds)</p>");
     out.println("<p><a href='" + request.getRequestURI() +  "'>Refrescar</a>");
     out.println("<p><a href='" + response.encodeURL(request.getRequestURI()) + "'>
      Refrescar con reescritura de URLs</a>");
     out.println("</body></html>");
   } finally {
      out.close();  // Cerrar el flujo de salida
   }
  }
}
```

4. Añade ahora la siguiente información al descriptor de despliegue:

```
<servlet>
   <servlet-name>ServletPruebaSesion</servlet-name>
   <servlet-class>SessionServlet</servlet-class>
</servlet>

<servlet-mapping>
   <servlet-name>ServletPruebaSesion</servlet-name>
   <url-pattern>/pruebasesion</url-pattern>
</servlet-mapping>
```

5. Exporta el proyecto **Sesiones** a un fichero **Sesiones.war**, marcando la opción sobreescribir si el fichero ya existe.

6. Inicia ahora el servidor *Tomcat*, ejecutando el *script* de arranque *startup*, situado en **CATALINA_HOME\bin**

7. Para desplegar el proyecto en *Tomcat*, copia el fichero **Sesiones.war** en la carpeta **CATALINA_HOME\webapps**.

8. Accede a la aplicación con la siguiente *URL* `http://localhost:8080/Sesiones/pruebasesion`, Figura 7.48.

Figura 7.48: Sesiones

9. Realiza las siguientes pruebas:

   - Actualiza la página pulsado **F5** o el enlace **Refrescar**.
   - Abre otra pestaña en el navegador y comprueba cómo se conserva la sesión.
   - Cierra el navegador y vuelve a abrirlo y comprueba cómo se ha perdido la sesión.

10. A continuación accede utilizando *Firefox* a nuestra aplicación `http://localhost:8080/Sesiones/pruebasesion`.

11. Sobre la página web, pulsando con el botón derecho del ratón en **Ver información de la página**, en la pestaña **Seguridad**, **Ver cookies**, puedes observar la *cookie* que ha enviado nuestra aplicación para gestionar la sesión, Figura 7.49.

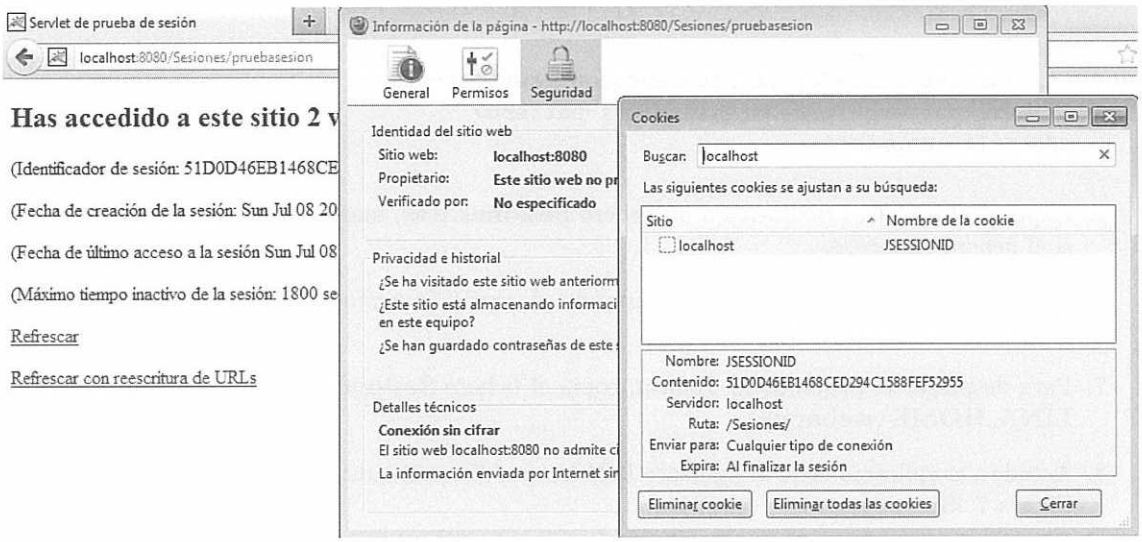


Figura 7.49: *Cookie* de sesión

12. Si observas la *cookie* puedes ver cómo su nombre es *JSESSIONID* y su contenido es el identificador de la sesión que también puedes ver en la página web.

13. Por defecto el método que utiliza *API Servlet* para gestionar la sesión son las *cookies*, pero si estas estuvieran deshabilitadas en el navegador del cliente, automáticamente utilizaría reescritura de *URLs*.

14. Cuando se utiliza este método de gestión de sesiones, todas las *URLs* reescritas emitidas por el servidor deben pasar a través del método *response.encodeURL( )* para añadir la información de la sesión.

15. Para comprobar cómo funciona la gestión de sesión utilizando reescritura de *URLs* deshabilita las *Cookies* en *Firefox*, Figura 7.50.

16. Ahora accede a la aplicación `http://localhost:8080/Sesiones/pruebasesion` y comprueba cómo no funciona el seguimiento de la sesión actualizando la página o pulsando en el enlace **Refrescar**. Solo utilizando el enlace **Refrescar con reescritura de URLs** funcionará la gestión de la sesión sin utilizar *Cookies*, Figura 7.51.

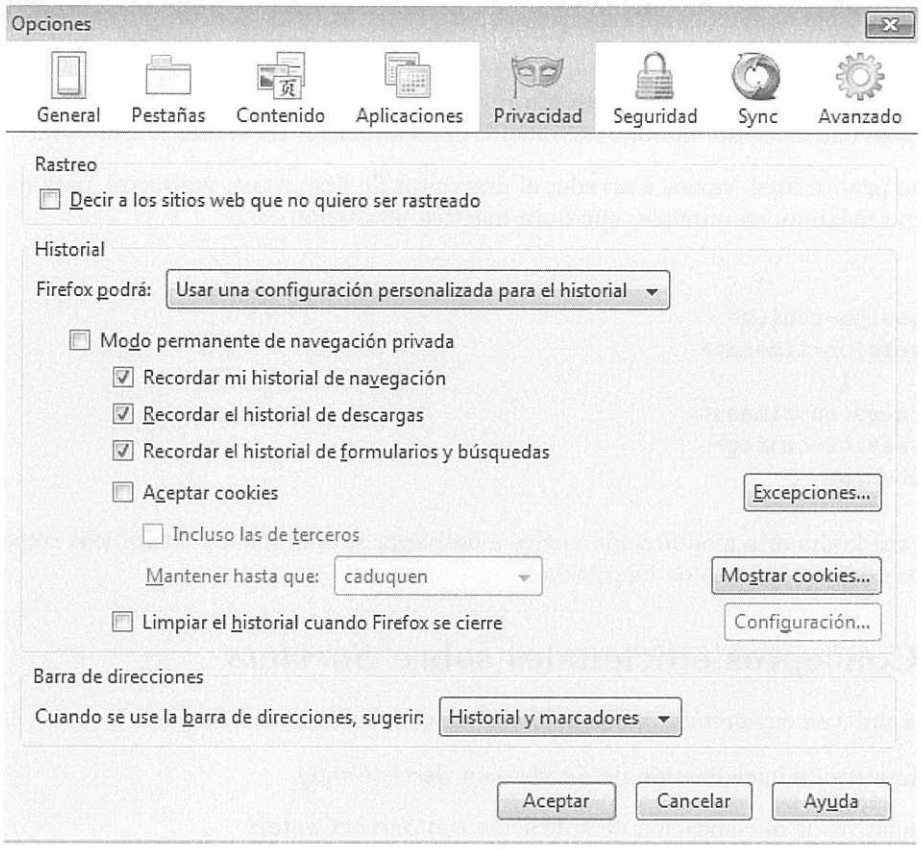


Figura 7.50: Deshabilita *Cookies*

Figura 7.51: Sesión con reescritura de URLs

17. Observa cómo el identificador de la sesión escrito en la URL es el mismo que el mostrado en la página.

18. Recuerda que si utilizamos el método *response.encodeURL( )* y las *cookies* no están deshabilitadas entonces este método no añadirá el identificador de la sesión a la *URL*.

19. Como prueba final, vamos a acceder al descriptor de despliegue, **web.xml**, para modificar el tiempo máximo, en minutos, que dura inactiva una sesión.

```
...
 <session-config>
  <session-timeout>
        1
  </session-timeout>
 </session-config>
</web-app>
```

20. Una vez hecha esta modificación vuelve a desplegar la aplicación y comprueba cómo la sesión caduca tras un minuto de inactividad.

7.7. Conceptos adicionales sobre *Servlets*

En esta práctica seguiremos analizando conceptos relativos a los *Servlets*:

- Parámetros de inicialización de *Servlet* con *ServletConfig*.
- Parámetros de inicialización de aplicación con *ServletContext*.
- Parámetros de inicialización de servidor.
- Filtros.

1. **Parámetros de inicialización de *Servlet* con *ServletConfig*.**

La clase *ServletConfig* permite al contenedor de *Servlets*, en nuestro caso *Tomcat*, pasar información al *Servlet* durante su inicialización. Los parámetros de inicialización son declarados en el descriptor de despliegue **web.xml** y son recuperados con el método:

```
ServletConfig.getInitParam("nombreParametro")
```

Para probar este concepto crea un nuevo *Dynamic Web Project* llamándolo **Filtros**. Copia y pega los recursos del proyecto **AccesoDatos**: el *Servlet ConsultaServlet.java*, la página **ConsultaLibros.html** y el descriptor de despliegue **web.xml**

Modifica el fichero **ConsultaLibros.html**, que será el punto de entrada a la aplicación, cambiando la acción del método *get*.

```
<form method="get" action="http://localhost:8080/Filtros/consulta">
```

Modifica el descriptor de despliegue:

```
<display-name>Filtros</display-name>
```

Ahora vamos a modificar el proyecto para probar los parámetros de inicialización del *Servlet*.

1.1. Primero edita el descriptor de despliegue para añadir al servlet ya existente, **ConsultaServlet**, tres parámetros de inicialización:

- La URL de la base de datos.
- El usuario con el que nos vamos a conectar.
- La contraseña de dicho usuario.

```
<servlet>
    <servlet-name>ConsultaUsuario</servlet-name>
    <servlet-class>ConsultaServlet</servlet-class>
    <init-param>
        <param-name>URLBaseDeDatos</param-name>
        <param-value>jdbc:mysql://localhost/TiendaLibros</param-value>
    </init-param>
    <init-param>
        <param-name>usuario</param-name>
        <param-value>root</param-value>
    </init-param>
    <init-param>
        <param-name>password</param-name>
        <param-value>despliegue</param-value>
    </init-param>
  </servlet>
```

1.2. Ahora modifica el código del *Servlet* para recoger los parámetros en la inicialización del *Servlet*, método *init*.

```
import java.io.*;
  import java.sql.*;
  import javax.servlet.*;
  import javax.servlet.http.*;

  public class ConsultaServlet extends HttpServlet {
  // El método doGet() se ejecuta una vez por cada petición HTTP GET.
   private String userName;
   private String password;
   private String url;

   @Override
   public void doGet(HttpServletRequest request, HttpServletResponse response)
   throws ServletException, IOException {
   // Establecemos el tipo MIME del mensaje de respuesta
   response.setContentType("text/html");
   // Creamos un objeto para poder escribir la respuesta
   PrintWriter out = response.getWriter();
   Connection conn = null;
   Statement stmt = null;
   try {
   //Paso 1: Cargar el driver JDBC.
```

```
  Class.forName("com.mysql.jdbc.Driver").newInstance();
  // Paso 2: Conectarse a la Base de Datos utilizando la clase Connection

  //URL de la base de datos(equipo, puerto, base de datos)

  conn = DriverManager.getConnection(url, userName, password);
  // Paso 3: Crear sentencias SQL, utilizando objetos de tipo Statement
  stmt = conn.createStatement();
  // Paso 4: Ejecutar las sentencias SQL a través de los objetos Statement
  String sqlStr = "select * from libros where autor = "
  + "'" + request.getParameter("autor") + "'";
  // Generar una página HTML como resultado de la consulta
  out.println("<html><head><title>Resultado de la consulta</title></head><body>");
  out.println("<h3>Gracias por tu consulta.</h3>");
  out.println("<p>Tu consulta es: " + sqlStr + "</p>");
  ResultSet rset = stmt.executeQuery(sqlStr);

     // Paso 5: Procesar el conjunto de registros resultante utilizando ResultSet
  int count = 0;
  while(rset.next()) {
   out.println("<p>" + rset.getString("autor")
   + ", " + rset.getString("titulo")
   + ", " + rset.getDouble("precio") + "</p>");
   count++;
  }
  out.println("<p>==== " + count + " registros encontrados =====</p>");
  out.println("</body></html>");
  } catch (Exception ex) {
  ex.printStackTrace();
  } finally {
  out.close(); // Cerramos el flujo de escritura
  try {

  // Cerramos el resto de recursos
  if (stmt != null) stmt.close();
  if (conn != null) conn.close();
  } catch (SQLException ex) {
  ex.printStackTrace();
  }
  }
  }
 @Override
 public void init(ServletConfig config) throws ServletException {
  super.init(config);
  // Lectura de los parámetros de inicialización del Servlet
  userName=config.getInitParameter("usuario");
  password=config.getInitParameter("password");
  url=config.getInitParameter("URLBaseDeDatos");
}
}
```

1.3. Comprueba cómo funciona de nuevo la aplicación accediendo a la *URL* de la misma, `http://localhost:8080/Filtros/ConsultaLibros.html`. Con la diferencia de que ahora los parámetros de configuración del *Servlet* no residen en el código, sino en el descriptor de despligue con las ventajas que ello conlleva.

2. **Parámetros de inicialización de aplicación con *ServletContext*.**

Esta clase encapsula el contexto de una aplicación web por lo que sería más correcto si su nombre fuera *ApplicationContext*. Permite la comunicación entre el contenedor, *Tomcat*, y el *Servlet* y también pasar información entre los diferentes *Servlets* y *JSPs* de la misma aplicación mediante los métodos: *setAttribute("nombre", objeto)* y *getAttribute("nombre")*. Vamos a realizar un cambio en el *Servlet* anterior para que los parámetros de inicialización se conviertan en atributos del contexto disponibles para cualquier *Servlet* o *JSP* de la aplicación.

2.1. Continuando con el ejemplo, modifica el método *init* del *Servlet ConsultaServlet* con el siguiente código:

```
@Override
public void init(ServletConfig config) throws ServletException {
  super.init(config);
  // Se leen los parámetros de inicialización de Servlet y
  // se convierten en atributos del contexto para compartirlos con
  // cualquier servlet y JSP de la aplicación
  ServletContext context = config.getServletContext();
  context.setAttribute("URLBaseDeDatos", config.getInitParameter("URLBaseDeDatos"));
  context.setAttribute("usuario", config.getInitParameter("usuario"));
  context.setAttribute("password", config.getInitParameter("password"));
  // Se recuperan las variables de contexto de la aplicación
  userName=(String)context.getAttribute("usuario");
  password=(String)context.getAttribute("password");
  url=(String)context.getAttribute("URLBaseDeDatos");
}
```

2.2. Prueba de nuevo la aplicación para comprobar su funcionamiento. Con este cambio los parámetros de inicialización ahora se han convertido en variables de contexto utilizables por cualquier *Servlet* o *JSP* de la aplicación.

2.3. También es posible definir parámetros de inicializaicón a nivel de contexto en el descriptor de despliegue y recuperarlos desde cualquier *Servlet* o *JSP* de la aplicación.

2.4. Modifica el descriptor de despliegue añadiendo los siguientes parámetros de contexto. Observa cómo no están dentro de ningún *Servlet* como ocurría con los parámetros de inicialización del *Servlet*.

```
...
<context-param>
    <param-name>URLBaseDeDatos</param-name>
    <param-value>jdbc:mysql://localhost/TiendaLibros</param-value>
</context-param>
<context-param>
    <param-name>usuario</param-name>
    <param-value>root</param-value>
```

```
</context-param>
<context-param>
    <param-name>password</param-name>
    <param-value>despliegue</param-value>
</context-param>
</web-app>
```

2.5. Modifica el método *init* del *Servlet ConsultaServlet* con el siguiente código:

```
@Override
public void init(ServletConfig config) throws ServletException {
  super.init(config);
  // Se leen los parámetros de inicialización del contexto
  // Estos parámetros podrán ser utilizados por otros Servlet o JSP de la aplicación
  ServletContext context = config.getServletContext();
  url=context.getInitParameter("URLBaseDeDatos");
  userName=context.getInitParameter("usuario");
  password= context.getInitParameter("password");
}
```

2.6. Comprueba cómo el proyecto sigue funcionando correctamente.

3. **Parámetros de inicialización de servidor**.

3.1. De manera similiar a como hemos declarado parámetros de inicialización en el descriptor de despliegue de la aplicación, podemos crear parámetros de inicialización a nivel de servidor. Los definiríamos en el fichero **CATALINA_HOME\conf\web.xml** con la misma sintaxis que los parámetros de aplicación.

```
<context-param>
  <param-name>usuario</param-name>
  <param-value>root</param-value>
</context-param>
```

3.2. La diferencia radica en que estos parámetros de inicialización a nivel de servidor pueden ser recuperados por cualquier aplicación utilizando el método *getInitParameter* de *ServletContext*.

4. **Filtros**.

Un filtro es una clase que implementa la interfaz `javax.servlet.Filter`, y que intercepta las peticiones a recursos del contenedor Web, *Tomcat*, previo al procesamiento de los mismos por parte del contenedor.

Un filtro "envuelve" los componentes del servidor con funcionalidad adicional, interceptando las peticiones del cliente, lo que nos permite una gran versatilidad a la hora de gestionarlas.

4.1. Vamos a añadir un filtro que registre los accesos a nuestra aplicación para ilustrar su uso.

4.2. En nuestro proyecto **Filtros**, en la carpeta **Java Resources\src** pulsa sobre el paquete por defecto y añade un nuevo filtro, llamado **FiltroContador**, Figura 7.52.

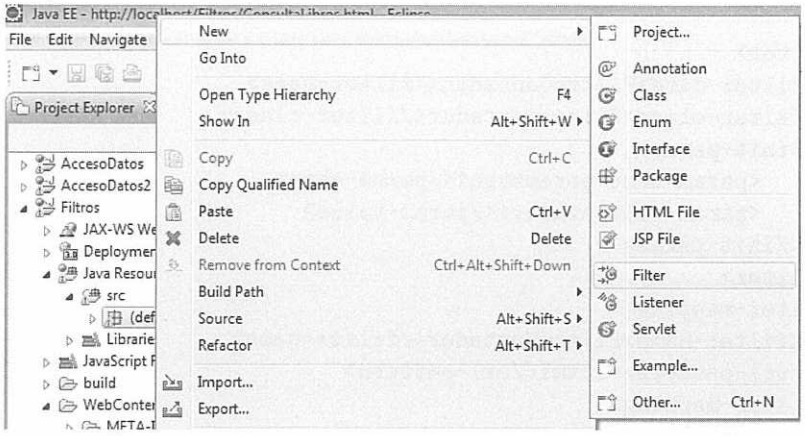


Figura 7.52: Añadir filtro

4.3. El código del filtro será el siguiente:

```
import java.io.IOException;
import javax.servlet.*;
public class FiltroContador implements Filter {

 // Para tener una referencia al contexto de filtro
 FilterConfig config;

 public void init(FilterConfig config) { this.config = config; }

 public void doFilter(ServletRequest request, ServletResponse response,
        FilterChain chain) throws IOException, ServletException {
 // Tomamos el Contexto de sesión donde almacenaremos el contador
 ServletContext miContexto = config.getServletContext();
 // Vemos si el atributo contador existe dentro del contexto
 // En caso contrario lo creamos e inicializamos a cero
 Integer intContador = (Integer)miContexto.getAttribute("contador");
 if(intContador == null) {
   intContador = new Integer(0);
 }
 intContador = new Integer(intContador.intValue() + 1);
 miContexto.setAttribute("contador", intContador);
 // Instruimos al contenedor para que invoque al siguiente filtro
 chain.doFilter(request, response);
 }
 public void destroy() {}
}
```

4.4. Lo que hace este filtro es crear una variable de aplicación **contador** donde almacena el número de peticiones a ficheros ***.html** de nuestra aplicación.

4.5. A continuación hay que declararlo en el descriptor de despliegue de nuestra aplicación añadiendo la siguiente información.

```
...
<filter>
  <filter-name>FiltroContador</filter-name>
  <filter-class>FiltroContador</filter-class>
   <init-param>
      <param-name>parametro1</param-name>
      <param-value>valor1</param-value>
   </init-param>
</filter>
<filter-mapping>
   <filter-name>FiltroContador</filter-name>
   <url-pattern>*.html</url-pattern>
</filter-mapping>
</web-app>
```

4.6. Observa diferentes aspectos sobre esta declaración:

- Fijate cómo es parecida a la declaración de un *Servlet*, primero declaramos el Filtro y luego declaramos qué peticiones va a interceptar.
- Recuerda que en *filter-class* deberás poner la ruta completa de la clase si estás utilizando paquetes.
- Observa cómo también se pueden crear parámetros de inicialización del filtro.
- Luego, hemos decidido que nuestro filtro intercepte todas las peticiones a archivos ***.html**. Mira estos dos ejemplos:

  ```
  <filter-mapping>
       <filter-name>unFiltro</filter-name>
       <url-pattern>/*<url-pattern>
    </filter-mapping>
  ```

  Interceptaría cualquier petición a nuestra aplicación, ya sea dinámica o estática.

  ```
  <filter-mapping>
     <filter-name>unFiltro</filter-name>
     <servlet-name>unServlet</servlet-name>
  </filter-mapping>
  ```

  Interceptaría las peticiones realizadas al *Servlet* **unServlet**.
- Hay que indicar que si declaramos varios filtros, el orden en el que se declaren en el descriptor de despliegue determina el orden en el que se ejecutan.

4.7. Para finalizar modifica el *Servlet* **ConsultaServlet** para mostrar el contador junto con el resultado de la consulta.

```
import java.io.*;
import java.sql.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class ConsultaServlet extends HttpServlet {
// El método doGet() se ejecuta una vez por cada petición HTTP GET.

 private String userName;
 private String password;
 private String url;
```

```
private Integer contador;
private ServletContext context;

@Override
public void doGet(HttpServletRequest request, HttpServletResponse response)
 throws ServletException, IOException {
// Establecemos el tipo MIME del mensaje de respuesta
response.setContentType("text/html");
// Creamos un objeto para poder escribir la respuesta
PrintWriter out = response.getWriter();
Connection conn = null;
Statement stmt = null;
try {
 //Paso 1: Cargar el driver JDBC.
 Class.forName("com.mysql.jdbc.Driver").newInstance();
 // Paso 2: Conectarse a la Base de Datos utilizando la clase Connection

 //URL de la base de datos(equipo, puerto, base de datos)

 conn = DriverManager.getConnection(url, userName, password);
 // Paso 3: Crear sentencias SQL, utilizando objetos de tipo Statement
 stmt = conn.createStatement();
 // Paso 4: Ejecutar las sentencias SQL a través de los objetos Statement
 String sqlStr = "select * from libros where autor = "
 + "'" + request.getParameter("autor") + "'";
 // Generar una página HTML como resultado de la consulta
 out.println("<html><head><title>Resultado de la consulta</title></head><body>");
 out.println("<h3>Gracias por tu consulta.</h3>");
 out.println("<p>Tu consulta es: " + sqlStr + "</p>");
 ResultSet rset = stmt.executeQuery(sqlStr);
 // Paso 5: Procesar el conjunto de registros resultante utilizando ResultSet
 int count = 0;
 while(rset.next()) {
  out.println("<p>" + rset.getString("autor")
  + ", " + rset.getString("titulo")
  + ", " + rset.getDouble("precio") + "</p>");
  count++;
 }
 out.println("<p>==== " + count + " registros encontrados =====</p>");
 // Se recupera la variable contador del ServletContext
 contador= (Integer)context.getAttribute("contador");
 out.println("<p>==== " + contador.intValue() + " peticiones *.html===</p>");
 out.println("</body></html>");
} catch (Exception ex) {
ex.printStackTrace();
} finally {
 out.close(); // Cerramos el flujo de escritura
try {

// Cerramos el resto de recursos
```

```
    if (stmt != null) stmt.close();
    if (conn != null) conn.close();
   } catch (SQLException ex) {
    ex.printStackTrace();
   }
  }
  }

  @Override
  public void init(ServletConfig config) throws ServletException {
    super.init(config);
    // Se leen los parametros de inicialización de Servlet y
    // se convierten en atributos del contexto para compartirlos con
    // cualquier servlet y JSP de la aplicación
    // Se guarda el ServletContext que representa el contexto
    // de la aplicación para usarlo en el doGet
    context = config.getServletContext();
    url=context.getInitParameter("URLBaseDeDatos");
    userName=context.getInitParameter("usuario");
    password= context.getInitParameter("password");
   }
  }
```

4.8. Finalmente ejecuta el proyecto y comprueba cómo el filtro contabiliza las peticiones. Puedes ir actualizando la página **ConsultaLibros.html**, antes de pulsar en **Buscar**, para poder comprobarlo, Figura 7.11.

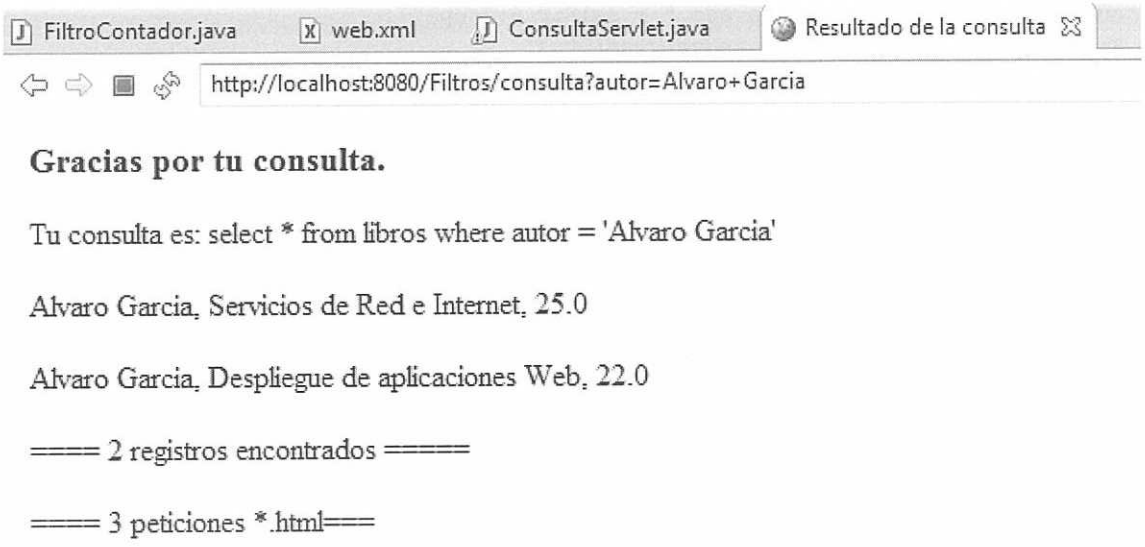


Figura 7.53: Prueba filtro

7.8. *Pool de conexiones con JDNI y Servlets 3.0*

En las aplicaciones con acceso a datos, crear una nueva conexión para cada petición es algo totalmente ineficiente debido a que el coste que conlleva crear la conexión e inicializarla suele ser mayor que el de la operación que queremos realizar.

La opción de mantener una conexión abierta compartida puede acarrear problemas de concurrencia.

Una solución habitual a estos problemas consiste en mantener un grupo, *pool*, de conexiones abiertas que funciona de la siguiente manera: cuando necesitamos una conexión la solicitamos al *pool*, este busca una que esté libre y nos la da, apuntando que la tenemos nosotros y que deja de estar libre. Tras realizar la operación cerramos la conexión. El *pool* recibe esta petición de cierre pero no cierra la conexión, sino que la deja abierta y la vuelve a marcar como libre para la siguiente petición. Esta reutilización de conexiones es más eficiente que andar abriendo nuevas conexiones para cada petición.

Tomcat 7.0 permite utilizar un *pool* de conexiones utilizando *JDNI* (*Java Directory and Naming Interface*). En el siguiente apartado vamos a desarrollar un ejemplo para comprobar cómo funcionaría.

1. <u>**Pool de conexiones con JDNI**</u>.

   1.1. Crea un nuevo *Dynamic Web Project*, en *Eclipse* llamado **PoolConexiones**. Esta aplicación pedirá un usuario y contraseña y comprobará si existen en la base de datos, en tal caso creará una variable de sesión para almacenar el usuario.

   1.2. Ejecuta el siguiente *script* en la base de datos **TiendaLibros**, con él crearemos una tabla de usuarios y contraseñas.

```
use TiendaLibros;

drop table if exists usuarios;

create table usuarios(
usuario char(16) not null,
password char(41) not null,
primary key (usuario));

insert into usuarios values ('usuario1', password('usuario1')),
 ('administrador1', password('administrador1'));


select * from usuarios;
```

   1.3. El siguiente paso consistirá en definir el recurso *JNDI DataSource* en *Tomcat* en el fichero de contexto **context.xml** de la instancia del servidor de **tomcat** definida en **eclipse**. Este fichero permite definir configuraciones válidas para todos los contextos de *Tomcat*, véase Figura 7.54:

Figura 7.54: *Pool* de conexiones

```
<Resource name="jdbc/mysql_tiendalibros" auth="Container" type="javax.sql.DataSource"
    maxActive="100" maxIdle="30" maxWait="10000" removeAbandoned="true"
    username="root" password="despliegue" driverClassName="com.mysql.jdbc.Driver"
    url="jdbc:mysql://localhost/TiendaLibros" />
```

1.4. A la hora de definir el recurso se pueden establecer diferentes parámetros:

- *Resource name*: Establece el nombre del recurso disponible para las aplicaciones.
- *type*: Establece el tipo del recurso disponible para las aplicaciones.
- *url*: La cadena de conexión de la base de datos.
- *driverClassName*: El nombre de la clase del *driver JDBC*.
- *username*: El nombre de usuario para acceder a la base de datos.
- *password*: La contraseña del usuario para acceder a la base de datos.
- *maxActive*: El número máximo de conexiones en el pool de conexiones.
- *maxIdle*: El número máximo de conexiones inactivas a retener en el *pool* de conexiones.
- *maxWait*: El tiempo máximo a esperar para obtener una conexión, en milisegundos.
- *removeAbandoned*: Si *removeAbandoned* = *true* entonces cuando haya pocas conexiones disponibles en el *pool* de conexiones se recuperará y reciclará cualquier conexión abandonada que se encuentre.
- etc.

1.5. A continuación añade una página **index.html**, a la carpeta **WebContent**, que será el punto de entrada a la aplicación:

```
<!DOCTYPE html>
<html>
<head>
  <title>Login</title>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
</head>
<body>
  <h2>Login</h2>
  <form method="get" action="login">
  <table>
    <tr>
      <td>Introduce tu usuario:</td>
      <td><input type='text' name='usuario' /></td>
    </tr>
    <tr>
      <td>Introduce tu contraseña:</td>
```

```
      <td><input type='password' name='password' /></td>
    </tr>
  </table>
  <br />
  <input type="submit" value='LOGIN' />
  <input type="reset" value='LIMPIAR' />
  </form>
</body>
</html>
```

1.6. Ahora añade el *servlet* **LoginServlet.java** con el siguiente código:

```
import java.io.*;
import java.util.*;
import java.util.logging.*;
import javax.naming.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.sql.*;
import javax.sql.*;

public class LoginServlet extends HttpServlet {

   private DataSource pool;  // Pool de conexiones a la base de datos

   @Override
   public void init(ServletConfig config) throws ServletException {
      try {
         // Crea un contexto para poder luego buscar el recurso DataSource
         InitialContext ctx = new InitialContext();
         // Busca el recurso DataSource en el contexto
         pool = (DataSource)ctx.lookup("java:comp/env/jdbc/mysql_tiendalibros");
         if (pool == null)
            throw new ServletException("DataSource desconocida 'mysql_tiendalibros'");
      } catch (NamingException ex) {
         Logger.getLogger(LoginServlet.class.getName()).log(Level.SEVERE, null, ex);
      }
   }



   @Override
   protected void doGet(HttpServletRequest request, HttpServletResponse response)
           throws ServletException, IOException {
      response.setContentType("text/html;charset=UTF-8");
      PrintWriter out = response.getWriter();

      Connection conn = null;
      Statement  stmt = null;
      try {
         out.println("<html><head><title>Login</title></head><body>");
```

```
   out.println("<h2>Login</h2>");

   conn = pool.getConnection();  // Obtiene un conexión del pool
   stmt = conn.createStatement();

   // recupera los parámetros de la petición request usuario y password
   String usuario = request.getParameter("usuario");
   String password = request.getParameter("password");
   boolean noUsuario = usuario != null && ((usuario=usuario.trim()).length()>0);
   boolean noPwd = password != null && ((password=password.trim()).length()>0);

   // Valida los parámetros de la petición request
   if (!noUsuario) {
      out.println("<h3>Debes introducir tu usuario</h3>");
   } else if (!noPwd) {
      out.println("<h3>Debes introducir tu password</h3>");
   } else {
      // Verifica que existe algún usuario con ese login y password
      StringBuilder sqlStr = new StringBuilder();
      sqlStr.append("SELECT * FROM usuarios WHERE ");
      sqlStr.append("STRCMP(usuarios.usuario, '")
            .append(usuario).append("') = 0 ");
      sqlStr.append("AND STRCMP(usuarios.password, PASSWORD('")
            .append(password).append("')) = 0 ");

      ResultSet rset = stmt.executeQuery(sqlStr.toString());

      if (!rset.next()) {  // si no está vacío el resulset
         out.println("<h3>Nombre o contraseña incorrecta</h3>");
         out.println("<p><a href='index.html'>Vuelve a la página login</a></p>");
      } else {

         // Crea una nueva sesión y guarda el usuario como variable de sesión
         // Primero, invalida la sesión si existe
         HttpSession session = request.getSession(false);
         if (session != null) {
            session.invalidate();
         }
         session = request.getSession(true);
         synchronized (session) {
            session.setAttribute("usuario", usuario);
          }

         out.println("<p>Hola, " + usuario + "!</p>");
         out.println("<p><a href='hazalgo'>Haz algo</a></p>");
      }
   }
   out.println("</body></html>");

} catch (SQLException ex) {
```

```
            out.println("<h3>Servicio no disponible</h3></body></html>");
            Logger.getLogger(LoginServlet.class.getName()).log(Level.SEVERE, null, ex);
        } finally {
            out.close();
            try {
                if (stmt != null) stmt.close();
                if (conn != null) conn.close();  // Devuelve la conexión al pool
            } catch (SQLException ex) {
                Logger.getLogger(LoginServlet.class.getName()).log(Level.SEVERE, null, ex);
            }
        }
    }

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        doGet(request, response);
    }
}
```

1.7. Observa en el código cómo se solicita una conexión al *pool* de conexiones y cómo se devuelve al pool cuando se cierra.

1.8. Fijate cómo se crea una instancia de *InitialContext* que permitirá buscar el recurso *DataSource* `jdbc/mysql_tiendalibros`. Los recursos definidos están ubicados en la sección `java:comp/env` del directorio de nombres *JDNI*.

```
InitialContext ctx = new InitialContext();
pool = (DataSource)ctx.lookup("java:comp/env/jdbc/mysql_tiendalibros");
```

1.9. Crea el *servlet* **HazAlgo.java**, con el siguiente código:

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.util.*;

public class HazAlgo extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();

        try {
            out.println("<html><head><title>Haz algo</title></head><body>");
            out.println("<h2>Haz algo...</h2>");

            // Recupera el nombre de usuario
            String usuario;
```

```
            HttpSession session = request.getSession(false);
            if (session == null) {
               out.println("<h3>No has iniciado sesión</h3>");
            } else {
               synchronized (session) {
                  usuario = (String) session.getAttribute("usuario");
               }

               out.println("<table>");
               out.println("<tr>");
               out.println("<td>Usuario:</td>");
               out.println("<td>" + usuario + "</td></tr>");
               out.println("<tr>");
               out.println("</table>");

               out.println("<p><a href='logout'>Logout</a></p>");
            }
            out.println("</body></html>");
         } finally {
            out.close();
         }
      }
   }
```

1.10. A continuación añade el *servlet* **LogoutServlet.java** con el siguiente código:

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class LogoutServlet extends HttpServlet {
   @Override
   protected void doGet(HttpServletRequest request, HttpServletResponse response)
           throws ServletException, IOException {
      response.setContentType("text/html;charset=UTF-8");
      PrintWriter out = response.getWriter();

      try {
         out.println("<html><head><title>Logout</title></head><body>");
         out.println("<h2>Logout</h2>");
         HttpSession session = request.getSession(false);
         if (session == null) {
            out.println("<h3>No has iniciado sesión</h3>");
         } else {
            session.invalidate();
            out.println("<p>Adios</p>");
            out.println("<p><a href='index.html'>Login</a></p>");
         }
         out.println("</body></html>");
      } finally {
         out.close();
```

```
        }
    }
}
```

1.11. Finalmente añade la siguiente información al descriptor de despliegue de la aplicación, **web.xml**:

```
<servlet>
    <servlet-name>LoginServlet</servlet-name>
    <servlet-class>LoginServlet</servlet-class>
  </servlet>
  <servlet>
    <servlet-name>HazAlgoServlet</servlet-name>
    <servlet-class>HazAlgo</servlet-class>
  </servlet>
  <servlet>
    <servlet-name>LogoutServlet</servlet-name>
    <servlet-class>LogoutServlet</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>LoginServlet</servlet-name>
    <url-pattern>/login</url-pattern>
  </servlet-mapping>
  <servlet-mapping>
    <servlet-name>HazAlgoServlet</servlet-name>
    <url-pattern>/hazalgo</url-pattern>
  </servlet-mapping>
  <servlet-mapping>
    <servlet-name>LogoutServlet</servlet-name>
    <url-pattern>/logout</url-pattern>
  </servlet-mapping>
  <session-config>
    <session-timeout>30</session-timeout>
  </session-config>
  <welcome-file-list>
    <welcome-file>index.html</welcome-file>
  </welcome-file-list>
```

1.12. Comprueba el funcionamiento de la misma, cómo valida usuario y contraseña contra la base de datos utilizando una conexión del pool, y cómo utilizan la variable de sesión los *servlets*, véase Figura 7.55:

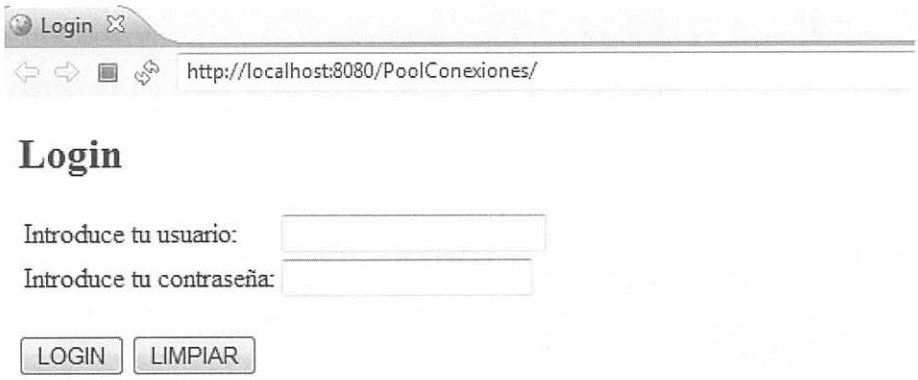


Figura 7.55: Funcionamiento aplicación

2. **Servlets 3.0**.

La especificación *Servlet 3.0* introduce la posibilidad de facilitar el despliegue de los *servlets* introduciendo anotaciones dentro del código del *servlet* en vez de en el descriptor de despliegue.

- @WebServlet: Para definir un *serlvet*.
- @WebInitParam: Para definir los parámetros de inicialización de un *servlet*.
- @WebListener: Para definir un *listener*.
- @WebFilter: Para definir un filtro.
- @MultipartConfig: Para subida múltiple de archivos.

2.1. Modifica los tres *servlets* de la aplicación para que incluyan las anotaciones relativas al despliegue. A continuación se muestran las modificaciones necesarias en el *servlet* **LoginServlet.java**:

```
...
import javax.servlet.annotation.WebServlet;

@WebServlet(
    name = "LoginServlet",
    urlPatterns = {"/login"})
public class LoginServlet extends HttpServlet {
...
```

2.2. Una vez realizadas las anotaciones en los tres *servlets* podemos eliminar la información relativa a los *servlets* que añadimos al descriptor de despliegue en su momento.

2.3. Para finalizar comprueba cómo la aplicación sigue funcionando de igual forma.

7.9. *Java Server Pages*

Las *Java Server Pages (JSP)* constituyen una tecnología complementaría a los *Servlets* que acabamos de ver. Surgieron como respuesta a las *Active Server Pages* de *Microsoft* y permiten separar los bloques de *HTML* estáticos de los dinámicos generados por código *Java* que se ejecuta en el servidor. En resumen, permiten separar el código de presentación de la lógica de programación.

Simplificando, podemos afirmar que los *Servlet* son "*HTML* dentro de *Java*" y los *JSP* son "*Java* dentro de *HTML*". Son tecnologías complementarias, en los modelos MVC (Modelo Vista Controlador) los *Servlets* se encargarán de controlar las peticiones, mientras que los *JSP* se encargarán de generar la vista de la aplicación.

Cuando un cliente solicita una página **.jsp**, se ejecuta en el servidor el código *JSP* de la página, dando como resultado una página *HTML* que se fusiona con el *HTML* original, generando una página *HTML* de respuesta que será enviada al cliente.

1. **Primera página *JSP*.**

   1.1. Crea un nuevo *Dynamic Web Project* llamado **JavaServerPages**.

   1.2. Pulsando con el botón derecho del ratón sobre la carpeta *WebContent* añade un nuevo fichero *JSP* llamado **expresiones.jsp**, Figura 7.56.

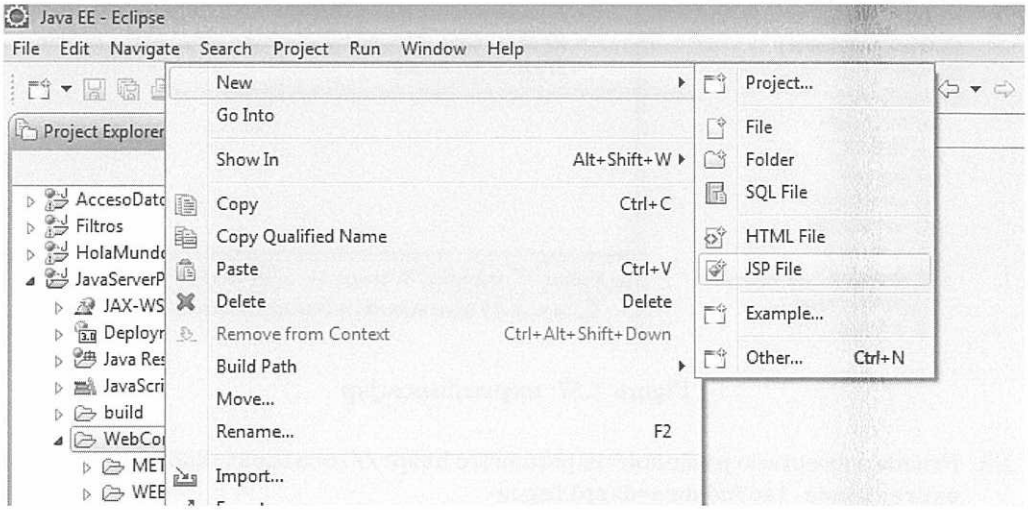


Figura 7.56: Nuevo fichero JSP

   1.3. Para probar el uso de expresiones en *JSP*, añade el siguiente código al fichero **expresiones.jsp**:

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
```

```
    <title>Expresiones JSP</title>
  </head>
<body>
  <h1>Ejemplo de expresiones JSP</H1>
  <ul>
<li>Fecha actual: <%=new java.util.Date()%>
<li>Nombre del host: <%=request.getRemoteHost()%>
<li>ID de la sesión: <%=session.getId()%>
<li>El parámetro es: <%=request.getParameter("nombre")%>
  </ul>
</body>
</html>
```

1.4. Pulsando con el botón derecho del ratón sobre el fichero **expresiones.jsp** , selecciona la acción **Run As, Run on Server**, Figura 7.57.

Figura 7.57: **expresiones.jsp**

1.5. Prueba a ejecutarlo pasándole un parámetro `http://localhost:8080/JavaServerPages/expresiones.jsp?nombre=despliegue`

2. <u>**Más ejemplos de *JSPs***</u>.

2.1. Para probar el uso de *scriptlets* añade la página **color.jsp** con el siguiente código:

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
  "http://www.w3.org/TR/html4/loose.dtd">
  <html>
    <head>
```

```
    <title>Scriptlets JSP</title>
    <%
      String bgColor=request.getParameter("bgColor");
      boolean hayColor;
      if (bgColor!=null) hayColor=true; else
      {
       hayColor=false;
       bgColor="WHITE";
      }
    %>
    </head>
    <body BGCOLOR="<%=bgColor%>">
    <h1>Ejemplo de scriptlets JSP</h1>
    <%
     if (hayColor) out.println("Se ha utilizado el color: " + bgColor);
     else out.println("Se ha utilizado el color por defecto: WHITE");
    %>
    </body>
  </html>
```

2.2. Prueba a ejecutarlo con las siguientes *URLs*: `http://localhost:8080/JavaServerPages/color.jsp` y `http://localhost:8080/JavaServerPages/color.jsp?bgColor=RED`.

2.3. Para probar el empleo de declaraciones y de la directiva *page* crea la página **contador.jsp** con el siguiente código:

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
  "http://www.w3.org/TR/html4/loose.dtd">
  <html>
    <head>
    <title>Scriptlets JSP</title>
    </head>
  <%@page import="java.util.*"%>
    <%-- Esto en un comentario de JSP --%>
    <%!
      private int cont=0;
      private Date fecha= new Date();
    %>
    <body>
    <p>Esta página ha sido accedida <b><%= ++cont%></b> veces.</p>
    <p>El último acceso ha sido con fecha <b><%=fecha%></b></p>
    <%
     fecha=new Date();
    %>
    </body>
  </html>
```

2.4. Ejecútalo para comprobar su funcionamiento.

2.5. Ahora, para probar la directiva *include* crea un fichero **incluir.jsp** con el siguiente código:

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
  <title>Ejemplo de uso de un contador incluido en un JSP</title>
  </head>
  <body>
  <h1>Ejemplo de uso de un contador incluido en un JSP</h1>
  <%@include file="contador.jsp"%>
  </body>
</html>
```

2.6. Finalmente vamos a utlizar la directiva *page* para que ante cualquier error se ejecute una página *JSP* de tratamiento de errores.

2.7. Crea una página **division.jsp** que se encargue de dividir dos enteros que recibe como parámetros:

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
  <title>Manejo de errores de JSP</title>
  </head>
  <body >
  <%@page errorPage="errores.jsp"%>
  <h1>Ejemplo de manejo de errores en JSP</h1>
  <%!
     private double toDouble(String value)
     {
     return(Double.valueOf(value).doubleValue());
     }
  %>
  <%
     double op1 = toDouble(request.getParameter("op1"));
     double op2= toDouble(request.getParameter("op2"));
     double res = op1/op2;
  %>
  <table border=1>
  <tr><th></th><th>División</th></tr>
  <tr><th>Operando 1: </th><td><%=op1%></td></tr>
  <tr><th>Operando 2: </th><td><%=op2%></td></tr>
  <tr><th>Resultado: </th><td><%=res%></td></tr>
  </table>
```

```
</body>
</html>
```

Observa cómo con la directiva *page* indicamos que ante un error se ejecute la página **errores.jsp**.

2.8. Ahora crea la página **errores.jsp** con el siguiente código:

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
  <title>Página de errores en JSP</title>
  </head>
  <body >
  <%@page isErrorPage="true"%>
  <h1>Página de errores en JSP</h1>
  <p>division.jsp ha reportado el siguiente error:
  <b><%=exception%></b>
  </p>
  <p>El error que ha ocurrido es: <pre>
  <% exception.printStackTrace(new java.io.PrintWriter(out)); %>
  </pre>
  </p>
  </body>
</html>
```

Observa cómo ahora se emplea la directiva *page* para indicar que esta es una página de errores, *isErrorPage*.

2.9. Ejecuta estas dos *URLs* `http://localhost:8080/JavaServerPages/division.jsp?op1=12&op2=6` y `http://localhost:8080/JavaServerPages/division.jsp?op1=12&op2=a` para comprobar cómo funciona si hay o no error, Figura 7.58.

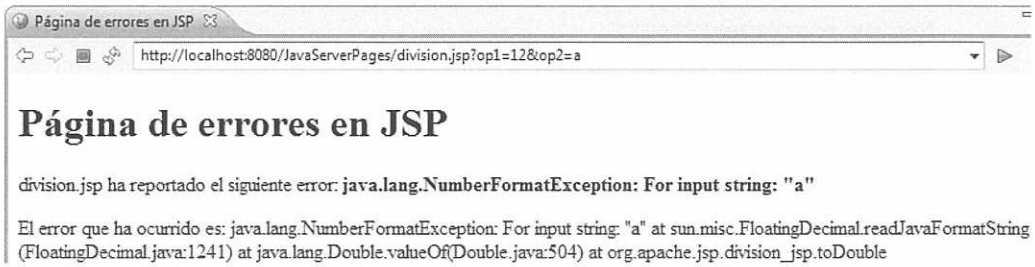


Figura 7.58: Página de errores

3. **Ejemplo de *JSP* con acceso a datos.**

3.1. Para finalizar vamos a realizar un ejemplo de *JSP* que accede a nuestra base de datos *TiendaLibros* con la misma funcionalidad que el *Servlet* que realizamos en la práctica de **Acceso a Datos con JDBC**.

3.2. Añade un fichero **consulta.jsp** a nuestro proyecto **Java Server Pages** con el siguiente código:

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
  "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
  <title>Consulta Libros</title>
</head>
<body>
  <h1 align="center">Tienda Libros</h1>
  <h3>Elige autor(s):</h3>
  <form method="get">
    <input type="checkbox" name="autor" value="Alvaro Garcia">Alvaro Garcia
    <input type="checkbox" name="autor" value="Aleksa Vukotic">Aleksa Vukotic
    <input type="checkbox" name="autor" value="Giulio Zambon">Giulio Zambon
    <input type="submit" value="consulta">
  </form>
<%
  String[] autores = request.getParameterValues("autor");
  if (autores != null) {
%>
<%@ page import = "java.sql.*" %>
<%
  //Paso 1: Cargar el driver JDBC.
  Class.forName("com.mysql.jdbc.Driver").newInstance();
  //Paso 2: Conectarse a la Base de Datos utilizando la clase Connection
  String userName="root";
  String password="despliegue";
  String url="jdbc:mysql://localhost/TiendaLibros";
  Connection conn = DriverManager.getConnection(url, userName, password);
  //Paso 3: Crear sentencias SQL, utilizando objetos de tipo Statement
  Statement stmt = conn.createStatement();
  String sqlStr = "SELECT * FROM libros WHERE ";
  for (int i = 0; i < autores.length; i++ ) {
    sqlStr = sqlStr + "autor = '" + autores[i] + "' ";
    if (i != autores.length - 1) {
       sqlStr += "OR ";
    }
  }
  sqlStr += "AND cantidad > 0 ORDER BY precio DESC";
```

```
  // para depuración
  System.out.println("La consulta sql es " + sqlStr);
  //Paso 4: Ejecutar las sentencias SQL a través de los objetos Statement
  ResultSet rset = stmt.executeQuery(sqlStr);
%>
<hr>
  <table border=1 cellpadding=5>
    <tr>
      <th>Autor</th>
      <th>Titulo</th>
      <th>Precio</th>
      <th>Cantidad</th>
    </tr>
<%
//Paso 5: Procesar el conjunto de registros resultante utilizando ResultSet
while (rset.next()) {
%>
 <tr>
  <td><%= rset.getString("autor") %></td>
  <td><%= rset.getString("titulo") %></td>
  <td><%= rset.getInt("precio") %></td>
  <td><%= rset.getInt("cantidad") %></td>
 </tr>
<%
 }
%>
 </table>
 <%
  //Cierre de recursos
  rset.close();
  stmt.close();
  conn.close();
 }
%>

</body>
</html>
```

Observa cómo el código es muy similar al que utilizamos en el *Servlet* para acceder a datos, la diferencia es que ahora el código *Java* está embebido dentro de código *HTML*.

3.3. Ejecuta la página **consulta.jsp** desde *Eclipse* y comprueba su correcto funcionamiento, Figura 7.59.

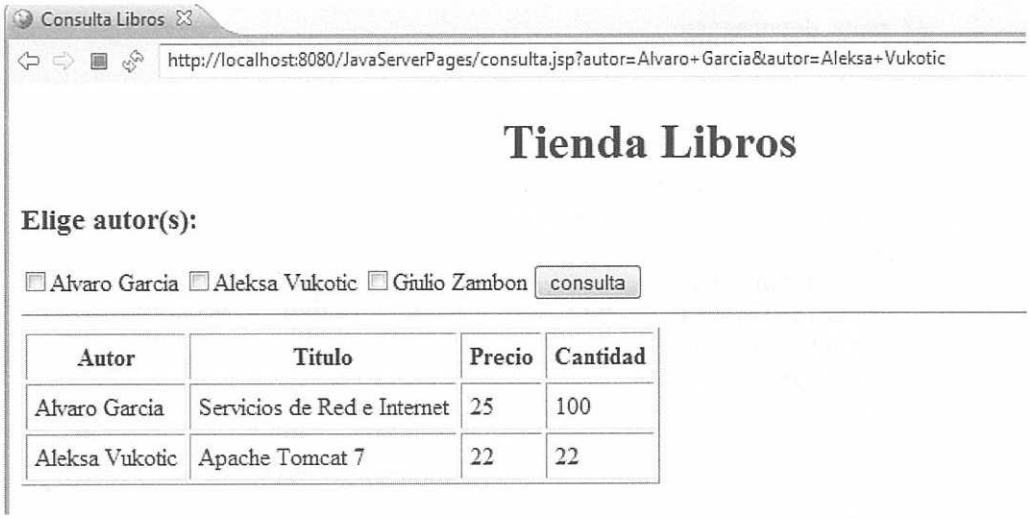


Figura 7.59: *JSP con acceso a datos*

7.10. *Arquitectura MVC*

En esta práctica abordaremos el empleo del patrón **Modelo Vista Controlador MVC**.

Como introducción, podemos hablar de los **patrones de software** como soluciones reutilizables en el diseño de una aplicación. En nuestro caso la utilización del patrón **MVC** proporciona una arquitectura uniforme que permite una fácil expansión, mantenimiento y modificación de una aplicación.

El principal objetivo de la arquitectura **MVC** es aislar tanto los datos de la aplicación como el estado (**modelo**) de la misma, del mecanismo utilizado para representar (**vista**) dicho estado, así como para modularizar esta vista y modelar la transición entre estados del modelo (**controlador**).

Las aplicaciones **MVC** se dividen en tres grandes áreas funcionales:

- **Vista**: Presentación de los datos.
- **Controlador**: Atenderá las peticiones y toma de decisiones de la aplicación.
- **Modelo**: La lógica del negocio y los datos asociados con la aplicación.

El propósito del **MVC** es aislar los cambios. Es una arquitectura preparada para los cambios, que desacopla los datos y la lógica de negocio de la lógica de presentación, permitiendo la actualización y desarrollo independiente de cada uno de los citados componentes.

Las aplicaciones de **MVC** pueden ser implementadas con *J2EE* utilizando *JSP* para las **vistas**, *servlets* como **controladores** y *JavaBeans* para el **modelo**, siendo los *JavaBeans* componentes de código reutilizables en cualquier aplicación, Figura 7.60.

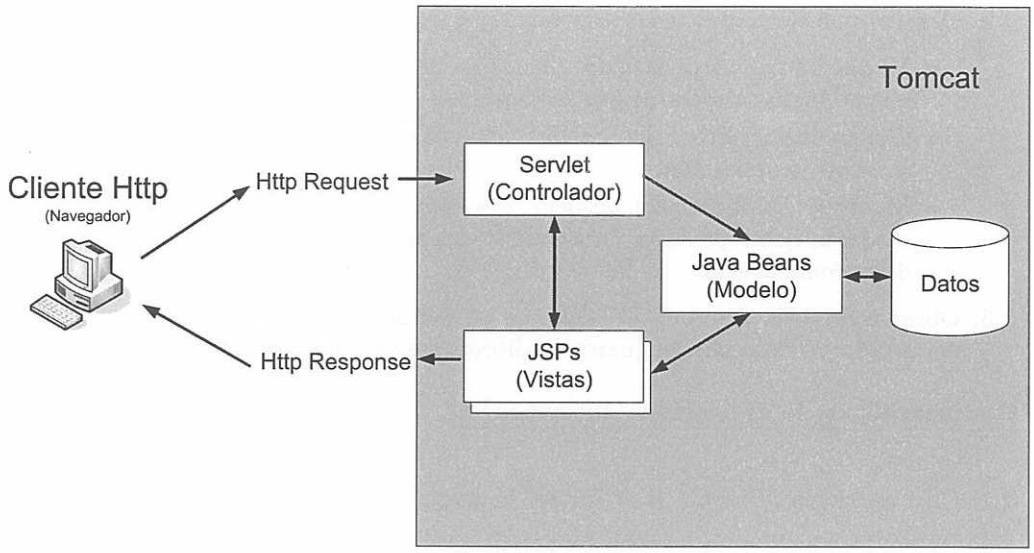


Figura 7.60: Arquitectura MVC

1. **Diseño de la aplicación LibreríaMVC.**

   1.1. Como primer paso vamos a diseñar la aplicación utilizando el patrón **MVC**, Figura 7.61.

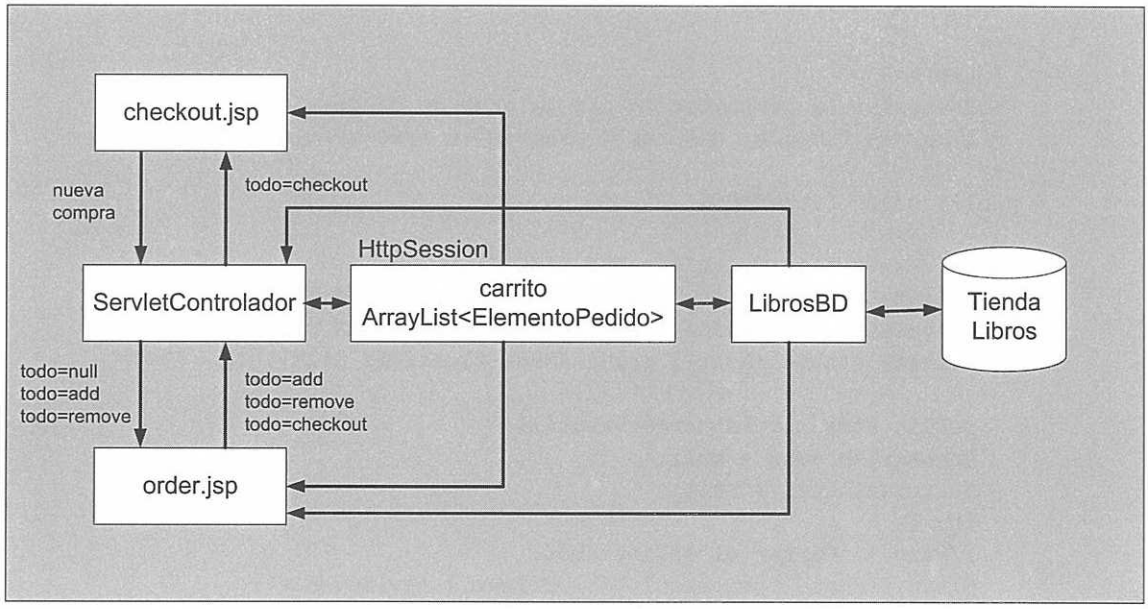


Figura 7.61: Diseño LibreríaMVC

1.2. Observa qué elementos se corresponden con cada uno de los componentes del patrón.

- **Vista**: La página **order.jsp** se encarga de mostrar la página de compras mientras que la página **checkout.jsp** se encargará de mostrar el resumen de la compra.
- **Controlador**: El *servlet* **ServletControlador** se encarga de atender las peticiones y dirigir la aplicación.
- **Modelo**: La lógica del negocio está representada por la clase **ElementoPedido.java**, que representa un elemento del pedido y **LibrosBD.java** que se encarga de la comunicación con la base de datos.

1.3. Observa también cómo en el diagrama se muestra la variable de sesión **carrito**, que permitirá que cada cliente guarde los libros que va seleccionando.

2. **Codificación de la aplicación LibreríaMVC**.

2.1. Crea un nuevo *Dynamic Web Project* llamado **LibreriaMVC**.

2.2. Pulsando con el botón derecho del ratón sobre **Java Resources\src** añadimos la clase **LibrosBD.java** con el siguiente código:

```
package practica;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

/**
 * LibrosBD
 * Encapsula la comunicación con la base de datos
 * Almacena títulos, autores y precios en tres arrays
 */
public class LibrosBD {

   private static final int MAX_SIZE=6;
   private static String[] titulos=new String[MAX_SIZE];
   private static String[] autores=new String[MAX_SIZE];
   private static float[] precios=new float[MAX_SIZE];

   public static void cargarDatos(){
   Connection conn = null;
   Statement stmt = null;
   try {
   //Paso 1: Cargar el driver JDBC.
   Class.forName("com.mysql.jdbc.Driver").newInstance();
   // Paso 2: Conectarse a la Base de Datos utilizando la clase Connection
   String userName="root";
   String password="despliegue";
   //URL de la base de datos(equipo, puerto, base de datos)
   String url="jdbc:mysql://localhost/TiendaLibros";
```

```
        conn = DriverManager.getConnection(url, userName, password);
        // Paso 3: Crear sentencias SQL, utilizando objetos de tipo Statement
        stmt = conn.createStatement();
        // Paso 4: Ejecutar las sentencias SQL a través de los objetos Statement
        String sqlStr = "select * from libros;";
        ResultSet rset = stmt.executeQuery(sqlStr);
        // Paso 5: Procesar el conjunto de registros resultante utilizando ResultSet
        int count = 0;
        while(rset.next()) {
        titulos[count]= rset.getString("autor");
            autores[count]=rset.getString("titulo");
            precios[count]=(float)rset.getDouble("precio");
             count++;
        }
           // Cerramos el resto de recursos
        if (stmt != null) stmt.close();
        if (conn != null) conn.close();
        } catch (Exception ex){
          ex.printStackTrace();
        }
        }

        /** Devuelve el número de libros */
        public static int tamanyo() {
           return titulos.length;
        }

        /** Devuelve el título del libro  idLibro*/
        public static String getTitulo(int idLibro) {
           return titulos[idLibro];
        }

        /** Devuelve el autor del libro  idLibro */
        public static String getAutor(int idLibro) {
           return autores[idLibro];
        }

        /** Devuelve el precio del libro idLibro */
        public static float getPrecio(int idLibro) {
           return precios[idLibro];
        }
    }
```

2.3. Esta clase se encarga de cargar en unos *arrays* los datos de los libros existentes en la base de datos.

2.4. De igual forma añadimos la clase **ElementoPedido.java** con el siguiente código:

```
package practica;

/**
```

```
 * Representa un elemento del pedido
 * Incluye identificador del libro y cantidad
 */

public class ElementoPedido {
   private int idLibro;
   private int cantidad;

   public ElementoPedido(int idLibro, int cantidad) {
      this.idLibro = idLibro;
      this.cantidad = cantidad;
   }

   public int getIdLibro() {
      return idLibro;
   }

   public void setIdLibro(int idLibro) {
      this.idLibro = idLibro;
   }

   public int getCantidad() {
      return cantidad;
   }

   public void setCantidad(int cantidad) {
      this.cantidad = cantidad;
   }

   public String getAutor() {
      return LibrosBD.getAutor(idLibro);
   }

   public String getTitulo() {
      return LibrosBD.getTitulo(idLibro);
   }

   public float getPrecio() {
      return LibrosBD.getPrecio(idLibro);
   }
}
```

2.5. Los objetos creados con esta clase representaran los libros que estan en el carrito.

2.6. Pulsando con el botón derecho del ratón sobre **Java Resources\src** añadiremos el *servlet* **ServletControlador.java** con el siguiente código:

```
package practica;


import java.io.*;
```

```
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class ServletControlador extends HttpServlet {
   private static final long serialVersionUID = 1L;

   @Override
   public void init(ServletConfig conf) throws ServletException {
      super.init(conf);
      LibrosBD.cargarDatos();
   }

   @Override
   protected void doGet(HttpServletRequest request, HttpServletResponse response)
         throws ServletException, IOException {
      doPost(request, response);
   }

   @Override
   protected void doPost(HttpServletRequest request,
         HttpServletResponse response) throws ServletException, IOException {
      // Recupera la sesión actual o crea una nueva si no existe.
      HttpSession session = request.getSession(true);

      // Recupera el carrito de la sesión actual
      List<ElementoPedido> elCarrito =
                           (ArrayList<ElementoPedido>) session.getAttribute("carrito");

      // Determina a que página jsp redirigirá
      String nextPage = "";
      String todo = request.getParameter("todo");

      if (todo == null) {
         // Primer acceso - redirección a order.jsp
         nextPage = "/order.jsp";
      } else if (todo.equals("add")) {
         // Mandado por order.jsp con los parámetros idLibro y cantidad.
         // crea un elementoPedido y lo añade al carrito
         ElementoPedido nuevoElementoPedido = new ElementoPedido(
                 Integer.parseInt(request.getParameter("idLibro")),
                 Integer.parseInt(request.getParameter("cantidad")));
         if (elCarrito == null) { // el carrito está vacío
            elCarrito = new ArrayList<>();
            elCarrito.add(nuevoElementoPedido);
            // enlaza el carrito con la sesión
            session.setAttribute("carrito", elCarrito);
         } else {
            // Comprueba si el libro está en el carrito
            // si lo está actualiza la cantidad
```

```
    // si no lo añade
       boolean encontrado = false;
       Iterator iter = elCarrito.iterator();
       while (!encontrado && iter.hasNext()) {
          ElementoPedido unElementoPedido = (ElementoPedido)iter.next();
          if (unElementoPedido.getIdLibro() == nuevoElementoPedido.getIdLibro())
          {
           unElementoPedido.setCantidad(unElementoPedido.getCantidad()
                   + nuevoElementoPedido.getCantidad());
             encontrado = true;
          }
       }
       if (!encontrado) { // Lo añade al carrito
          elCarrito.add(nuevoElementoPedido);
       }
    }
    // Vuelve a order.jsp para más pedidos
    nextPage = "/order.jsp";
 } else if (todo.equals("remove")) {
    // Enviado por order.jsp con el parámetro indiceElemento
    // Borra el elemento indiceElemento del carrito
    int indiceCarrito = Integer.parseInt(request.getParameter("indiceElemento"));
    elCarrito.remove(indiceCarrito);
    // Vuelve a order.jsp para más pedidos
    nextPage = "/order.jsp";
 } else if (todo.equals("checkout")) {
    // Enviado por order.jsp.
    // Calcula el precio total de todos los elementos del carrito
    float precioTotal = 0;
    int cantidadTotalOrdenada = 0;
    for (ElementoPedido item: elCarrito) {
       float precio = item.getPrecio();
       int cantidadOrdenada = item.getCantidad();
       precioTotal += precio * cantidadOrdenada;
       cantidadTotalOrdenada += cantidadOrdenada;
    }
    // Formatea el precio con dos decimales
    StringBuilder sb = new StringBuilder();
    Formatter formatter = new Formatter(sb);
    formatter.format("%.2f", precioTotal);
    // Coloca el precioTotal y la cantidadTotal en el request
    request.setAttribute("precioTotal", sb.toString());
    request.setAttribute("cantidadTotal", cantidadTotalOrdenada + "");
    // Redirige a checkout.jsp
    nextPage = "/checkout.jsp";
 }

 ServletContext servletContext = getServletContext();
 RequestDispatcher requestDispatcher =
                   servletContext.getRequestDispatcher(nextPage);
```

```
        requestDispatcher.forward(request, response);
    }

}
```

2.7. Observa las últimas líneas de código, utiliza la interfaz *RequestDispatcher* que se utiliza para encaminar solicitudes a otros *Servlet* o *JSP* de la misma aplicación, permitiendo delegar el procesamiento de solicitud respuesta en ellos.

2.8. Para que el *servlet* sea accesible añade la siguiente información al descriptor de despliegue:

```
<servlet>
   <servlet-name>Controlador</servlet-name>
   <servlet-class>practica.ServletControlador</servlet-class>
 </servlet>
 <servlet-mapping>
   <servlet-name>Controlador</servlet-name>
   <url-pattern>/shopping</url-pattern>
 </servlet-mapping>
 <session-config>
   <session-timeout>30</session-timeout>
 </session-config>
 <welcome-file-list>
   <welcome-file>shopping</welcome-file>
 </welcome-file-list>
```

2.9. Añade sobre **WebContent** la página **order.jsp** :

```
<%-- Página de Ordenes  --%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@page session="true" import="java.util.*, practica.*" %>
<!DOCTYPE html>
<html>
   <head>
      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
      <title>Pedido a la Librería MVC</title>
   </head>
   <body>
      <h2>Librería MVC</h2>
      <hr /><br />

      <p><strong>Elige un libro y la cantidad:</strong></p>
      <form name="AnyadirForm" action="shopping" method="POST">
         <input type="hidden" name="todo" value="add">
         Título: <select name=idLibro>
            <%
            // Scriptlet 1: Carga los libros en el select.
            for (int i = 0; i < LibrosBD.tamanyo(); i++) {
               out.println("<option value='" + i + "'>");
               out.println(LibrosBD.getTitulo(i) + " | " + LibrosBD.getAutor(i)
                       + " | " + LibrosBD.getPrecio(i));
```

```
            out.println("</option>");
         }
         %>
      </select>
        Cantidad: <input type="text" name="cantidad" size="10" value="1">
      <input type="submit" value="Añadir a la cesta">
   </form>
   <br /><hr /><br />

   <%
   // Scriptlet 2: Chequea si la cesta esta vacia.
   List<ElementoPedido> cesta =
               (List<ElementoPedido>) session.getAttribute("carrito");
   if (cesta != null && cesta.size() > 0) {
   %>
   <p><strong>Tu cesta contiene:</strong></p>
   <table border="1" cellspacing="0" cellpadding="5">
      <tr>
         <th>Título</th>
         <th>Autor</th>
         <th>Precio</th>
         <th>Cantidad</th>
         <th> </th>
      </tr>
      <%
      // Scriptlet 3: Muestra los libros del carrito.
      for (int i = 0; i < cesta.size(); i++) {
         ElementoPedido elementoPedido = cesta.get(i);
      %>
      <tr>
         <form name="borrarForm" action="shopping" method="POST">
            <input type="hidden" name="todo" value="remove">
            <input type="hidden" name="indiceElemento" value="<%= i %>">
            <td><%= elementoPedido.getTitulo() %></td>
            <td><%= elementoPedido.getAutor() %></td>
            <td align="right"><%= elementoPedido.getPrecio() %></td>
            <td align="right"><%= elementoPedido.getCantidad() %></td>
            <td><input type="submit" value="Eliminar de la cesta"></td>
         </form>
      </tr>
      <%
      }
      %>
    </table><br />

   <form name="checkoutForm" action="shopping" method="POST">
      <input type="hidden" name="todo" value="checkout">
      <input type="submit" value="Checkout">
   </form>
   <%
```

```
      } // if
      %>
   </body>
</html>
```

2.10. Añade sobre **WebContent** la página **checkout.jsp**:

```
<%-- Página de Checkout   --%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@page session="true" import="java.util.*, practica.*" %>
<!DOCTYPE html>
<html>
   <head>
      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
      <title>Libreria MVC Checkout</title>
   </head>
   <body>
      <h2>Libreria MVC - Checkout</h2>
      <hr /><br />

      <p><strong>Has comprado los siguientes libros:</strong></p>
      <table border="1" cellspacing="0" cellpadding="5">
         <tr>
            <th>Título</th>
            <th>Autor</th>
            <th>Precio</th>
            <th>Cantidad</th>
         </tr>
         <%
         // Scriptlet 1: Muestra los elementos del carrito
         List<ElementoPedido> cesta =
                              (List<ElementoPedido>) session.getAttribute("carrito");
         for (ElementoPedido item : cesta) {
         %>
         <tr>
            <td><%= item.getTitulo()%></td>
            <td><%= item.getAutor()%></td>
            <td align="right"><%= item.getPrecio()%></td>
            <td align="right"><%= item.getCantidad()%></td>
         </tr>
         <%
         } // for
         session.invalidate();
         %>
         <tr>
            <th align="right" colspan="2">Total</th>
            <th align="right"><%= request.getAttribute("precioTotal")%></th>
            <th align="right"><%= request.getAttribute("cantidadTotal")%></th>
         </tr>
      </table>
      <br />
```

```
        <a href="shopping">Pulsa aquí para comprar más libros</a>
    </body>
</html>
```

3. **Funcionamiento de la aplicación LibreríaMVC.**

   3.1. Fíjate en que el *servlet* será el punto de entrada a la aplicación. Podemos acceder indicando la *URL* asociada al *servlet* `http://localhost:8080/LibreriaMVC/shopping`, o la *URL* de la aplicación `http://localhost:8080/LibreriaMVC`, porque en el descriptor de despliegue hemos indicado como *welcome-file-list* de la aplicación el recurso *shopping*.

   3.2. En el primer acceso se crea un sesión HTTP que mantendrá el carrito de compra en el atributo de sesión **"carrito"**. Después se redirige la salida a la pagína **order.jsp** (todo=null).

   3.3. La página **order.jsp** muestra los libros disponibles y el carrito de compra. Cada vez que el usuario añade o borra un libro del carrito una petición *request* con los parámetros "todo=add" o "todo=remove" es enviada al **ServletControlador**.

   3.4. Cuando el usuario pulsa en **Checkout** una petición *request* con los parámetros "todo=checkout" es enviada al **ServletControlador**.

   3.5. Cuando el *ServletControlador* recibe los parámetros "todo=add" o "todo=remove", el controlador accede a la variable de sesión y actualiza el carrito, después redirige a **order.jsp** para seguir comprando.

   3.6. Cuando el *ServletControlador* recibe el parámetro "todo=checkout", el controlador contabiliza el precio total y la cantidad total de libros y se redirige a **checkout.jsp**.

   3.7. Ejecuta la aplicación desde *Eclipse* y comprueba su funcionamiento, Figura 7.62, 7.63:

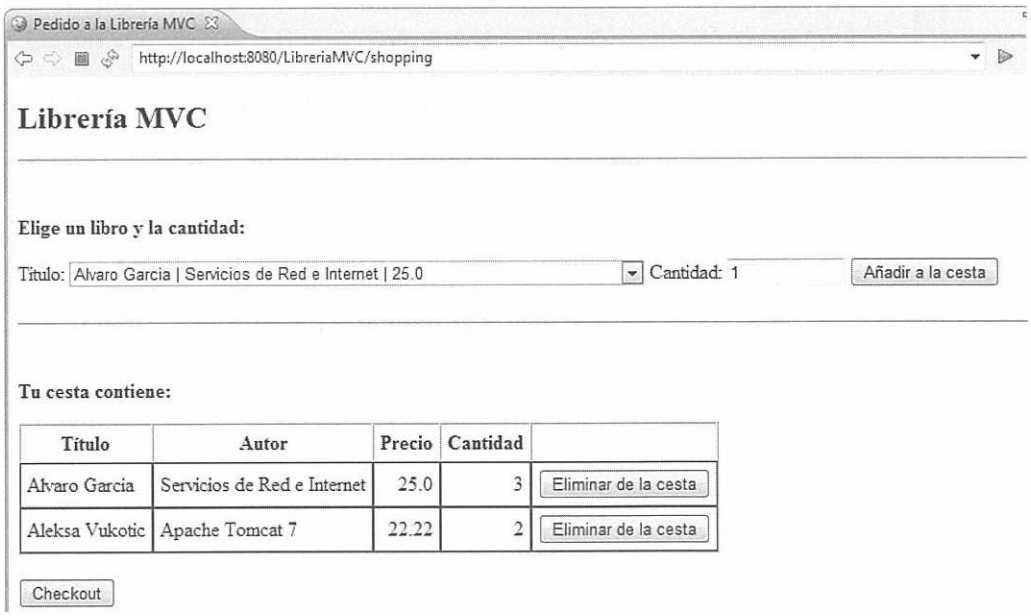


Figura 7.62: Funcionamiento LibreríaMVC

Libreria MVC Checkout

http://localhost:8080/LibreriaMVC/shopping

Libreria MVC - Checkout

Has comprado los siguientes libros:

Título	Autor	Precio	Cantidad
Alvaro Garcia	Servicios de Red e Internet	25.0	3
Aleksa Vukotic	Apache Tomcat 7	22.22	2
	Total	**119,44**	**5**

Pulsa aquí para comprar más libros

Figura 7.63: Funcionamiento LibreríaMVC

4. Finalmente, exporta la aplicación como fichero **.war**, despliégala y accede a la misma simultáneamente con los navegadores *Internet Explorer* y *Firefox*. Comprueba cómo se mantienen las dos sesiones independientes, Figura 7.64:

5. La aplicación está desarrollada suponiendo que ambos navegadores aceptan *cookies* para la gestión de la sesión. Si esto no fuera así el funcioamiento no sería el esperado.

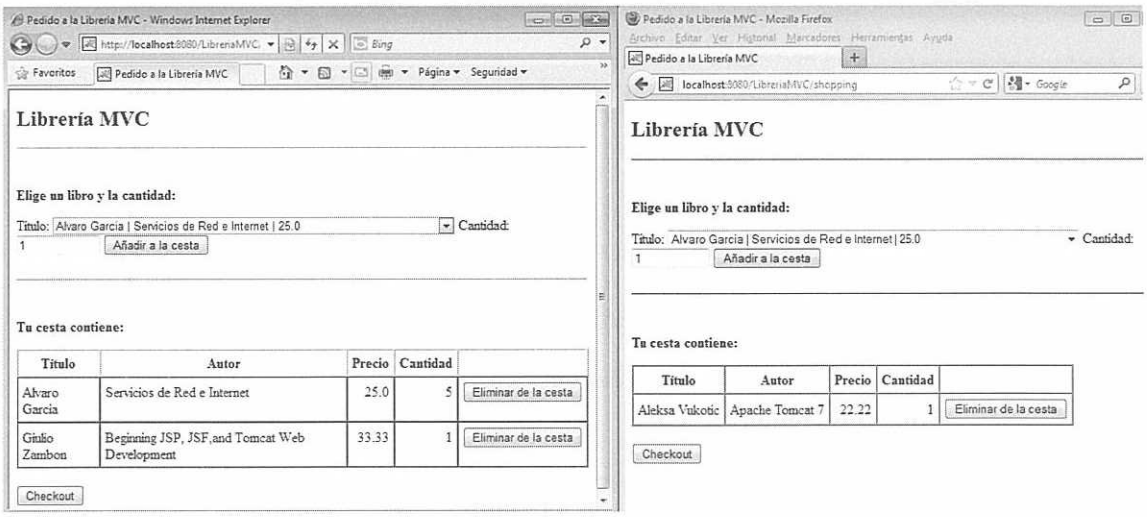


Figura 7.64: Sesiones simultáneas LibreríaMVC

7.11. Instalación de *Spring Framework* en *Windows*

Spring es el *framework* más utilizado para el desarrollo de aplicaciones empresariales con *Java*, ya que facilita la creación de aplicaciones de alto redimiento, facilmente testeables y con un código fácil de reutilizar.

Entre los beneficios de usar *Spring* podemos destacar las siguientes:

- Permite desarrollar aplicaciones empresariales utilizando *POJOs*. El beneficio de usar *POJOs* radica en que no es necesario un servidor de aplicaciones para contener *EJB*, ya que con un contenedor de *servlets* como *Tomcat* será suficiente.
- Está organizado en módulos independientes, por lo cual sólo nos preocuparemos de los módulos que vayamos a utilizar e ignoraremos la complejidad de los módulos restantes.
- Otra característica destacable es su popular modulo Web que utiliza el patrón **MVC**, modelo vista controlador.

Dependency Injection (DI): Una de las tecnologías con las que se identifica *Spring* es *Dependency Injection (DI)*. Cuando desarrollamos una aplicación compleja en *Java*, las clases deberían ser tan independientes como sea posible para aumentar las posibilidades de reutilizarlas. *DI* nos ayudará a conseguir que esas clases trabajen juntas pero se mantengan independientes.

En cuanto a la arquitectura de *Spring*, Figura 7.65, podemos identificar los siguientes componentes:

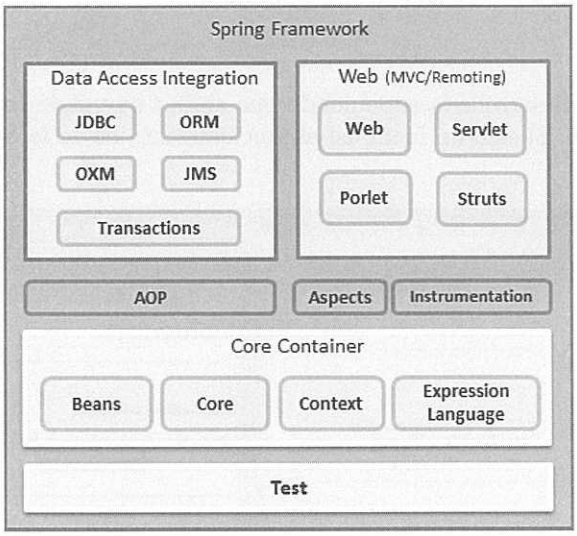


Figura 7.65: *Arquitectura del framework de Spring*

- El *Core Container* es la parte fundamental del *framework*, se encarga de crear los objetos y gestionar su ciclo de vida. Utiliza *Dependency Injection* para gestionar los objetos, llamados *Spring Beans*, que constituyen la aplicación. El modulo *Context* nos permitirá acceder a los objetos gracias a la interfaz *ApplicationContext*.

- En *Data Access Integration* tenemos todos los módulos relativos al acceso a datos, desde el módulo *JDBC* que permite simplificar el tedioso código *JDBC*, al módulo *ORM* que permite la integración con *JPA* o *Hibernate*.
- Dentro de Web, tenemos el módulo *Web-Servlet* que contiene la versión de *Spring* de *MVC* para implementar aplicaciones Web.

1. **Configuración de *Spring Framework* en *Eclipse*.**

   1.1. Accede a la página `http://www.springsource.org/download/community` y descarga el *Framework* de *Spring*. En el momento de hacer esta documentación estaba disponible la versión *Latest GA release: 3.2.3.RELEASE* (**spring-framework-3.2.3.RELEASE-dist.zip**)

   1.2. Descomprime el fichero en la carpeta **C:\Spring**.

   1.3. Comprueba cómo en el directorio **C:\Spring\spring-framework-3.2.3.RELEASE-dist\libs** aparecen los ficheros **.jar** que componen el *framework* de *Spring*, Figura 7.66.

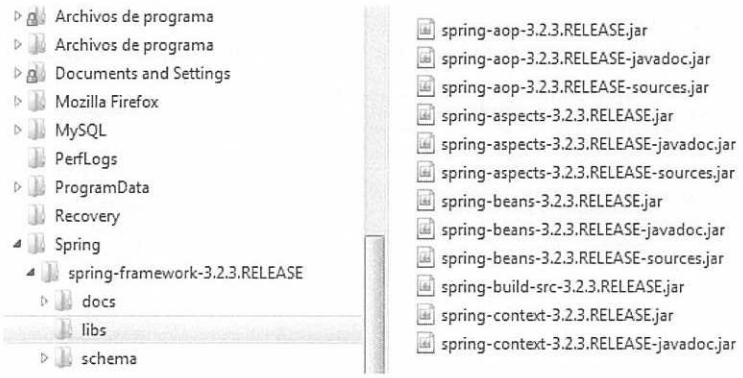


Figura 7.66: Ficheros **jar** del *framework* de *Spring*

   1.4. Para que *Spring* pueda funcionar necesita tener disponible el sistema de *logging Apache Commons Logging*, `http://commons.apache.org/proper/commons-logging/`.
   Para obtenerlo accede a la dirección:
   `http://commons.apache.org/proper/commons-logging/download_logging.cgi` y descarga el fichero **commons-logging-1.1.3-bin.zip**, Figura 7.67.

Figura 7.67: *Commons Logging*

   1.5. Una vez descargado lo descomprimiremos en la carpeta **C:\Spring**

2. **Creación de un proyecto *Java* utilizando *Spring*.**

Para empezar a introducirnos en el mundo de *Spring*, vamos a crear una aplicación *Java* utilizando *Spring*.

2.1. Crea un nuevo proyecto de *Java*, *Java Project*, llamado **HolaMundoSpring**.

2.2. Ahora es necesario que las librerías de *Spring* y *Common Logging* estén disponibles para nuestro proyecto. Para ello pulsamos con el botón derecho del ratón sobre el directorio del proyecto, y accedemos a la opción **Build Path, Configure Build Path**, Figura 7.68.

Figura 7.68: ***Build Path***

2.3. Ahora en **Libraries**, añadiremos como *external JARs* los ficheros *.jar* que componen el *framework* de *Spring* y que previamente habiamos descargado, Figura 7.69.

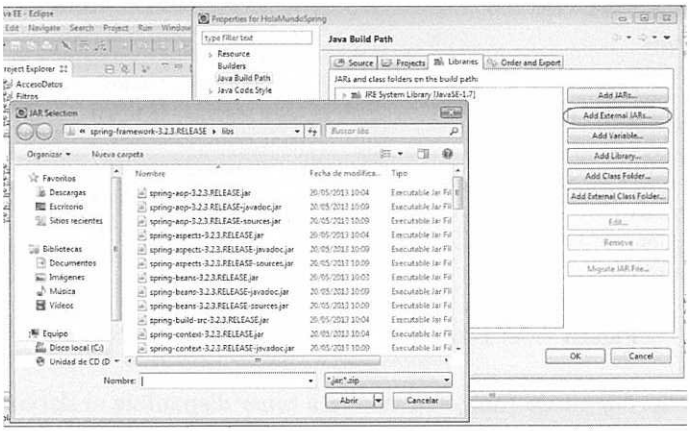


Figura 7.69: ***Add External JARs***

2.4. También añadiremos los ficheros **.jar** que forman *Common Logging*.

2.5. Ahora sobre el directorio **src**, añade una clase *Java*, llamada **HolaMundo**, con el siguiente código:

```
public class HolaMundo {
    private String message;
    public void setMessage(String message){
this.message=message;
    }
    public void getMessage(){
     System.out.println("Hola y bienvenido a Spring: " + message);
    }
}
```

Esta clase es lo que denominamos un *POJO* (*Plain old Java Object*), es decir, una clase que no depende de ningún *framework* ni implementa ninguna interfaz especial. El contenedor de *Spring* tomará estas clases (*POJOs*), las instanciará y las configurará mediante metadatos, anotaciones o código *Java*. El resultado serán los *Spring Beans*, los objetos que, contenidos por el *framework*, constituirán nuestra aplicación Figura 7.70.

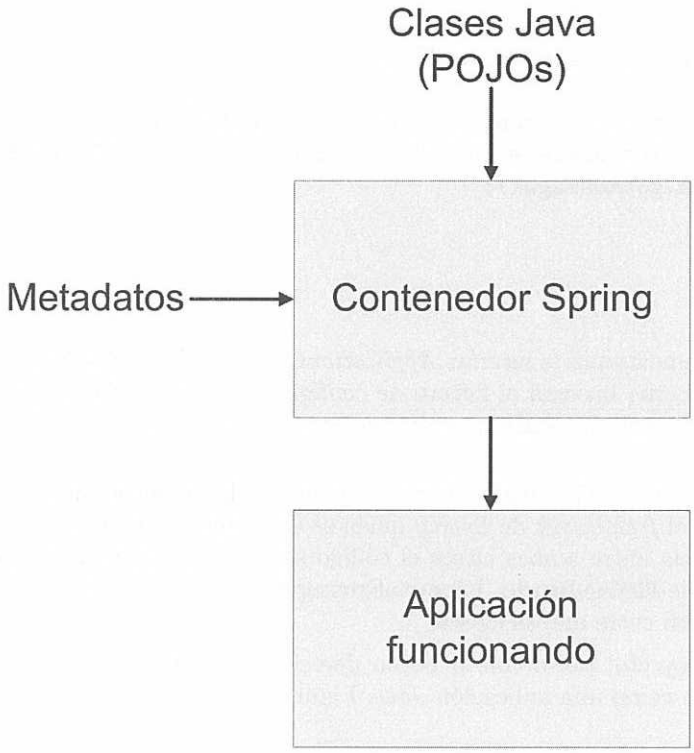


Figura 7.70: ***Funcionamiento aplicación Spring***

2.6. A continuación, añade en la carpeta **src** un fichero *XML*, llamado **Beans.xml**, con el siguiente contenido:

```
<?xml version="1.0" encoding="UTF-8"?>

<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans-3.0.xsd">

   <bean id="holamundo" class="HolaMundo">
       <property name="message" value="alumno"/>
   </bean>

</beans>
```

En este fichero definimos el *Bean* con *id* **holamundo** que corresponde a la clase **HolaMundo** y establecemos un valor para su propiedad **message**.

2.7. Sobre el mismo directorio **src** añade otra clase llamada **EjecutaSpring**, con el siguiente código:

```
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class EjecutaSpring
{
    public static void main(String[] args)
    {
        ApplicationContext context = new ClassPathXmlApplicationContext("Beans.xml");
        HolaMundo instancia = (HolaMundo) context.getBean("holamundo");
        instancia.getMessage();

      }
}
```

Observa cómo implementamos la interfaz *ApplicationContext*, con la clase *ClassPathXmlApplicationContext*, que buscará el fichero de configuración de los *Spring Beans* en el *CLASSPATH*. Con el método *getBean*, podremos acceder a la instancia del *Spring Bean*.

Comprueba también como **EjecutaSpring** no depende de la implementación de la clase **HolaMundo**, será el *framework* de *Spring* quién se encargue de crear el objeto. Eliminada esa dependencia entre ambas clases el código será más independiente, si cambia la implementación de **HolaMundo**, **EjecutaSpring** no se verá afectada, consiguiendo un bajo acoplamiento entre ambas clases.

2.8. Para ejecutar el proyecto, pulsa con el botón derecho del ratón sobre la carpeta del proyecto y ejecútalo como una aplicación *Java*, Figura 7.71.

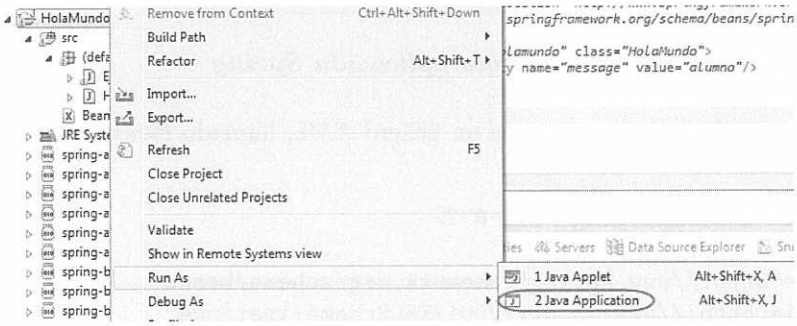


Figura 7.71: ***Ejecución del proyecto***

2.9. Obtendrás el siguiente resultado, Figura 7.72.

```
Markers  Properties  Servers  Data Source Explorer  Snippets  Console  Debug
<terminated> EjecutaSpring [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (24/06/2013 11:37:31)
jun 24, 2013 11:37:31 AM org.springframework.context.support.ClassPathXmlApplicationContext prepareRefresh
INFO: Refreshing org.springframework.context.support.ClassPathXmlApplicationContext@1cc5069: startup date [M
jun 24, 2013 11:37:31 AM org.springframework.beans.factory.xml.XmlBeanDefinitionReader loadBeanDefinitions
INFO: Loading XML bean definitions from class path resource [Beans.xml]
jun 24, 2013 11:37:32 AM org.springframework.beans.factory.support.DefaultListableBeanFactory preInstantiate
INFO: Pre-instantiating singletons in org.springframework.beans.factory.support.DefaultListableBeanFactory@1
Hola y bienvenido a Spring: alumno
```

Figura 7.72: ***Resultado de la ejecución***

7.12. *Spring* MVC

Spring MVC es un módulo del *framework Spring* para el desarrollo de aplicaciones Web utilizando *Servlets*. Utiliza el patrón MVC para organizar la arquitectura de la aplicación Web, y el patrón *Front Controller* para el diseño de los componentes de la misma.

Spring MVC no es más que un conjunto de clases de *Java* que serán contenidas por *Tomcat*, que nos permitirá hacer aplicaciones más eficientes, más fáciles de desarrollar y de mantener, Figura 7.73.

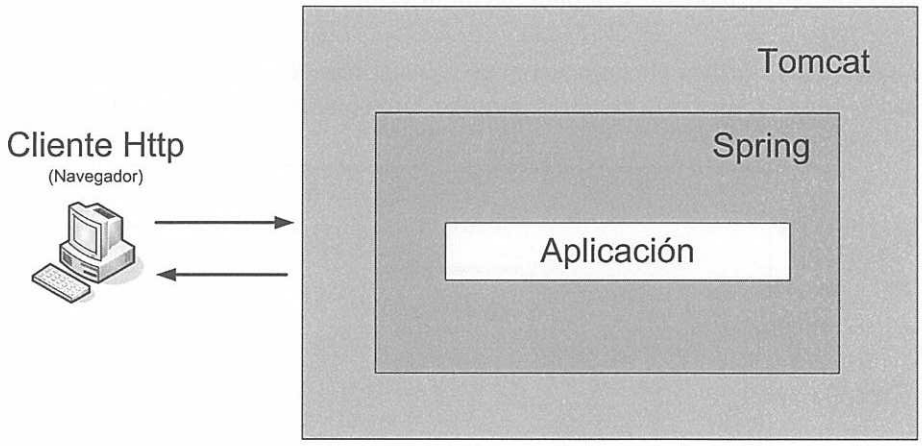


Figura 7.73: *Relación Tomcat y Spring*

Como vimos en la **práctica 7.10** utilizar el patrón *MVC*, permite dividir una aplicación web en la parte controla la aplicación (Controlador), la parte encargada de los datos y reglas de negocio (Modelo) y la parte encargada de mostrar la información (Vista). Gracias a este patrón es fácil disponer de múltiples vistas y los cambios en cualquiera de los componentes del patrón serán más sencillos.

En *Spring MVC*, el controlador es representado con la interfaz *Controller*, el modelo es representado como una colección *Map* de pares clave-valor, donde los valores son los objetos que representan el estado del modelo y son accesibles mediante claves. Finalmente, las vistas pueden ser creadas utilizando diferentes tecnologías como *HTML* o *JSP*.

1. **Patrón *Front Controller*.**

   El patrón *Front Controller* especifica que todas las peticiones web que llegan a nuestra aplicación son manejadas desde el mismo punto de entrada, llamado *Front Controller*, y que se

encargará de redirigir cada petición al componente que deba gestionarla. Una vez tratada la petición, se devuelve el control al *Front Controller* que es el responsable de enviar la respuesta al cliente. Además de esta tarea de redirección *Front Controller* es el encargado de más funcionalidades como la gestión de sesiones y los filtros.

En el *Spring MVC framework*, este papel lo realiza la implementación de un *Servlet*, en concreto la clase `org.springframework.web.servlet.DispatcherServlet`, que funcionará como el portero que guarda la entrada a nuestra aplicación y que no realiza ninguna tarea relativa a la lógica de negocio, si no que delega a clases *POJOs* que actúan de controladores y realizan el trabajo.

2. **Arquitectura *Spring MVC*.**

   La clase *DispatcherServlet*, que hace de *Front Controller* en *Spring MVC*, procesa todas las peticiones de clientes, delegando cada petición al componente controlador adecuado. Durante la gestión de la petición el controlador puede colaborar con cualquiera de los *Spring Beans*.

   Una vez finalizada la gestión de la petición, el controlador devuelve una instancia de *ModelAndView* al *DispatcherServlet*. *ModelAndView* es la abstracción que utiliza *Spring* para contener los datos (modelo) y la plantilla de la vista.

   DispatcherServlet utiliza el componente de *Spring ViewResolver* para cargar la plantilla de la vista y, una vez añadidos los datos, mandar la respuesta al cliente en *HTML*, Figura 7.74.

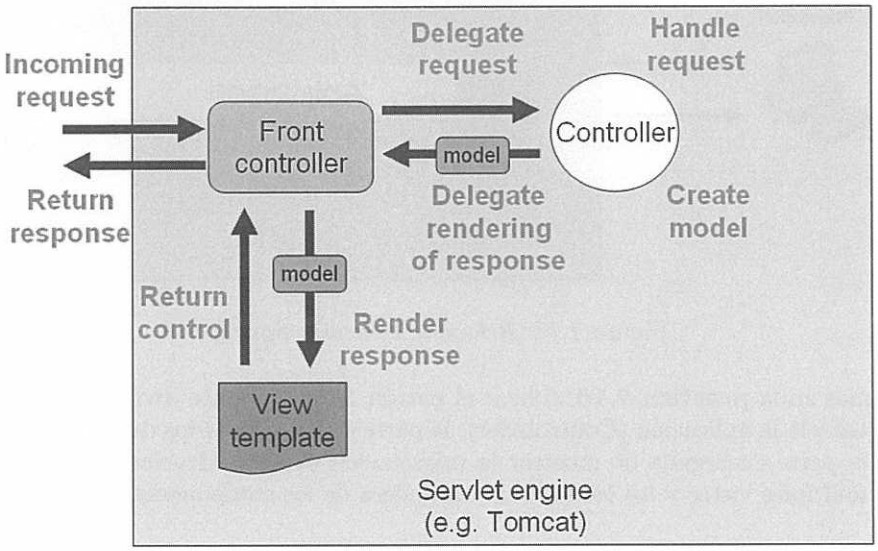


Figura 7.74: *Arquitectura Spring MVC*

3. **Creación de una aplicación utilizando *Spring MVC*.**

Para crear nuestra aplicación utilizando Spring MVC tendremos que configurar los componentes que necesitemos y codificar únicamente los controladores y las vistas.

3.1. Crea un nuevo proyecto *Dynamic Web Project*, llamado **HolaMundoSpringMVC**. Recuerda marcar la opción para crear el descriptor de despliegue **web.xml**.

3.2. Añade los ficheros **.jar** de *Spring* y *Common Logging* a la carpeta **/WEB-INF/lib**.

3.3. Lo primero será configurar el *servlet* `org.springframework.web.servlet.DispatcherServlet`, que va a actuar como *Front Controller*, en el descriptor de despliegue de la aplicación. Observa la configuración:

- Se registra el *servlet* **DispatcherServlet** con el nombre **ControladorPrincipal**.
- Se configura para que gestione todas las peticiones que reciba la aplicación, cualquier ruta que empiece con / relativa a la aplicación.
- Al desplegar la aplicación se carga el servlet en memoria.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<web-app xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
version="3.0">
<display-name>HolaMundoSpringMVC</display-name>
<description>Ejemplo de utilizacion de Spring MVC</description>
<servlet>
<servlet-name>ControladorPrincipal</servlet-name>
<servlet-class>org.springframework.web.servlet.DispatcherServlet
</servlet-class>
<load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
<servlet-name>ControladorPrincipal</servlet-name>
<url-pattern>/</url-pattern>
</servlet-mapping>
</web-app>
```

3.4. El siguiente paso será la implementación de las vistas. Crea el directorio **views** en **WEB-INF**, y añade el fichero **index.jsp** con el siguiente contenido:

```
<html>
  <head>
   <title>Hola Mundo Spring MVC</title>
   <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
  </head>
  <body>
    <div class="content">
      <b>
          ${message}
       </b>
```

```
    </div>
  </body>
</html>
```

3.5. A continuación implementaremos los controladores, añade el controlador **ControladorHolaMundo** en **src** en el paquete **unidad7**, con el siguiente código:

```
package unidad7;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.servlet.ModelAndView;

import java.util.HashMap;
import java.util.Map;

@Controller
public class ControladorHolaMundo {

    @RequestMapping(value = "/*.html", method = RequestMethod.GET)
    public ModelAndView hello(){
        Map<String, Object> model = new HashMap<String, Object>();
        model.put("message", "Hola Mundo Spring MVC");

        return new ModelAndView("index", model);
    }

}
```

Aparte del fichero de configuración de *Spring* que utilizamos anteriormente, *Spring* también se configura utilizando anotaciones, en este fichero utilizamos las anotaciones *@Controller* que indica que esa clase es un controlador en *Spring MVC* y *@RequestMapping*, que indica que peticiones gestionará el método **hello()**.

Observa cómo este controlador recibirá las peticiones a cualquier fichero *html* enviados por el método *GET*. El método **hello** creará la colección *model*, que representa el modelo y guardará en ella, el mensaje **Hola Mundo Spring MVC** asociado a la clave **message**. Después devuelve un objeto *ModelAndView* donde indica la vista **index** y el modelo creado.

3.6. El controlador principal que actua de *FrontController*, es el encargado de cargar el *Spring ApplicationContext*, fichero de configuración de *Spring*. El nombre del fichero de configuración debe ser **Nombre del Controlador Principal-servlet.xml** y debe guardarse en el directorio **WEB-INF**, en nuestro caso se llamará **ControladorPrincipal-servlet.xml**, y tendrá el código siguiente:

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
```

```
        xmlns:context="http://www.springframework.org/schema/context"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:mvc="http://www.springframework.org/schema/mvc"
        xsi:schemaLocation="http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
        http://www.springframework.org/schema/mvc
        http://www.springframework.org/schema/mvc/spring-mvc.xsd
        http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context-3.0.xsd">

  <context:component-scan base-package="unidad7" />

  <mvc:annotation-driven/>

  <bean id="viewResolver"
        class="org.springframework.web.servlet.view.InternalResourceViewResolver">
      <property name="prefix" value="/WEB-INF/views/"/>
      <property name="suffix" value=".jsp"/>
  </bean>

</beans>
```

- Comprueba, como indicamos, que los componentes *Spring*, en nuestro caso el controlador, estará en el paquete **unidad7**.
- `<mvc:annotation-driven/>`
  indica que estamos utilizando el modelo de controladores de *Spring MVC*.
- Observa también, cómo configuramos el componente que se encargará de resolver las vistas, indicando que las vistas se guardarán en **/WEB-INF/views/** y tendrán extensión **.jsp**. Si recuerdas, el controlador creaba el objeto *ModelAndView* indicando que la vista era **index**, gracias a lo que acabamos de configurar se puede saber que la vista será **/WEB-INF/views/index.jsp**.

3.7. Para comprobar el funcionamiento de la aplicación debes acceder a `http://localhost:8080/HolaMundoSpringMVC/cualquiercosa.html`, Figura 7.75.

Como resumen vamos a enumerar las tareas que realiza nuestra aplicación:

a. Primeramente cuando la aplicación se despliega, se carga el *servlet* que actúa de *FrontContoller*, *DispatcherServlet*, porque así no hemos establecido en el descriptor de despliegue de la apliación, **web.xml**.
b. *DispatcherServlet* carga el fichero de configuración de *Spring*, **ControladorPrincipal-servlet.xml**, que indica que los componentes *Spring* estarán en el paquete unidad7.
c. El contenedor localiza que existe un **ControladorHolaMundo** que atenderá a las peticiones ***.html**.
d. Accedo a `http://localhost:8080/HolaMundoSpringMVC/cualquiercosa.html`.
e. El *DispatcherServlet* captura la petición y la delega al **ControladorHolaMundo**.
f. El método **hello()** hace su trabajo y devuelve al *DispatcherServlet* un objeto *ModelAndView*, que incluye el nombre de la vista **index** y una referencia al modelo, **model**.

g. El **Controlador Principal** gracias al **viewResolver** configurado en el fichero de configuración de **Spring**, determina que la salida debe ser generada por el fichero **WEB-INF/views/index.jsp**, que internamente accede a la variable **message** del modelo. La respuesta generada por este fichero **jsp** es la enviada al cliente como respuesta.

Figura 7.75: *Funcionamiento aplicación MVC*

Capítulo 8

Administración de Tomcat

8.1. Instalación de *Apache Tomcat 7.0* en *Linux*

Instala el servidor *Apache Tomcat 7.0* en la máquina **ServidorLinuxXX** desde los repositorios de software de *Ubuntu*.

1. **Instalación de JDK**.

   Apache Tomcat 7.0 requiere para su funcionamiento tener instalada la máquina virtual de *Java* (*Java Standard Edition Runtime Environment (JRE)* en su versión 6.0 o posterior. Vamos a instalar *Java Standard Edition Development Kit (JDK)* que incluye, además de la máquina virtual, las herramientas de desarrollo.

   1.1. Inicia sesión en **ServidorLinuxXX** con un usuario con privilegios de administración.

   1.2. Instala la versión 7 del JDK.

   ```
   sudo apt-get update
   sudo apt-get install openjdk-7-jdk
   ```

   1.3. A continuación, edita el fichero **/etc/environment** y establece la variable `JAVA_HOME`, que debe apuntar al directorio donde está instalado *JDK*, y la variable `JRE_HOME`, que apuntará al directorio donde está instalada la máquina virtual de *Java* (puedes consultar información sobre las variables de entorno en *Ubuntu* en la siguiente dirección `https://help.ubuntu.com/community/EnvironmentVariables`).

   ```
   JAVA_HOME=/usr/lib/jvm/java-7-openjdk-i386
   JRE_HOME=/usr/lib/jvm/java-7-openjdk-i386/jre
   ```

   1.4. Reinicia el sistema.

2. **Instalación de *Tomcat***.

   2.1. Instala el servidor *Tomcat 7.0* desde los repositorios oficiales de *Ubuntu*.

   ```
   sudo apt-get install tomcat7
   ```

   - Se crean los archivos de configuración.
   - Se crea el usuario **tomcat7** que se añade al grupo **tomcat7**.
   - En la instalación de los paquetes precompilados de *Ubuntu* (.deb) (la que hemos realizado) *Tomcat* se ha instalado usando en las siguientes ubicaciones.

     ```
     /usr/share/tomcat7 (se denominará CATALINA_HOME)
     /var/lib/tomcat7 (se denominará CATALINA_BASE)
     ```
   - Si hubiésemos descargado y descomprimido el código de *Tomcat* por ejemplo **/opt/tomcat7** (localización habitual) las variables `CATALINA_HOME` y `CATALINA_BASE` apuntarían ambas a **/opt/tomcat7** y se suele usar `CATALINA_HOME`.

   2.2. Edita el fichero **/etc/environment** y establece las variables *CATALINA_HOME* y *CATALINA_BASE*, que apuntaran a los directorios **/usr/share/tomcat7** y **/var/lib/tomcat7** respectivamente.

   ```
   CATALINA_HOME = /usr/share/tomcat7
   CATALINA_BASE = /var/lib/tomcat7
   ```

   2.3. Reinicia la sesión para que las variables de entorno tomen valor.

2.4. Comprueba que el servidor está iniciado y escuchando en el puerto 8080/TCP.

```
ps -ef | grep tomcat
netstat -ltn
```

2.5. Comprueba que en **/etc/init.d** existe un *script* **tomcat7**, que es invocado en el arranque de *Linux* para iniciar *Tomcat* automáticamente.

2.6. Consulta el contenido del directorio **/var/lib/tomcat7/webapps/ROOT/**.

2.7. Consulta el contenido del fichero **/var/lib/tomcat7/webapps/ROOT/index.html**.

3. <u>**Prueba de conexión al servidor**</u>

3.1. Desde el equipo **DesarrolloW7XX** accede al servidor *Tomcat* para comprobar su funcionamiento, `http://192.168.1.X7:8080`, Figura 8.1.

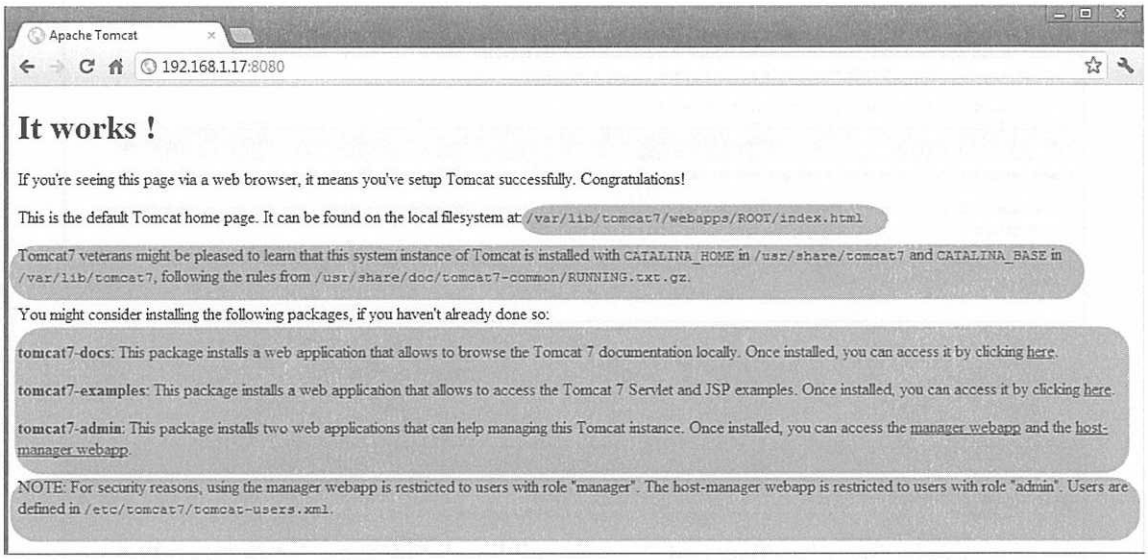


Figura 8.1: Comprobación del funcionamiento de *Tomcat*

3.2. También puedes usar los nombres DNS `servidorlinuxXX.dawXX.net` y `obelix.dawXX.net`

3.3. Observa que *Tomcat* sirve por defecto el fichero **index.html** (que está en el directorio **/var/lib/tomcat7/webapps/ROOT/**).

3.4. Observa la información sobre los directorios de instalación y los paquetes adicionales.

4. **Instalación de la documentación en local y de aplicaciones de ejemplo**

4.1. Instala los siguientes paquetes:

- *tomcat7-docs*, que instala la documentación en local.
- *tomcat7-examples*, con ejemplos de *servlets* y *JSPs*.

```
sudo apt-get install tomcat7-docs tomcat7-examples
```

4.2. Desde **DesarrolloW7XX** accede a la documentación de *Tomcat*, Figuras 8.2 y 8.3 .

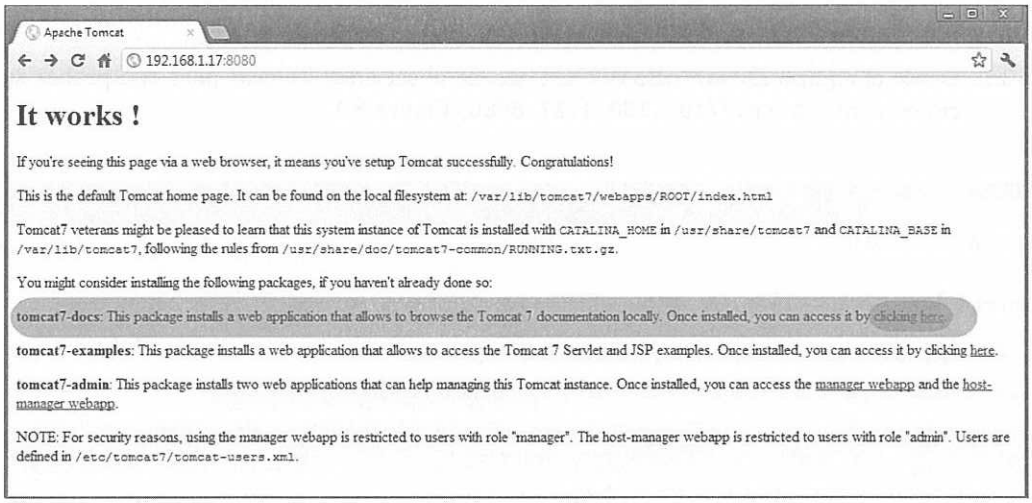


Figura 8.2: Acceso a la documentación

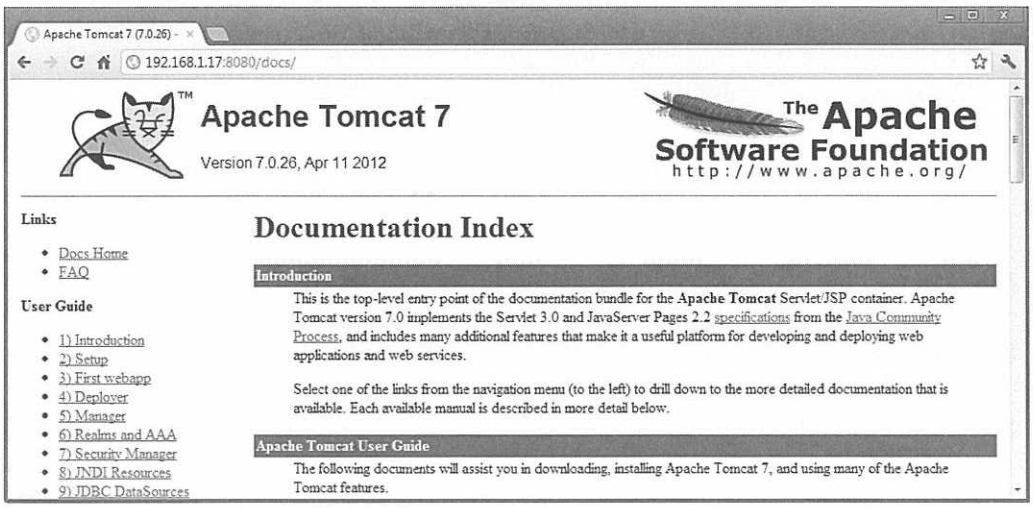


Figura 8.3: Acceso a la documentación

4.3. Desde **DesarrolloW7XX** accede a los ejemplos instalados y ejecuta el *servlet* de HolaMundo, Figuras 8.4, 8.5 y 8.6.

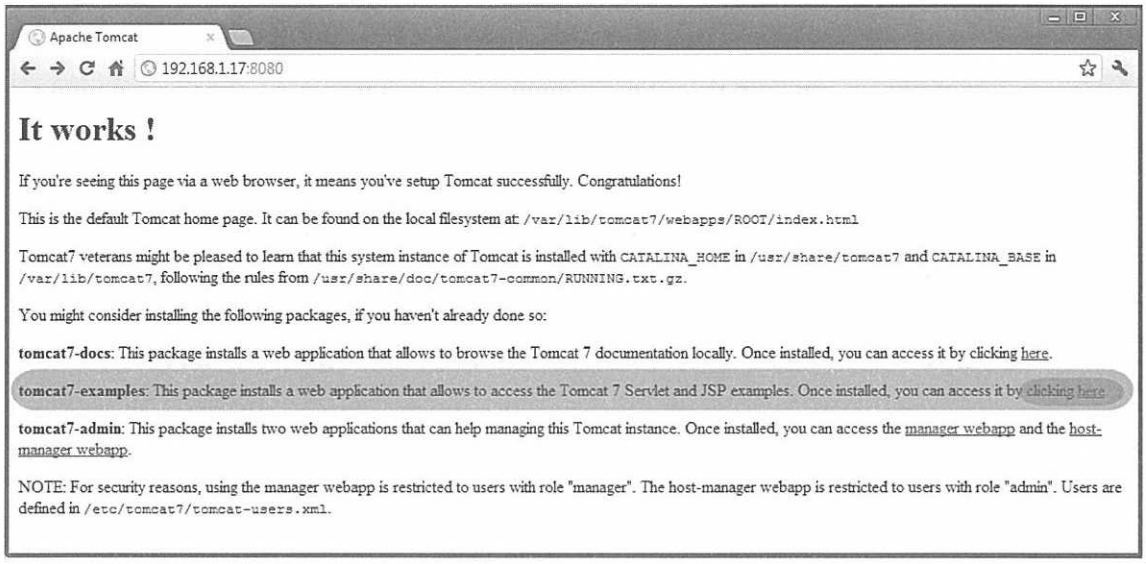


Figura 8.4: Comprobación del funcionamiento del *servlet*

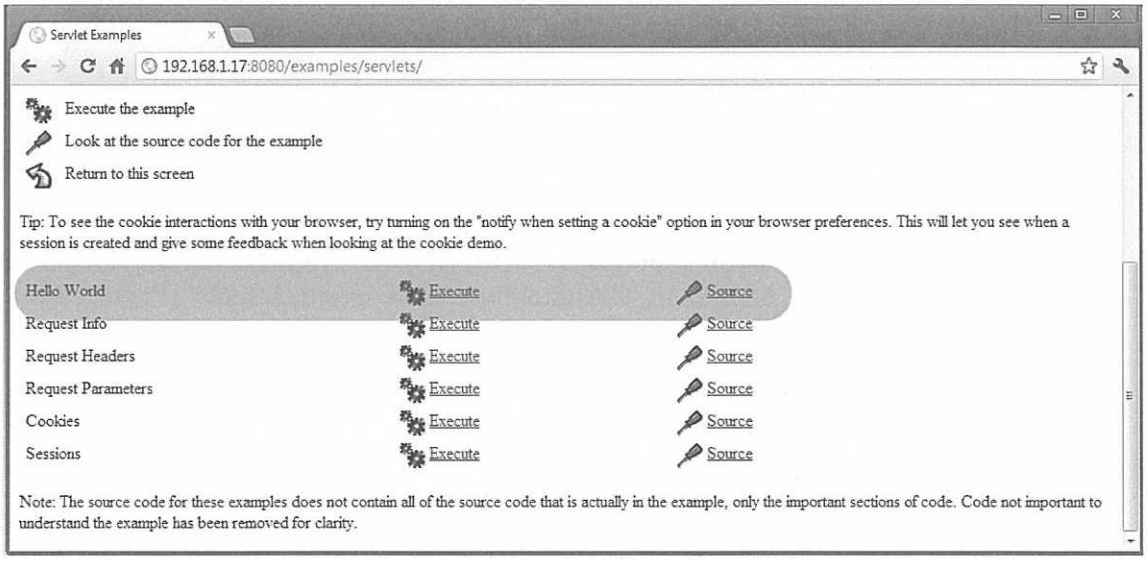


Figura 8.5: Comprobación del funcionamiento del *servlet*

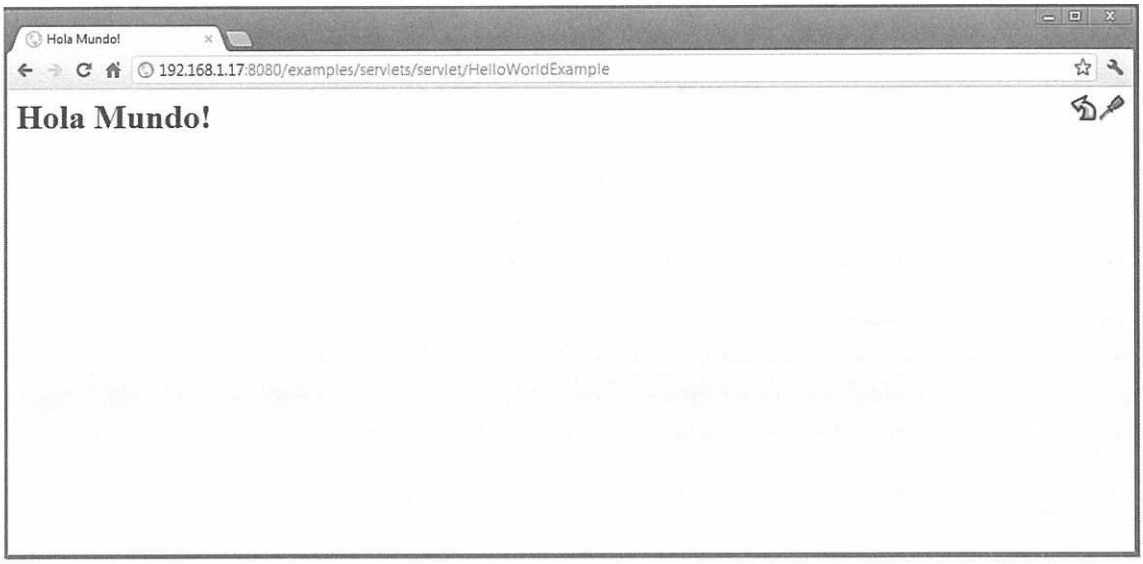


Figura 8.6: Comprobación del funcionamiento del *servlet*

4.4. Aunque todavía no quede muy claro con consulta el contenido del directorio **$CATALINA_BASE/conf/ catalina/localhost**, Figura 8.7.

```
alumno@ServidorLinux01:/var/lib/tomcat7/conf/Catalina/localhost$ ls -l
total 8
-rw-r--r-- 1 root root 922 mar  7 18:36 docs.xml
-rw-r--r-- 1 root root  78 mar  7 18:36 examples.xml
alumno@ServidorLinux01:/var/lib/tomcat7/conf/Catalina/localhost$
```

Figura 8.7: Directorio **/var/lib/tomcat7/conf/catalina/localhost**

4.5. Consulta el contenido de los ficheros **$CATALINA_BASE/conf/Catalina/localhost/doc.xml** y **$CATALINA_BASE/conf/Catalina/localhost/examples.xml**, Figuras 8.8 y 8.9.

```
<Context path="/docs"
        docBase="/usr/share/tomcat7-docs/docs"
        antiResourceLocking="false" />
```

Figura 8.8: Directorio **/var/lib/tomcat7/conf/Catalina/localhost/doc.xml**

```
<Context path="/examples"
        docBase="/usr/share/tomcat7-examples/examples" allowLinking="true"/>
```

Figura 8.9: Directorio **/var/lib/tomcat7/conf/Catalina/localhost/examples.xml**

8.2. Arquitectura y ficheros de configuración

Analiza los ficheros de configuración de *Tomcat* en **ServidorLinuxXX** y relaciona su contenido con la arquitectura de *Tomcat*.

1. **Arquitectura de Tomcat.**
   Recuerda la arquitectura de *Tomcat*, Figura 8.10.

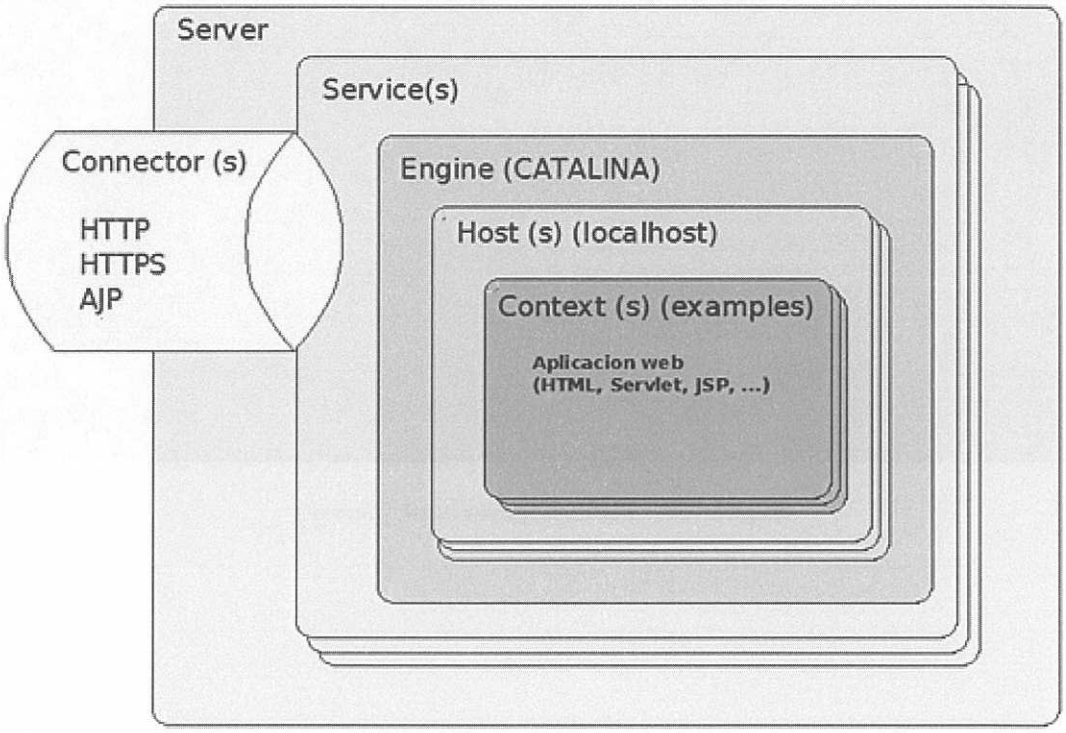


Figura 8.10: Arquitectura de *Tomcat*

2. **Estructura de directorios.**
   2.1. Inicia sesión en **ServidorLinuxXX** con un usuario con privilegios de administración.
   2.2. Consulta el directorio **/usr/share/tomcat7** (`CATALINA_HOME`).
   2.3. Consulta el directorio **/var/lib/tomcat7** (`CATALINA_BASE`). Observa que el directorio **conf** en un enlace simbólico hacía **/etc/tomcat7**.

3. **Ficheros de configuración.**
   3.1. Consulta el directorio **/var/lib/tomcat7/conf** o **/etc/tomcat7**.
   3.2. Edita el fichero **server.xml** y analiza su contenido. Observa las declaraciones:
   - `<Server>`, Figura 8.11.
   - `<Service>`, Figura 8.12.

```
<Server port="8005" shutdown="SHUTDOWN">
  <!-- Security listener. Documentation at /docs/config/listeners.html
  <Listener className="org.apache.catalina.security.SecurityListener" />
  -->
  <!--APR library loader. Documentation at /docs/apr.html -->
  <!--
  <Listener className="org.apache.catalina.core.AprLifecycleListener" SSLEngine="on" />
  -->
  <!--Initialize Jasper prior to webapps are loaded. Documentation at /docs/jasper-howto.html -->
  <Listener className="org.apache.catalina.core.JasperListener" />
  <!-- Prevent memory leaks due to use of particular java/javax APIs-->
  <Listener className="org.apache.catalina.core.JreMemoryLeakPreventionListener" />
  <Listener className="org.apache.catalina.mbeans.GlobalResourcesLifecycleListener" />
  <Listener className="org.apache.catalina.core.ThreadLocalLeakPreventionListener" />

  <!-- Global JNDI resources
       Documentation at /docs/jndi-resources-howto.html
  -->
  <GlobalNamingResources>
    <!-- Editable user database that can also be used by
         UserDatabaseRealm to authenticate users
    -->
    <Resource name="UserDatabase" auth="Container"
              type="org.apache.catalina.UserDatabase"
              description="User database that can be updated and saved"
              factory="org.apache.catalina.users.MemoryUserDatabaseFactory"
              pathname="conf/tomcat-users.xml" />
  </GlobalNamingResources>
```

Figura 8.11: Fichero **server.xml** (*Server*)

```
<Service name="Catalina">

  <!--The connectors can use a shared executor, you can define one or more named thread pools-->
  <!--
  <Executor name="tomcatThreadPool" namePrefix="catalina-exec-"
      maxThreads="150" minSpareThreads="4"/>
  -->


  <!-- A "Connector" represents an endpoint by which requests are received
       and responses are returned. Documentation at :
       Java HTTP Connector: /docs/config/http.html (blocking & non-blocking)
       Java AJP  Connector: /docs/config/ajp.html
       APR (HTTP/AJP) Connector: /docs/apr.html
       Define a non-SSL HTTP/1.1 Connector on port 8080
  -->
  <Connector port="8080" protocol="HTTP/1.1"
             connectionTimeout="20000"
             URIEncoding="UTF-8"
             redirectPort="8443" />
```

Figura 8.12: Fichero **server.xml** (*Service*)

- <Connectors>, Figura 8.13.

```
<Connector port="8080" protocol="HTTP/1.1"
           connectionTimeout="20000"
           URIEncoding="UTF-8"
           redirectPort="8443" />
```

Figura 8.13: Fichero **server.xml** (*Connectors*)

- <Engine>, Figura 8.14.

```
<Engine name="Catalina" defaultHost="localhost">

  <!--For clustering, please take a look at documentation at:
      /docs/cluster-howto.html  (simple how to)
      /docs/config/cluster.html (reference documentation) -->
  <!--
  <Cluster className="org.apache.catalina.ha.tcp.SimpleTcpCluster"/>
  -->

  <!-- Use the LockOutRealm to prevent attempts to guess user passwords
       via a brute-force attack -->
  <Realm className="org.apache.catalina.realm.LockOutRealm">
    <!-- This Realm uses the UserDatabase configured in the global JNDI
         resources under the key "UserDatabase".  Any edits
         that are performed against this UserDatabase are immediately
         available for use by the Realm.  -->
    <Realm className="org.apache.catalina.realm.UserDatabaseRealm"
           resourceName="UserDatabase"/>
  </Realm>

  <Host name="localhost"  appBase="webapps"
        unpackWARs="true" autoDeploy="true">
```

Figura 8.14: Fichero **server.xml** (*Engine*)

- <Host>, Figura 8.15.

```
    <Host name="localhost"  appBase="webapps"
          unpackWARs="true" autoDeploy="true">

      <!-- SingleSignOn valve, share authentication between web applications
           Documentation at: /docs/config/valve.html -->
      <!--
      <Valve className="org.apache.catalina.authenticator.SingleSignOn" />
      -->

      <!-- Access log processes all example.
           Documentation at: /docs/config/valve.html
           Note: The pattern used is equivalent to using pattern="common" -->
      <Valve className="org.apache.catalina.valves.AccessLogValve" directory="logs"
             prefix="localhost_access_log." suffix=".txt"
             pattern="%h %l %u %t &quot;%r&quot; %s %b" />

    </Host>
  </Engine>
 </Service>
</Server>
```

Figura 8.15: Fichero **server.xml** (*Host*)

3.3. Consulta el directorio **/var/lib/tomcat7/conf/Catalina/localhost**, Figura 8.16.

3.4. Consulta el contenido directorio **/var/lib/tomcat7/conf/Catalina/localhost/examples** y el fichero **examples.xml**, Figura 8.16.

3.5. Consulta el contenido directorio **/user/share/tomcat7-examples/examples**, Figura 8.16.

3.6. Inicia un navegador y accede a la aplicación **examples**, Figura 8.17.

```
alumno@ServidorLinux01:/var/lib/tomcat7/conf/Catalina/localhost$ ls -l
total 8
-rw-r--r-- 1 root root 922 feb 21  2014 docs.xml
-rw-r--r-- 1 root root  97 feb 21  2014 examples.xml
alumno@ServidorLinux01:/var/lib/tomcat7/conf/Catalina/localhost$ cat examples.xml
<Context path="/examples"
        docBase="/usr/share/tomcat7-examples/examples" allowLinking="true"/>
alumno@ServidorLinux01:/var/lib/tomcat7/conf/Catalina/localhost$ cd /usr/share/tomcat7-examples/
alumno@ServidorLinux01:/usr/share/tomcat7-examples$ ls
examples
alumno@ServidorLinux01:/usr/share/tomcat7-examples$ cd examples/
alumno@ServidorLinux01:/usr/share/tomcat7-examples/examples$ ls
index.html  jsp  servlets  WEB-INF  websocket  websocket-deprecated
alumno@ServidorLinux01:/usr/share/tomcat7-examples/examples$
```

Figura 8.16: Directorio **/var/lib/tomcat7/conf/Catalina/localhost**

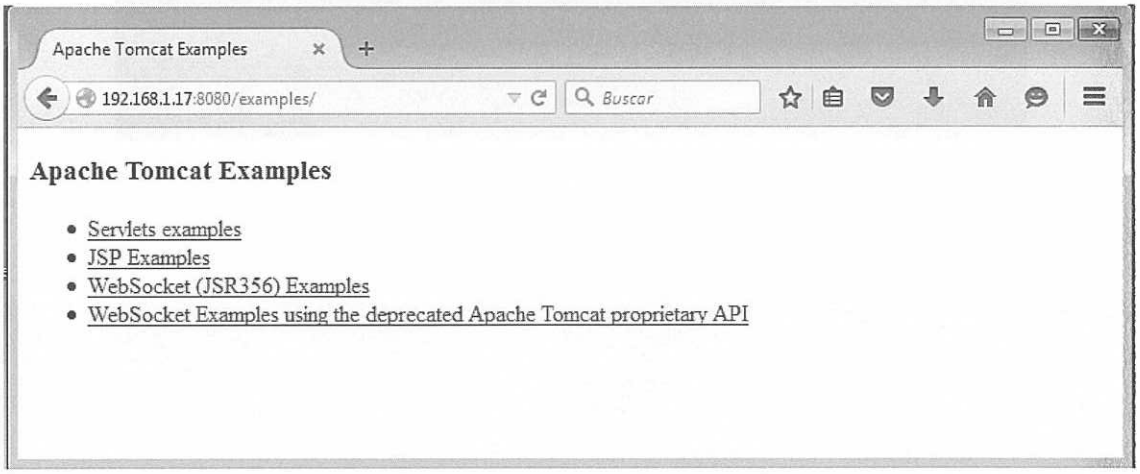


Figura 8.17: Acceso a la aplicación examples

8.3. Despliegue manual de aplicaciones web

Crea la aplicación web **curso** compuesta por un *Servlet* y un JSP en **DesarrolloW7XX** y prueba a desplegar la aplicación manualmente en el servidor *Tomcat* instalado en **ServidorLinuxXX**. Las especificaciones de la aplicación serán proporcionadas por el profesor.

1. **Desarrollo de la aplicación curso.**

   1.1. Inicia sesión en **DesarrolloW7XX**.

   1.2. Inicia *Eclipse* y selecciona el espacio de trabajo por defecto **C:\Users\alumno\workspace**.

   1.3. Crea un proyecto web dinámico en **File, New, Dynamic Web Project**.

   1.4. Crea los siguientes componentes:

   - **index.html**, Figura 8.18.
   - Servlet: **HolaServlet.java**, Figura 8.19.
   - JSP: **buenas.jsp**, Figura 8.20.
   - Descriptor de despliegue: **web.xml**, Figura 8.21.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Insert title here</title>
</head>
<body>
    <h1>Curso de despliegue de aplicaciones web</h1>
    <ul>
        <li><h2>
                <a href="hola">Hola (Servlet)</a>
            </h2>
        <li><h2>
                <a href="buenas.jsp">Buenas (JSP)</a>
            </h2>
    </ul>
</body>
</html>
```

Figura 8.18: **index.html**

```
package net.daw01.curso;

import java.io.*;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class HolaServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    public HolaServlet() {
        super();
    }

    protected void doGet(HttpServletRequest request,
            HttpServletResponse response) throws ServletException, IOException {

        response.setContentType("text/html");
        PrintWriter out = response.getWriter();

        try {
            out.println("<html>");
            out.println("<head><title>Hola, alumno</title></head>");
            out.println("<body>");
            out.println("<h1>Hola, alumno!</h1>");
            out.println("<p>URI: " + request.getRequestURI() + "</p>");
            out.println("<p>Protocolo: " + request.getProtocol() + "</p>");
            out.println("<p>Ip del cliente: " + request.getRemoteAddr()
                    + "</p>");
            out.println("</body></html>");
        } finally {
            out.close();
        }
    }
}
```

Figura 8.19: **HolaServlet.java**

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>

<%@
    page import="java.util.Calendar"%>

<!DOCTYPE html
    PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Insert title here</title>
</head>
<body>
    <div class="content">
        <b>Bienvenido al curso</b>

        <p>
            Hoy es
            <%=Calendar.getInstance().getTime()%>
        </p>
        <%
            String saludo;
            int horaDelDia = Calendar.getInstance().get(Calendar.HOUR_OF_DAY);
            if (horaDelDia < 12) {
                saludo = "Buenos días!";
            } else if (horaDelDia >= 12 && horaDelDia < 21) {
                saludo = "Buenas tardes!";
            } else {
                saludo = "Buenas noches!";
            }
        %>
        <p><%=saludo%></p>
    </div>
</body>
</html>
```

Figura 8.20: **buenas.jsp**

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="http://java.sun.com/xml/ns/javaee"
    xmlns:web="http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
        http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
    id="WebApp_ID" version="3.0">

    <display-name>curso</display-name>
    <welcome-file-list>
        <welcome-file>index.html</welcome-file>
    </welcome-file-list>

    <servlet>
        <servlet-name>HolaServlet</servlet-name>
        <servlet-class>net.daw01.curso.HolaServlet</servlet-class>
    </servlet>

    <servlet-mapping>
        <servlet-name>HolaServlet</servlet-name>
        <url-pattern>/hola</url-pattern>
    </servlet-mapping>

</web-app>
```

Figura 8.21: **web.xml**

1.5. Observa la estructura del proyecto en *Eclipse*, Figura 8.22.

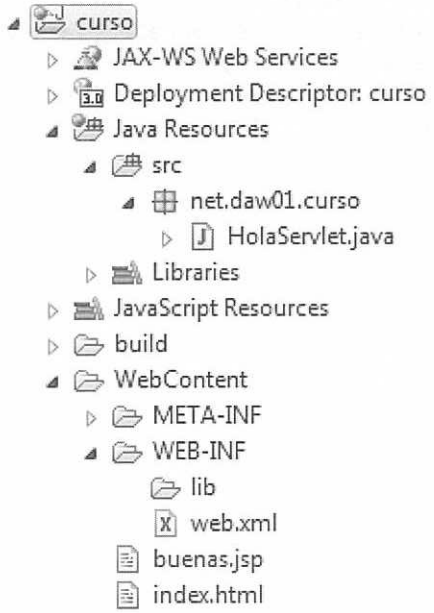


Figura 8.22: Proyecto **curso** en *Eclipse*

1.6. Observa la estructura de directorio y ficheros de la aplicación en **C:\Users\alumno\workspace\curso**, Figura 8.23.

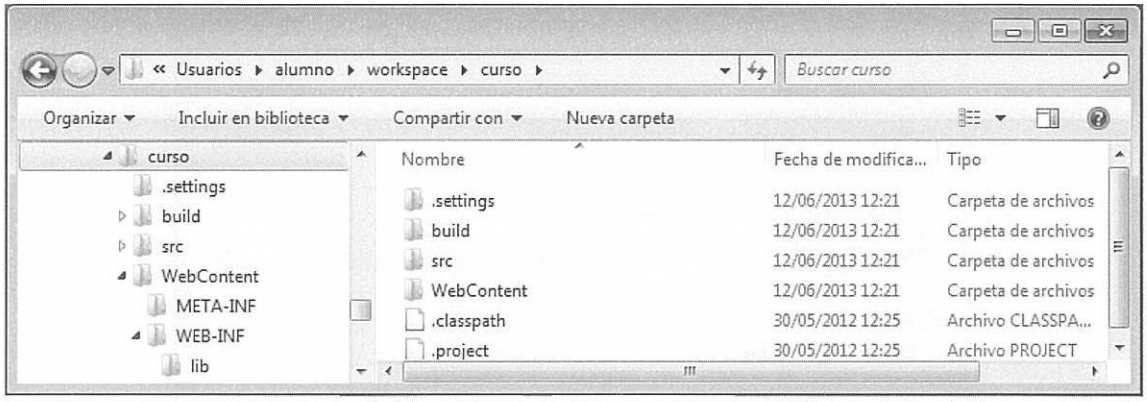


Figura 8.23: Directorio **C:\Users\alumno\workspace\curso**

1.7. Prueba la aplicación el servidor *Tomcat* local, Figura 8.24.

Figura 8.24: Prueba de la aplicación en local

2. **Exportar la aplicación a un fichero .war.**

2.1. Sobre el nombre del proyecto haz clic con el botón derecho del ratón y selecciona ***Export, WAR File***.

2.2. Indica el escritorio del usuario y selecciona que se optimice para *Apache Tomcat 7*, Figura 8.25.

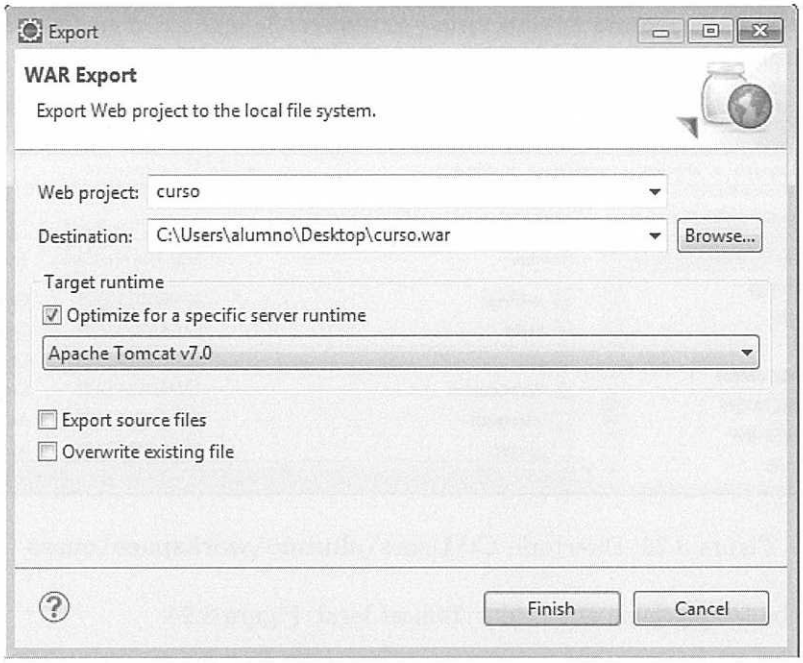


Figura 8.25: Exportar a un fichero WAR

2.3. Accede al escritorio y descomprime el fichero **curso.war**. Observa su contenido, Figuras 8.26 y 8.27.

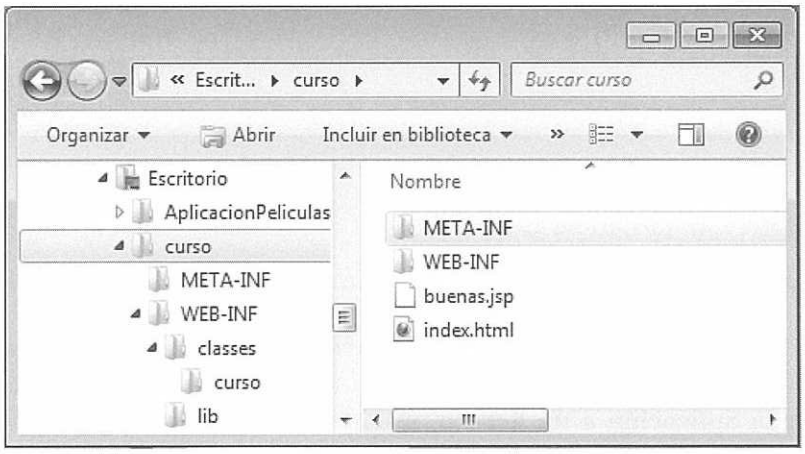


Figura 8.26: Contenido del fichero WAR

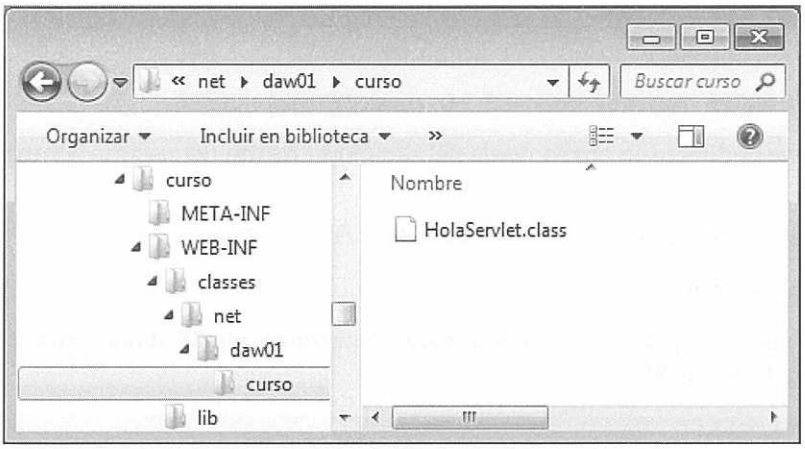


Figura 8.27: Contenido del fichero WAR

3. **Despliegue de aplicación.**

   3.1. Inicia el cliente *Filezilla* y establece una conexión al servidor como usuario **alumno**.

   3.2. Sube el directorio **curso** (el que se ha creado al descomprimir el war) al directorio **home** del usuario **alumno**, Figura 8.28.

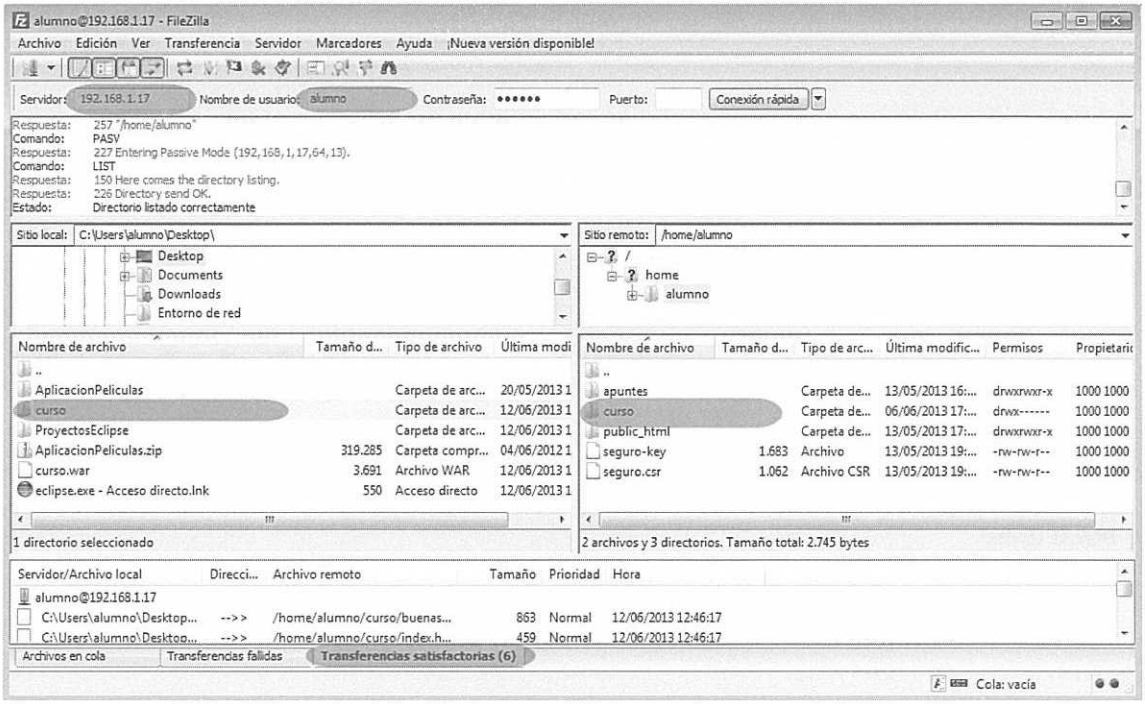


Figura 8.28: Conexión FTP como usuario alumno

   3.3. Inicia el cliente SSH *Putty* y establece una conexión SSH al servidor como usuario **alumno**.

3.4. Cambia el propietario y el grupo del directorio **/home/alumno/curso** y todo su contendido al usuario **tomcat7** y grupo **tomcat7**.

```
sudo chown tomcat7:tomcat7 -R /home/alumno/curso
```

3.5. Mueve el directorio **curso** desde del directorio **home** del usuario **alumno** al directorio **/var/lib/tomcat7/webapps**.

```
sudo mv /home/alumno/curso /var/lib/tomcat7/webapps
```

4. **Acceso a la aplicación**

4.1. Abre un navegador y accede a `http://servidorlinuxXX.dawXX.net:8080/curso`, Figuras 8.29 y 8.30.

Figura 8.29: Conexión a *curso*

Figura 8.30: Conexión a *curso*

5. **Despliegue de aplicaciones usando ficheros .war.**

   5.1. En **ServidorLinuxXX** borra el directorio **/var/lib/tomcat7/webapps/curso**.

   ```
   sudo rm -R /var/lib/tomcat7/webapps/curso
   ```

   5.2. En **DesarrolloW7XX** Inicia el cliente *Filezilla* y establece una conexión al servidor como usuario **alumno**.

   5.3. Sube el fichero **curso.war** al directorio **home** del usuario **alumno**, Figura 8.31.

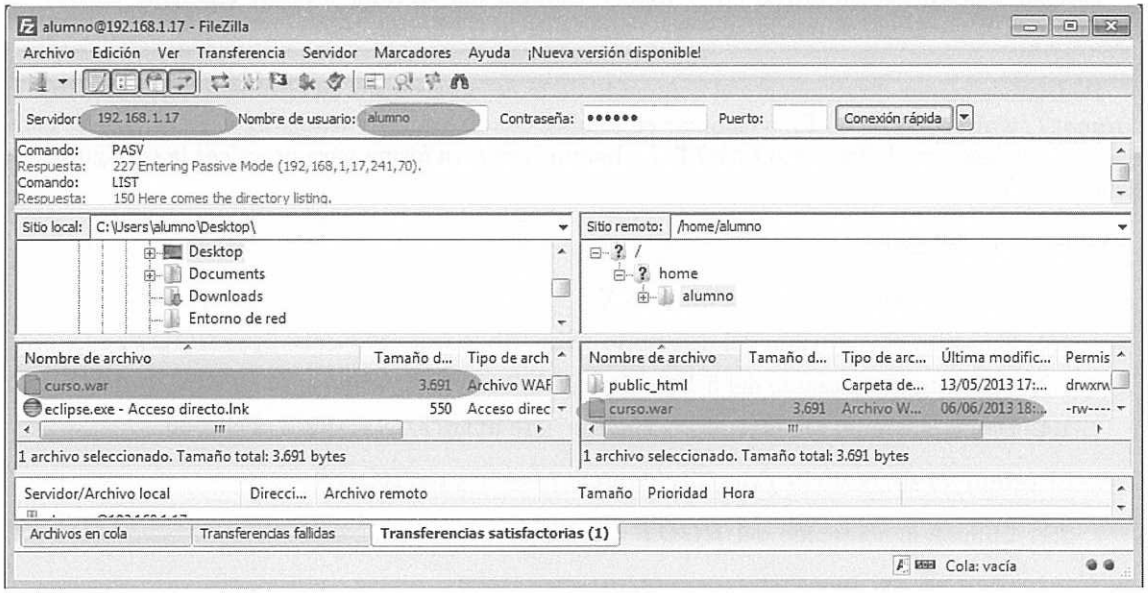


Figura 8.31: Conexión FTP como usuario root

   5.4. Inicia el cliente SSH *Putty* y establece una conexión SSH al servidor como usuario **alumno**.

   5.5. Cambia el propietario y el grupo del fichero **curso.war** al usuario **tomcat7** y grupo **tomcat7**.

   ```
   sudo chown tomcat7:tomcat7 /home/alumno/curso.war
   ```

   5.6. Mueve el fichero **curso.war** desde del directorio **home** del usuario **alumno** al directorio **/var/lib/tomcat7/webapps**.

   ```
   sudo mv /home/alumno/curso.war /var/lib/tomcat7/webapps
   ```

   5.7. En **ServidorLinuxXX** reinicia *Tomcat*.

   ```
   sudo service tomcat7 stop
   sudo service tomcat7 start
   ```

   5.8. En **ServidorLinuxXX** accede al directorio **/var/lib/tomcat7/webapps/** y observa que *Tomcat* ha descomprimido automáticamente el fichero **curso.war** en un carpeta con el nombre **curso**, Figura 8.32.

   5.9. En **DesarrolloW7XX** abre un navegador y accede a `http://servidorlinuxXX.dawXX.net:8080/curso`.

```
alumno@ServidorLinux01:/var/lib/tomcat7/webapps$ ls
curso  curso.war  ROOT
alumno@ServidorLinux01:/var/lib/tomcat7/webapps$
```

Figura 8.32: Directorio **/var/lib/tomcat7/webapps/**

8.4. Despliegue de aplicaciones en el *ROOT Context*

Despliega la aplicación **curso** en el *ROOT Context* del servidor *Tomcat* instalado en **ServidorLinuxXX** para que se pueda acceder a través de la URL `192.168.1.X7:8080`.

Podríamos copiar directamente el contenido de la carpeta **curso** en el directorio **/var/lib/tomcat7/webapps/ROOT/** o cambiar el nombre del directorio **curso.war** a **ROOT.war** para que se desplegase en el directorio **ROOT**. Lo haremos de otra forma para practicar la configuración de los *Contexts*.

1. ***ROOT Context***.
   1.1. Inicia sesión en **ServidorLinuxXX**.
   1.2. Consulta el contenido del directorio **/var/lib/tomcat7/webapps/ROOT/**.
   1.3. Consulta el contenido del fichero **/var/lib/tomcat7/webapps/ROOT/index.html**.
   1.4. Desde el equipo **DesarrolloW7XX** accede `http://192.168.1.X7:8080`.
2. **Eliminar el directorio /var/lib/tomcat7/webapps/ROOT**.
   2.1. Elimina el contenido del ***ROOT Context***.

   ```
   sudo rm -R /var/lib/tomcat7/webapps/ROOT
   ```

   2.2. Reinicia *Tomcat*.

   ```
   sudo service tomcat7 stop
   sudo service tomcat7 start
   ```

3. **Configurar curso para que se despliegue en el *ROOT Context***.
   3.1. Mueve el directorio **curso** a **/usr/share/** por ejemplo (podrías elegir otra ubicación).

   ```
   sudo mv /var/lib/tomcat7/webapps/curso /usr/share
   ```

   3.2. Elimina el fichero **/var/lib/tomcat7/webapps/curso.war**

   ```
   sudo rm /var/lib/tomcat7/webapps/curso.war
   ```

   3.3. Crea el fichero **/var/lib/tomcat7/conf/Catalina/localhost/ROOT.xml** y configura el atributo **docBase** como se muestra en la Figura 8.33.

```
<Context docBase="/usr/share/curso"/>
```

Figura 8.33: Fichero **/var/lib/tomcat7/conf/Catalina/localhost/ROOT.xml**

3.4. Reinicia *Tomcat*.

```
sudo service tomcat7 stop
sudo service tomcat7 start
```

4. Acceso a la aplicación

4.1. Abre un navegador y accede a `http://servidorlinuxXX.dawXX.net:8080`, Figura 8.34.

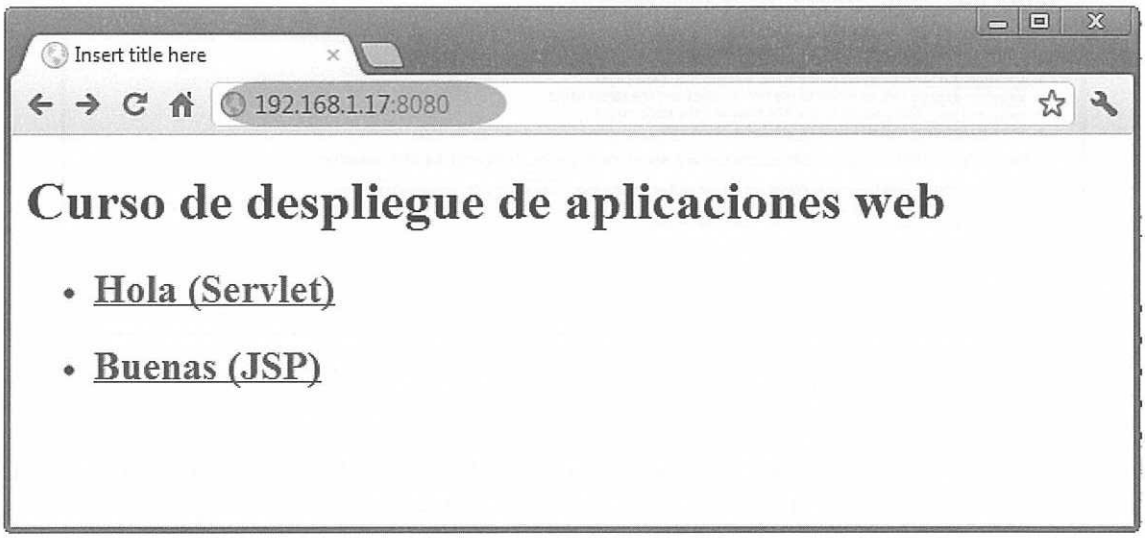


Figura 8.34: Conexión al directorio raíz

8.5. Instalación de *Tomcat Web Manager*

Instala *Tomcat Web Manager* en **ServidorLinuxXX** y configura el usuario **tomcat** con el rol **manager-gui** para que pueda utilizar el acceso web y consultar el estado del servidor.

1. Instalación.

1.1. Inicia sesión en **ServidorLinuxXX** con un usuario con privilegios de administración.

1.2. Instala el paquete **tomcat7-admin**.

```
sudo apt-get update
sudo apt-get install tomcat7-admin
```

1.3. Consulta el contenido del directorio **/var/lib/tomcat7/conf/Catalina/localhost/** y observa que se ha creado el fichero **manager.xml**.

2. Acceso.

2.1. Desde el equipo **DesarrolloW7XX** accede `http://192.168.1.X7:8080/manager` y observa que no puedes acceder. Pincha en **Cancelar** y observa las instrucciones, Figura 8.35.

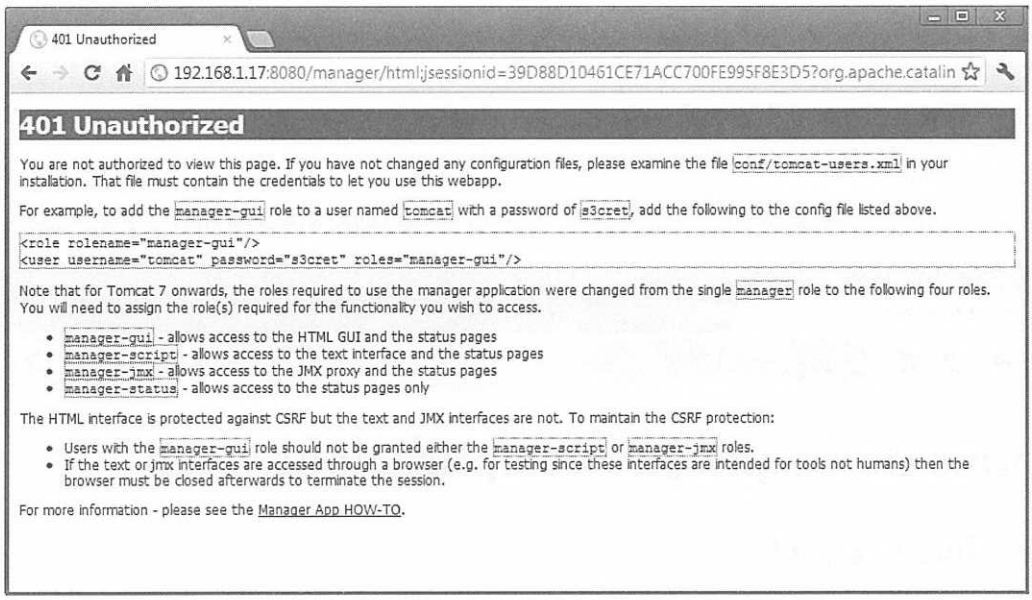


Figura 8.35: Acceso prohibido al *manager*

2.2. En **ServidorLinuxXX** edita el fichero **/var/lib/tomcat7/conf/tomcat-users.xml** y realiza la configuración que se muestra en la Figura 8.36.

```
<!--
  <role rolename="tomcat"/>
  <role rolename="role1"/>
  <user username="tomcat" password="tomcat" roles="tomcat"/>
  <user username="both" password="tomcat" roles="tomcat,role1"/>
  <user username="role1" password="tomcat" roles="role1"/>
-->
  <role rolename="manager-gui"/>
  <user username="tomcat" password="despliegue" roles="manager-gui"/>

</tomcat-users>
```

Figura 8.36: Fichero **/var/lib/tomcat7/conf/tomcat-users.xml**

2.3. Reinicia *Tomcat*.

2.4. Desde el equipo **DesarrolloW7XX** accede `http://192.168.1.X7:8080/manager/html` e introduce el usuario creado, Figuras 8.37 y 8.38.

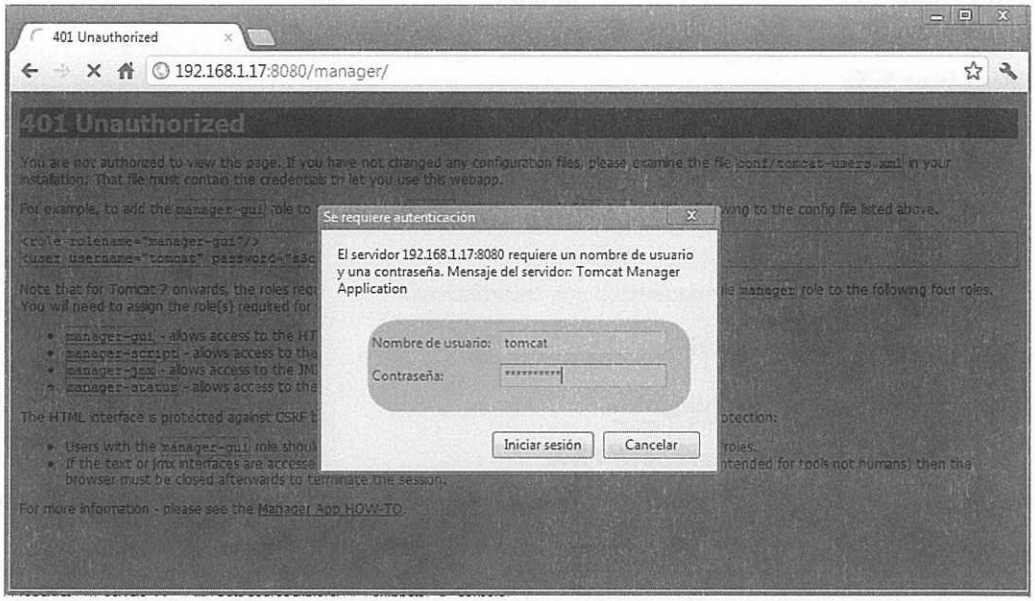


Figura 8.37: Acceso a *Tomcat Web Manager*

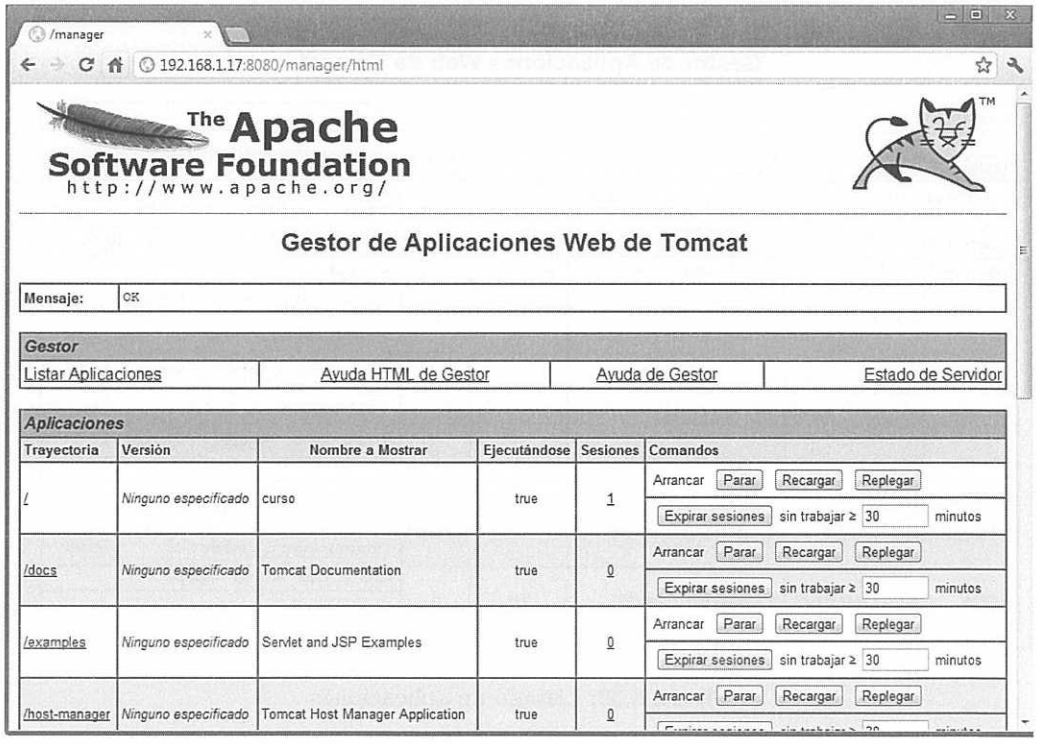


Figura 8.38: Acceso a *Tomcat Web Manager*

8.6. Administración con *Tomcat Web Manager*

Utiliza *Tomcat Web Manager* para realizar las siguientes tareas sobre el servidor *Tomcat* de **ServidorLinuxXX**:

- Listar las aplicaciones desplegadas.
- Iniciar/para aplicaciones.
- Desplegar aplicaciones.
- Listar la sesiones activas y consultar sus estadísticas.
- Consultar el estado del servidor.

1. **Listar aplicaciones desplegadas.**

   1.1. Inicia sesión en **DesarrolloW7XX**.

   1.2. Accede `http://192.168.1.X7:8080/manager/html` e inicia sesión con el usuario **tomcat**.

   1.3. Observa las aplicaciones desplegadas la información que *Tomcat manager* ofrece sobre ellas (trayectoria, versión, nombres, si están iniciadas o paradas y sus sesiones activas), Figura 8.39.

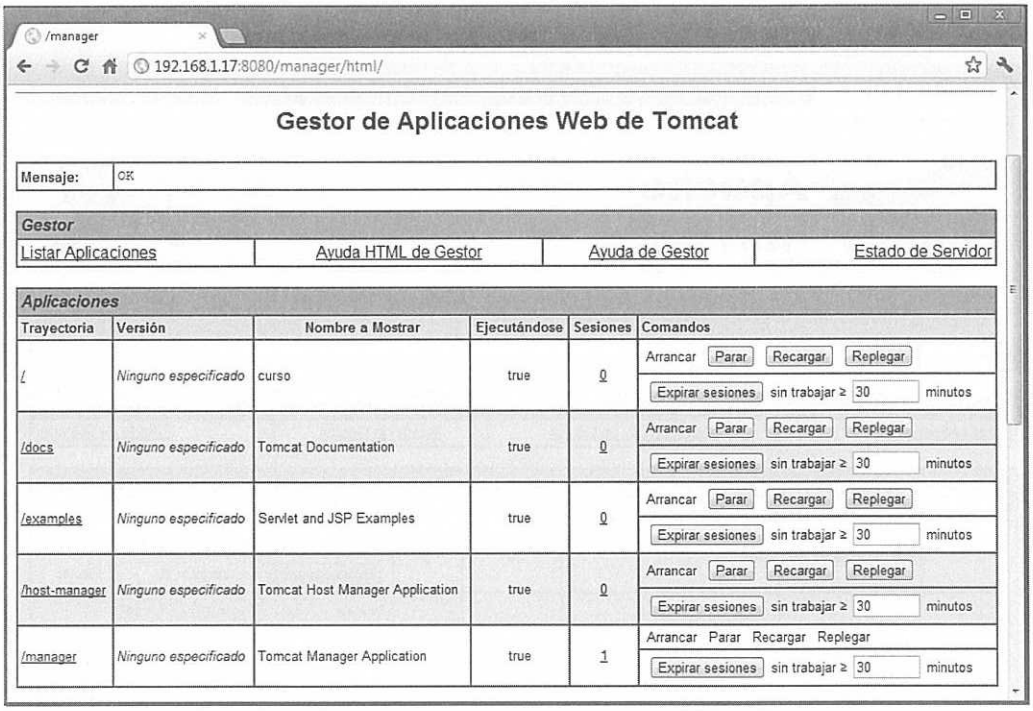


Figura 8.39: Listado de aplicaciones

2. **Iniciar/para aplicaciones.**

   2.1. Para la aplicación **examples**.

   2.2. Abre otra ventana del navegador, accede `http://192.168.1.X7:8080/examples` y observa que no es posible acceder, Figura 8.40.

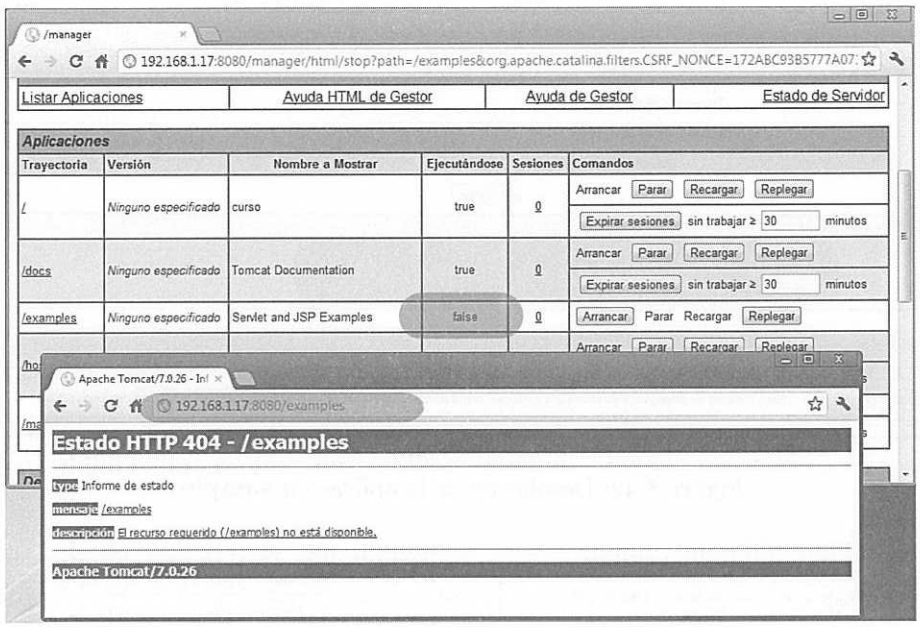


Figura 8.40: Aplicación examples parada

   2.3. Inicia la aplicación **examples** y prueba que puedes acceder a ella.

   2.4. Es importante observar el mensaje que ofrece *Tomcat Manger* cada vez que realizamos una acción para detectar posibles errores, Figura 8.41.

Gestor de Aplicaciones Web de Tomcat

Mensaje:	`OK - Arrancada aplicación en trayectoria de contexto /examples`

Figura 8.41: Mensaje de *Tomcat Web Manager*

3. **Desplegar aplicaciones.**

   3.1. Abre otra ventana del navegador, accede `http://tomcat.apache.org/tomcat-7.0-doc/appdev/sample/` y descarga la aplicación **sample.war**.

   3.2. En *Tomcat Web Manger* despliega la aplicación **sample** usando el fichero **sample.war** (**Desplegar, Archivo WAR a desplegar, Seleccionar Archivo**), Figuras 8.42 y 8.43.

   3.3. Despliega la aplicación **Calendar.war** en el servidor y pruébala, Figuras 8.44.

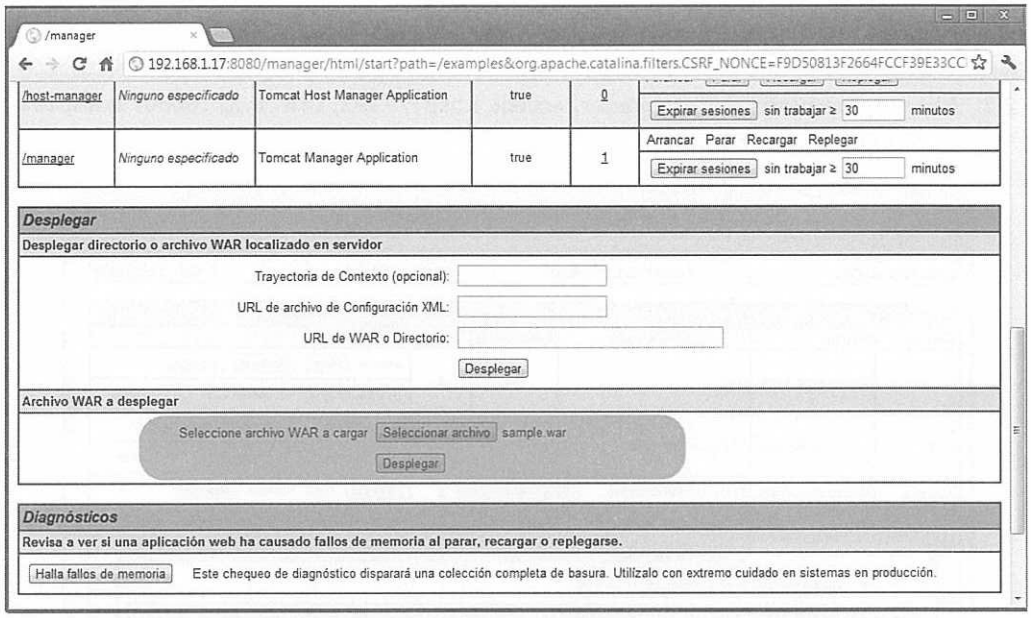


Figura 8.42: Despliegue de la aplicación **sample**

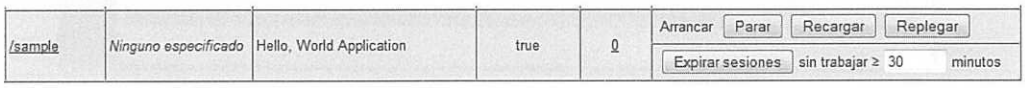


Figura 8.43: Despliegue de la aplicación **sample**

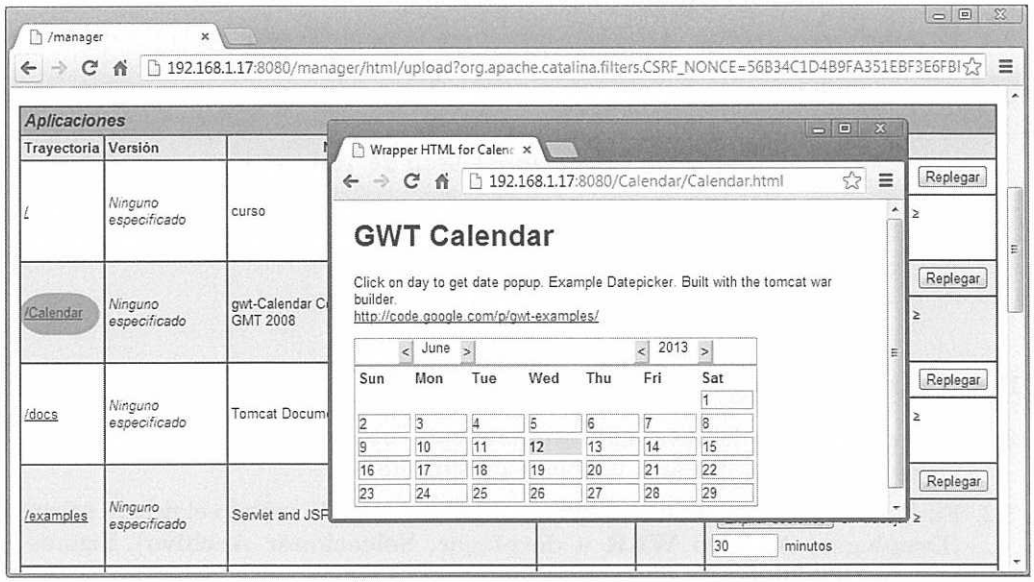


Figura 8.44: Despliegue de la aplicación **Calendar**

4. <u>Sesiones activas</u>.

   4.1. Abre otra ventana del navegador, y accede `http://192.168.1.X7:8080/examples`.

   4.2. Accede a *Servlets Examples* y ejecuta el ejemplo *Sessions*, Figura 8.45.

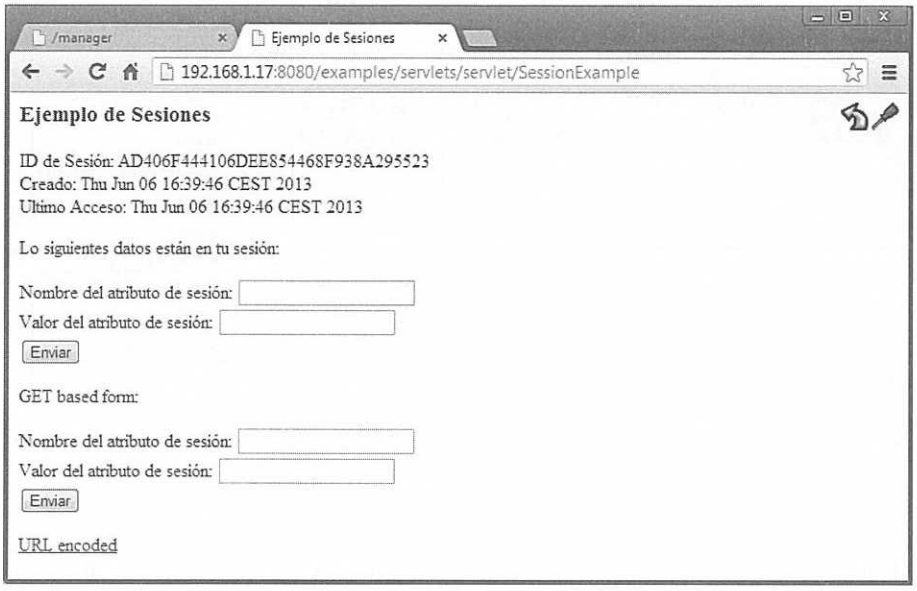


Figura 8.45: Ejemplo *Servlet*, *Sessions*

   4.3. Crea dos atributos de sesión (por ejemplo nombre y apellidos) y asígnales valor, Figura 8.46.

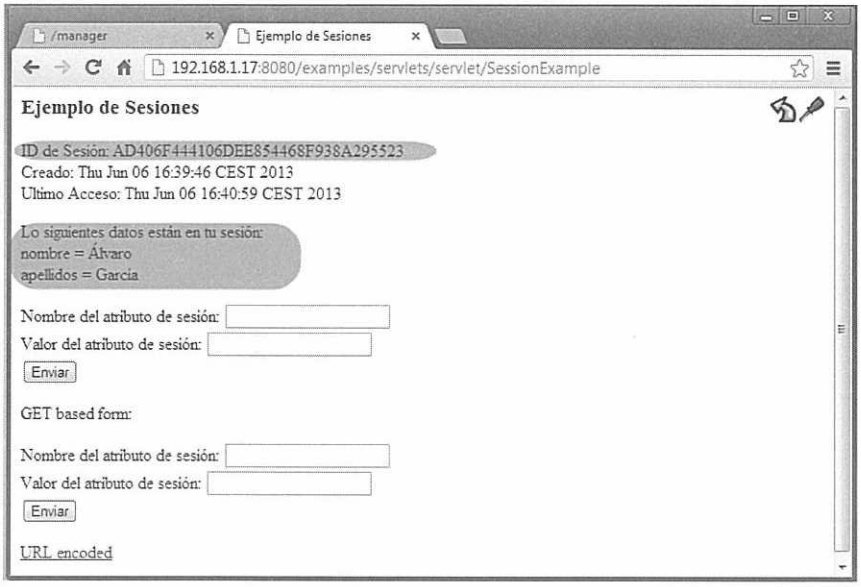


Figura 8.46: Atributos de sesión

4.4. Accede a la ventaba del navegador con el *manager*, refresca la lista de aplicaciones y observa que hay una sesión activa para la aplicación *Examples*, Figura 8.47.

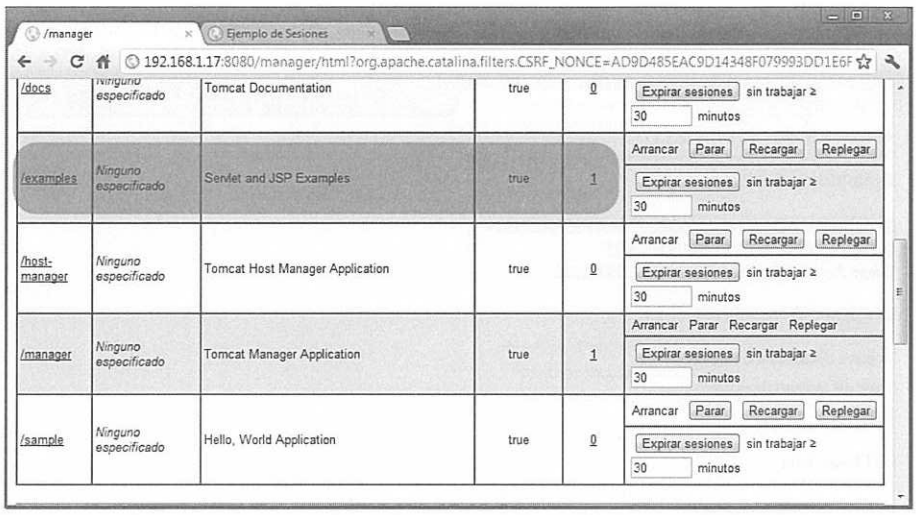


Figura 8.47: Sesiones activas

4.5. Pincha sobre el número 1 y observa la información de la sesión, Figura 8.48. Observa que puedes eliminar sesiones activas.

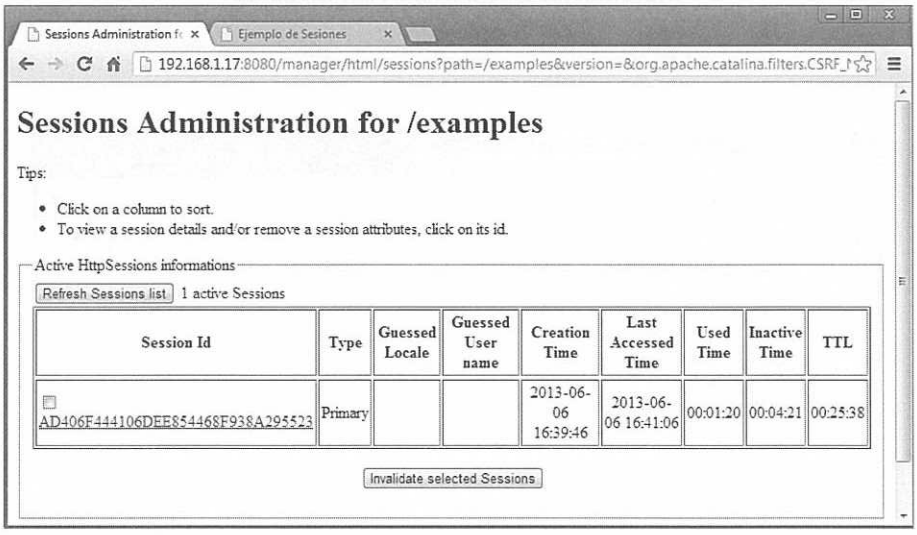


Figura 8.48: Información de la sesión

4.6. Pincha sobre el ID de la sesión y observa los atributos de la misma, Figura 8.49

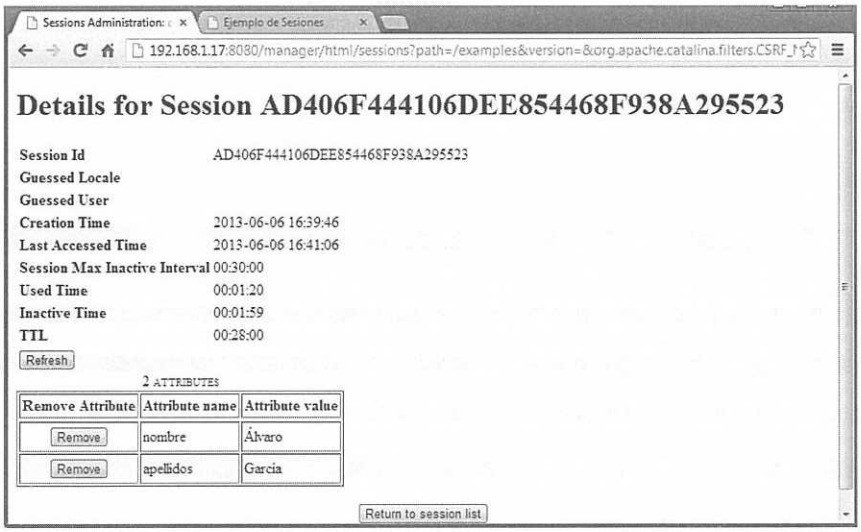


Figura 8.49: Atributos de la sesión

4.7. Observa que puedes borrar los atributos de la sesión.

5. <u>Consultar el estado del servidor</u>.

5.1. Accede a la ventana del navegador con el *manager* y pincha en **Estado del servidor**, Figura 8.50.

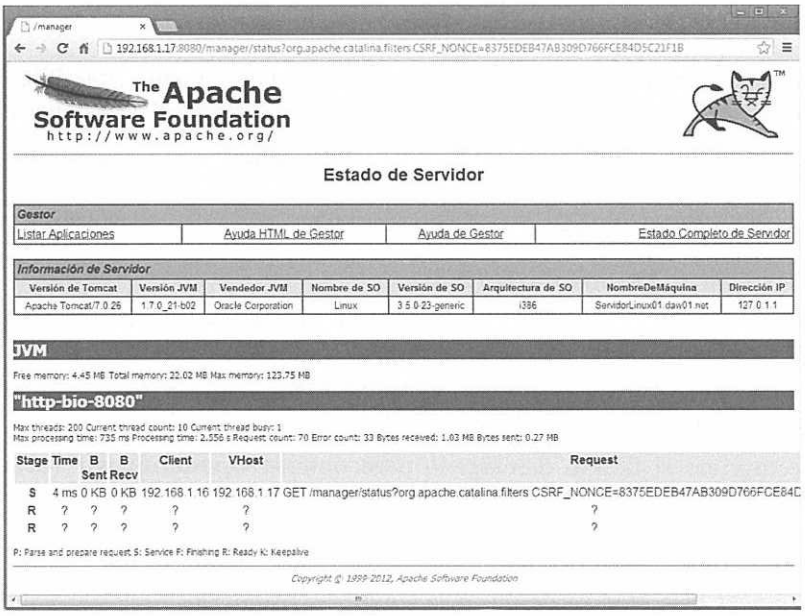


Figura 8.50: Estado del servidor

5.2. Pincha en **Estado completo del servidor**, Figura 8.51.

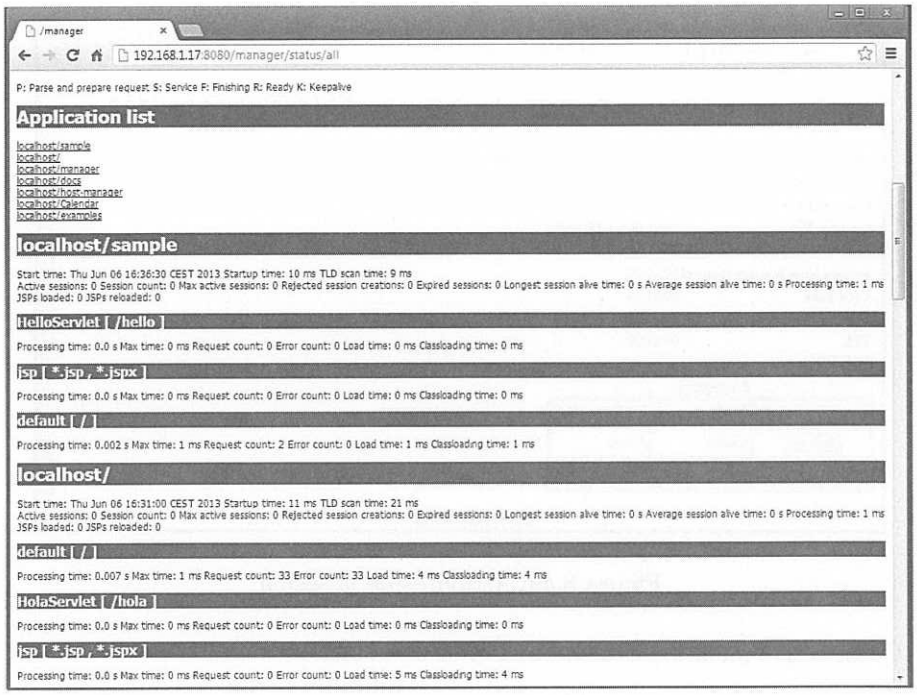


Figura 8.51: Estado completo del servidor

8.7. Despliegue de *OpenCms*

OpenCms (`www.opencms.org/`) es un sistema de gestión de contenido de código abierto basado en ***Java*** y en tecnología XML. Es distribuido por la empresa *Alkacon Software* bajo licencia LGPL.

Despliega e instala *OpenCms* en el equipo **ServidorLinuxXX**.

1. Descagar *OpenCms*.
   1.1. Inicia sesión en **DesarrolloW7XX**.
   1.2. Accede a `http://www.opencms.org/en/download/opencms.html` y descarga la distribución en código binario.
   1.3. Accede a `http://www.opencms.org/en/development/installation/server.html` y consulta la documentación sobre la instalación.
   1.4. Descomprime el fichero descargado para obtener **opencms.war** (observa que existe **install.html** con las instrucciones de instalación).
2. Desplegar OpenCms.
   2.1. En *Tomcat Web Manger* despliega la aplicación **opencms** usando el fichero **opencms.war** y observa que no se puede subir el archivo porque existe una limitación de tamaño, Figura 8.52. Podemos subirlo por FTP al servidor o aumentar el tamaño máximo de subida en el ***manager***.

Figura 8.52: Limitación de tamaño

2.2. En **ServidorLinuxXX** edita el descriptor de despliegue (web.xml) del ***manager*** **/usr/share/tomcat7-admin/manager/WEB-INF/web.xml** y aumenta el tamaño de subida de ficheros .war a 200MB, Figura 8.53.

```
<servlet>
  <servlet-name>HTMLManager</servlet-name>
  <servlet-class>org.apache.catalina.manager.HTMLManagerServlet</servlet-clas$
  <init-param>
    <param-name>debug</param-name>
    <param-value>2</param-value>
  </init-param>
  <!-- Uncomment this to show proxy sessions from the Backup manager in the
       sessions list for an application
  <init-param>
    <param-name>showProxySessions</param-name>
    <param-value>true</param-value>
  </init-param>
  -->
  <multipart-config>
    <!-- 50MB max -->
    <max-file-size>209715200</max-file-size>
    <max-request-size>209715200</max-request-size>
    <file-size-threshold>0</file-size-threshold>
  </multipart-config>
</servlet>
```

Figura 8.53: Fichero **/usr/share/tomcat7-admin/manager/WEB-INF/web.xml**

2.3. Reinicia Tomcat.

2.4. En *Tomcat Web Manger* despliega la aplicación **opencms** usando el fichero **opencms.war**, Figura 8.54.

3. **Instalación y configuración inicial de *OpenCms*.**

3.1. Accede `http://192.168.1.X7:8080/opencms/setup`, Figura 8.55.

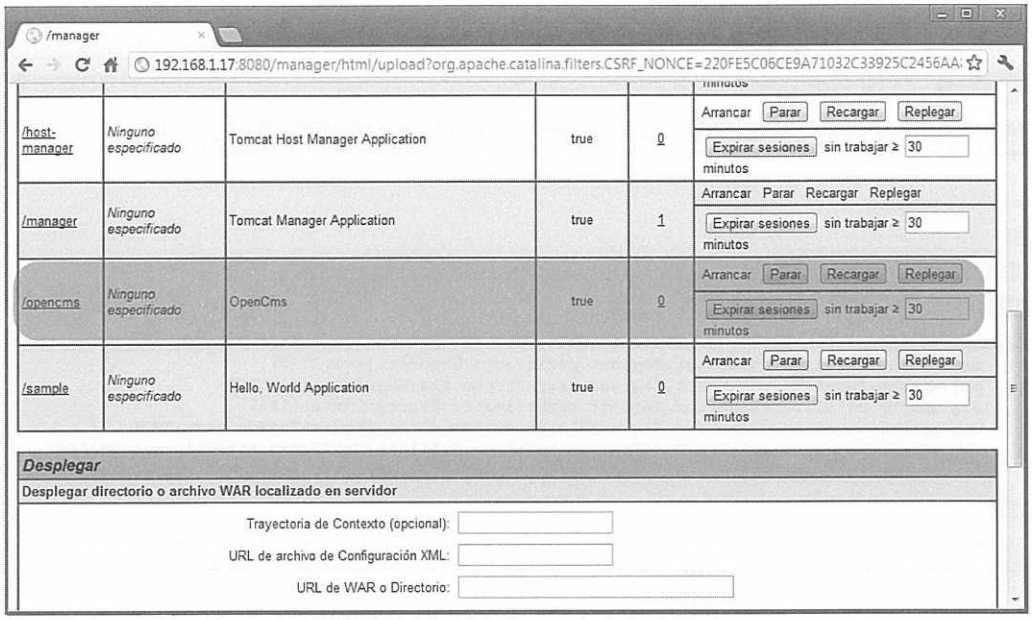


Figura 8.54: *OpenCms* desplegado

3.2. Sigue los pasos de la instalación. Algunos pasos se muestran en las Figuras 8.56, 8.57, 8.58 y 8.59.

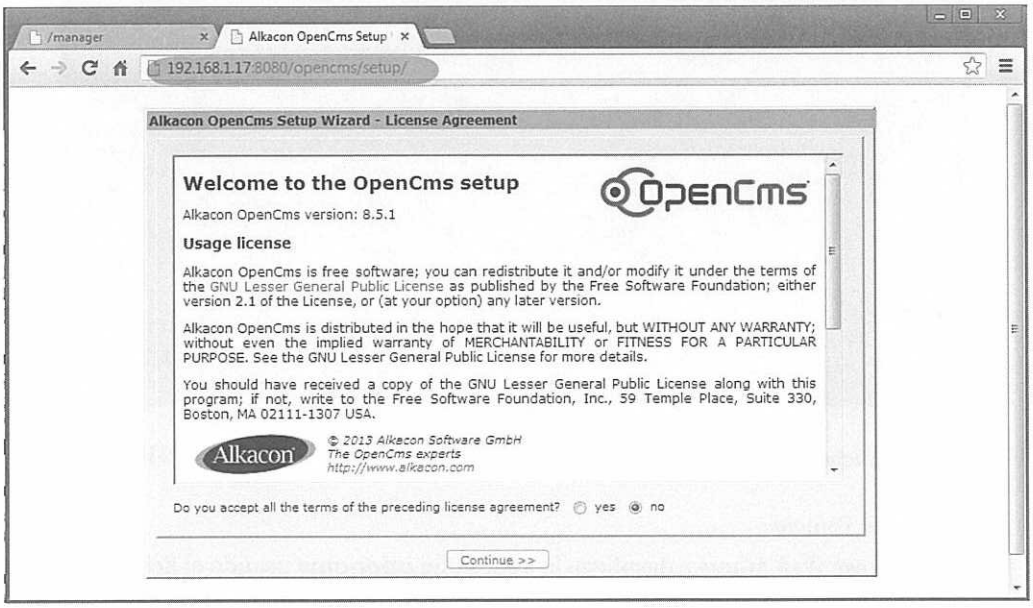


Figura 8.55: Configuración de **OpenCms**

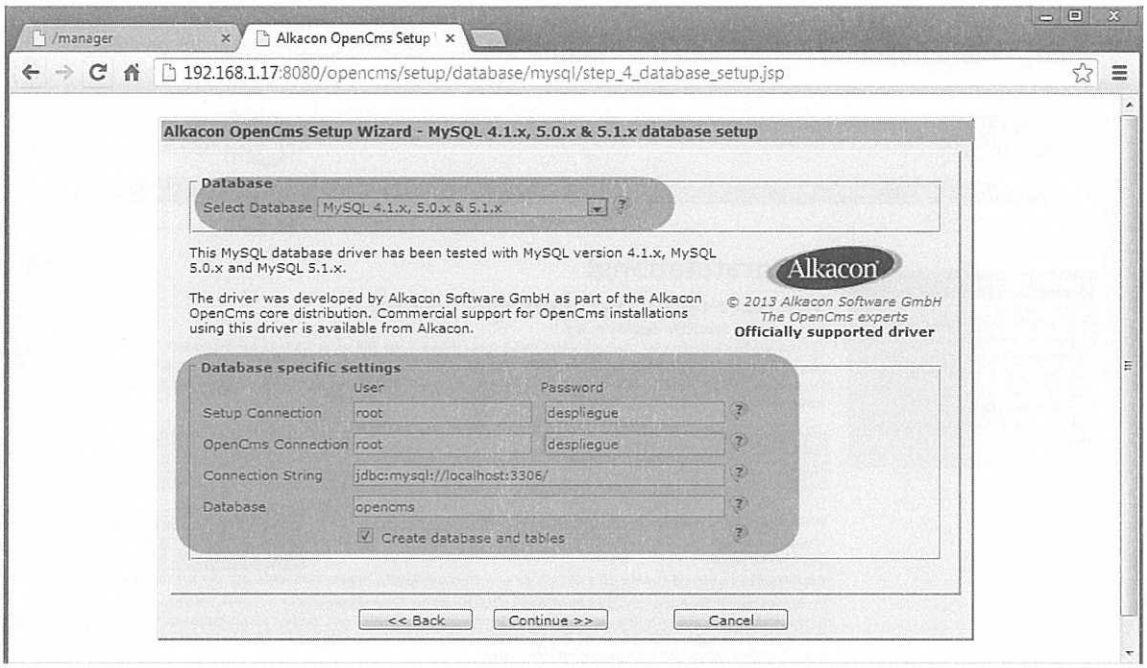


Figura 8.56: Configuración de la base de datos

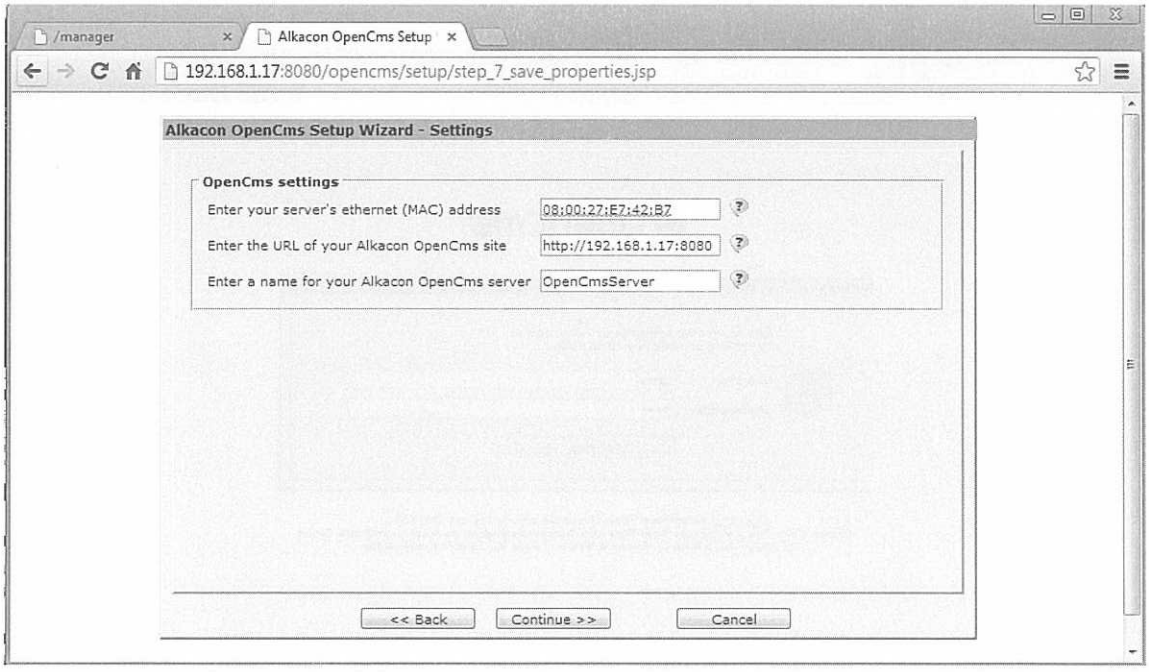


Figura 8.57: Instalación

Figura 8.58: *OpenCms* instalado

Figura 8.59: Acceso *OpenCms*

8.8. Despliegue de aplicaciones con *Apache Ant*

Realiza las siguientes configuraciones:

- En **ServidorLinuxXX** habilita *Tomcat Web Manager* el usuario **tomcatscript** con el rol **manager-script** para que pueda acceder al modo texto.
- En **DesarrolloLinuxXX** configura *Ant* en *Eclipse* para que pueda utilizar las librerías de *Tomcat* que definen las tareas que permiten interactuar con el modo *text* del *manager*.
- Crea un fichero **build.xml** en el proyecto de la aplicación **curso** y define las siguientes tareas de *Ant*.
  - Crear el fichero **curso.war**.
  - Listar las aplicaciones desplegadas en **ServidorLinuxXX**.
  - Desplegar la aplicación **curso** en **ServidorLinuxXX**.
  - Iniciar la aplicación **curso** en **ServidorLinuxXX**.
  - Parar la aplicación **curso** en **ServidorLinuxXX**.
  - Replegar la aplicación **curso** en **ServidorLinuxXX**.
- Prueba las tareas de *Ant*.

1. Configurar el acceso al modo *text* de *Tomcat Web Manager*.

   1.1. Inicia sesión en **ServidorLinuxXX** con un usuario con privilegios de administrador.

   1.2. En **ServidorLinuxXX** edita el fichero **/var/lib/tomcat7/conf/tomcat-users.xml** y realiza la configuración que se muestra en la Figura 8.60.

```
<role rolename="manager-gui"/>
<role rolename="manager-script"/>
<user username="tomcat" password="despliegue" roles="manager-gui"/>
<user username="tomcatscript" password="despliegue" roles="manager-script"/>
```

Figura 8.60: Fichero **/var/lib/tomcat7/conf/tomcat-users.xml/**

   1.3. Reinicia *Tomcat*.

   1.4. Desde el equipo **DesarrolloW7XX** accede `http://192.168.1.1X:8080/manager/text` e introduce el usuario creado, Figuras 8.61 y 8.62.

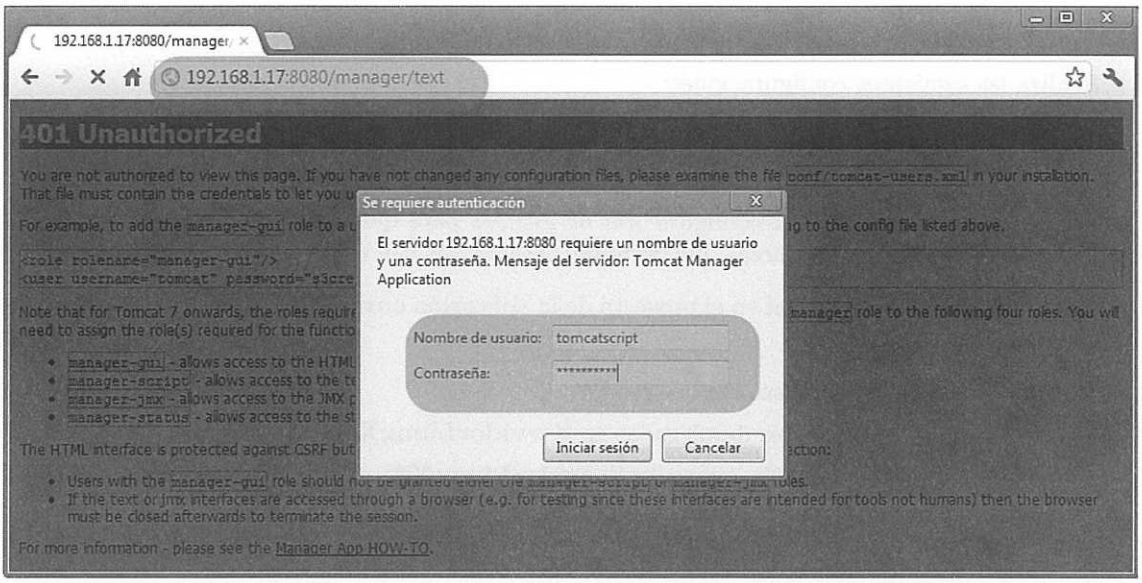


Figura 8.61: Acceso al modo *text Tomcat Web Manager*

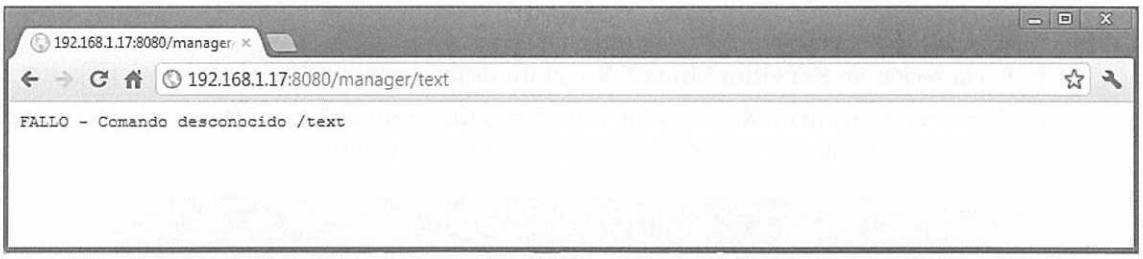


Figura 8.62: Acceso al modo *text Tomcat Web Manager*

1.5. Accede a `http://192.168.1.1X:8080/manager/text/list` para enviar el comando `list` que lista el contenido de las aplicaciones desplegadas en el servidor, Figura 8.63.

Figura 8.63: Comando list

2. **Configurar las librerías de *Tomcat* para *Ant* en *Eclipse*.**

   2.1. Inicia sesión en **DesarrolloW7XX**.

   2.2. Inicia *Eclipse* y abre el proyecto de la aplicación **curso**.

   2.3. Accede a ***Windows, Preferences, Ant y Runtime***, Figura 8.64.

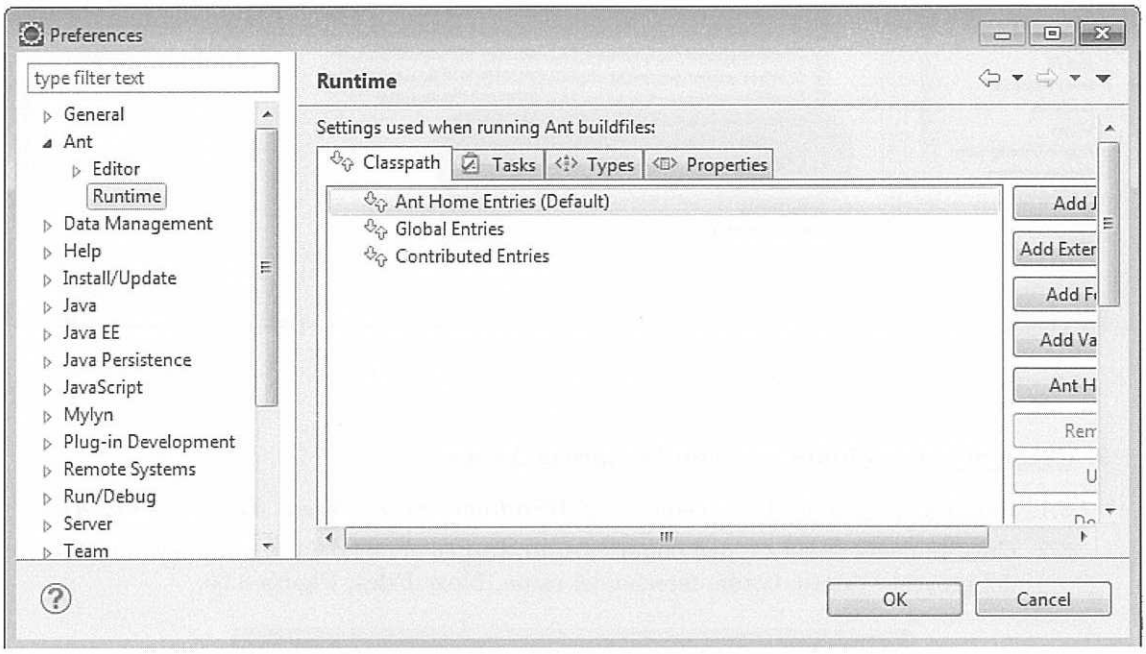


Figura 8.64: Librerías de *Ant* en *Eclipse*

   2.4. Selecciona ***Ant Home Entries*** y utiliza el botón ***Add External Jar*** para añadir las siguientes librerías, Figura 8.65 (están disponibles en la instalación local de *Tomcat*).

   - `CATALINA_HOME/lib/catalina-ant.jar`.
   - `CATALINA_HOME/lib/ tomcat-coyote.jar`.
   - `CATALINA_HOME/lib/tomcat-util.jar`.
   - `CATALINA_HOME/bin/ tomcat-juli.jar`.

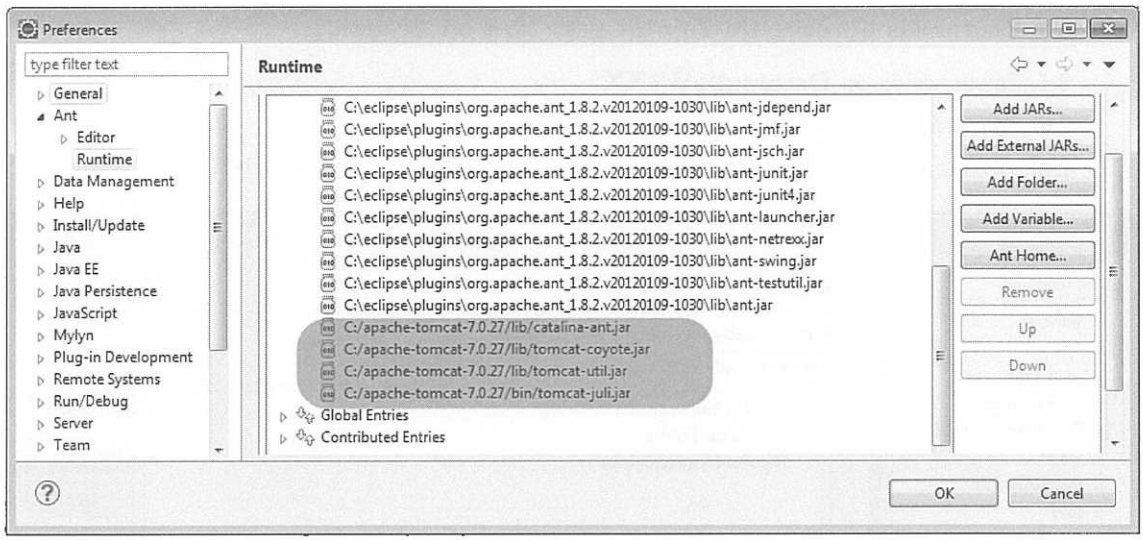


Figura 8.65: Librerías *Tomcat* para *Ant*

3. <u>**Crear el fichero build.xml con las tareas de *Ant*.**</u>

   3.1. Habilita la vista de *Ant* accediendo a ***Windows, Show View, Other, Ant, Ant.***

   3.2. Crea un nuevo fichero con el nombre **build.xml** en el raíz del proyecto (sobre el nombre del proyecto **curso**, botón derecho del ratón, **New File**), Figura 8.66.

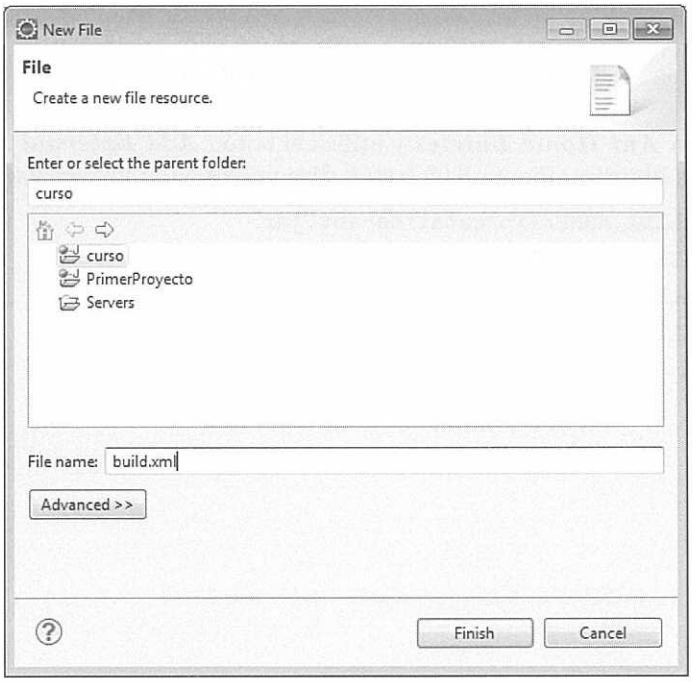


Figura 8.66: Crear el fichero **build.xml**

3.3. Configura las tarea de *Ant*, Figura 8.67.

```
<project name="Aplicación Curso" basedir=".">
    <property name="managerUrl" value="http://192.168.1.17:8080/manager/text" />
    <property name="username" value="tomcatscript" />
    <property name="password" value="despliegue" />
    <property name="contextPath" value="/curso" />
    <property name="warFileName" value="curso.war" />

    <taskdef name="list" classname="org.apache.catalina.ant.ListTask" />
    <taskdef name="deploy" classname="org.apache.catalina.ant.DeployTask" />
    <taskdef name="reload" classname="org.apache.catalina.ant.ReloadTask" />
    <taskdef name="start" classname="org.apache.catalina.ant.StartTask" />
    <taskdef name="stop" classname="org.apache.catalina.ant.StopTask" />
    <taskdef name="undeploy" classname="org.apache.catalina.ant.UndeployTask" />

    <target name="crearWAR">
        <war destfile="${warFileName}" webxml="WebContent/WEB-INF/web.xml" update="true">
            <classes dir="build\classes"/>
            <fileset dir="WebContent">
                <exclude name="WEB-INF/web.xml"/>
            </fileset>
        </war>
    </target>

    <target name="list">
        <list url="${managerUrl}" username="${username}" password="${password}" />
    </target>
    <target name="deploy" description="Despliega la aplicación">
        <deploy url="${managerUrl}" username="${username}" password="${password}" path="${contextPath}"
            war="file:${basedir}/${warFileName}" />
    </target>
    <target name="reload" description="Recarga la aplicación">
        <reload url="${managerUrl}" username="${username}" password="${password}" path="${contextPath}" />
    </target>
    <target name="undeploy" description="Borra la aplicación">
        <undeploy url="${managerUrl}" username="${username}" password="${password}" path="${contextPath}" />
    </target>
</project>
```

Figura 8.67: Fichero **build.xml**

3.4. Observa la vista ***Outline***, Figura 8.68.

Figura 8.68: Vista **Outline** con las tareas de *Ant*

3.5. En la vista inferior *Ant* añade el fichero **build.xml** pinchando sobre el botón ***Add Builds File***, Figuras 8.69 y 8.70.

Figura 8.69: Añadir fichero **build.xml** a la vista *Ant*

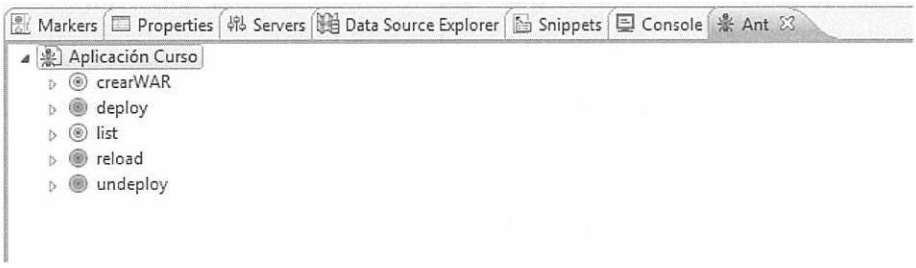


Figura 8.70: Vista **Ant**

4. <u>Probar las tareas de *Ant*</u>.

4.1. En la vista *Ant* selecciona la tarea **crearWAR**, haz clic con el botón derecho del ratón, selecciona ***Run As*** y ***Ant Build***, Figura 8.71.

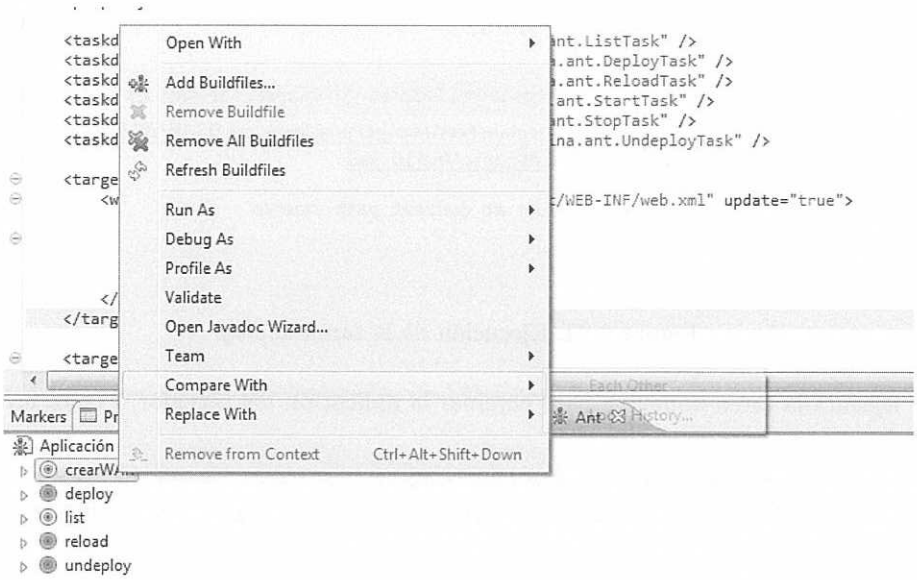


Figura 8.71: Ejecutar tarea en *Ant*

4.2. Observa el resultado de la tarea en la vista *Console*, Figura 8.72.

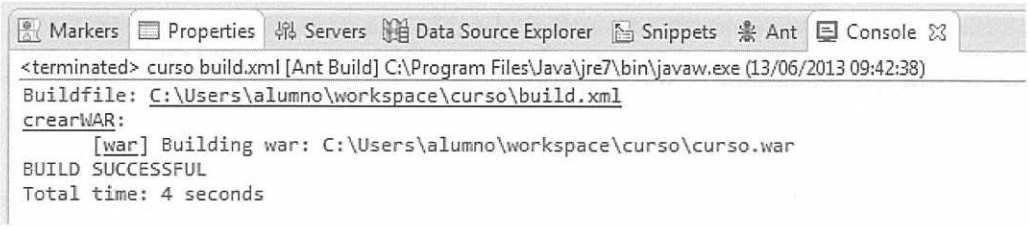


Figura 8.72: War creado

4.3. Ejecuta la tarea *list* para listar las aplicaciones desplegadas en el servidor, Figura 8.73.

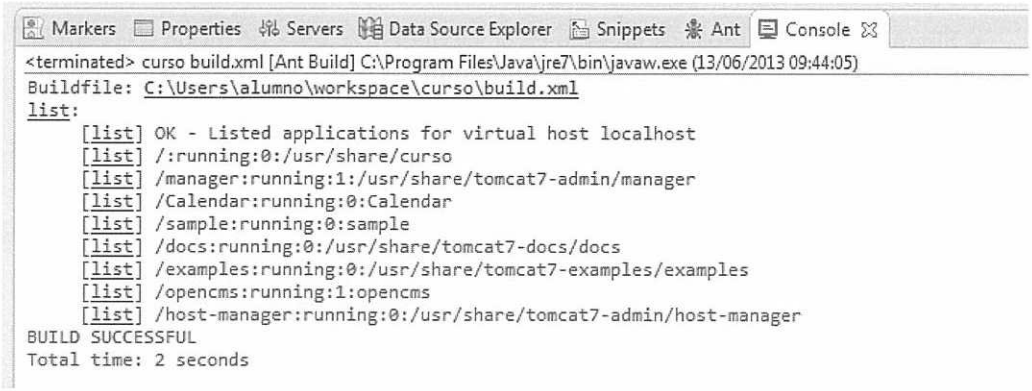


Figura 8.73: Ejecución de la tarea *list*

4.4. Ejecuta la tarea *deploy* para desplegar la aplicación en el servidor, Figura 8.74.

```
Markers  Properties  Servers  Data Source Explorer  Snippets  Ant  Console
<terminated> curso build.xml [Ant Build] C:\Program Files\Java\jre7\bin\javaw.exe (13/06/2013 09:45:08)
Buildfile: C:\Users\alumno\workspace\curso\build.xml
deploy:
   [deploy] OK - Deployed application at context path /curso
BUILD SUCCESSFUL
Total time: 2 seconds
```

Figura 8.74: Ejecución de la tarea *deploy*

4.5. Ejecuta la tarea *undeploy* para eliminar la aplicación del servidor, Figura 8.75.

```
Markers  Properties  Servers  Data Source Explorer  Snippets  Ant  Console
<terminated> curso build.xml [Ant Build] C:\Program Files\Java\jre7\bin\javaw.exe (13/06/2013 09:45:31)
Buildfile: C:\Users\alumno\workspace\curso\build.xml
undeploy:
 [undeploy] OK - Undeployed application at context path /curso
BUILD SUCCESSFUL
Total time: 1 second
```

Figura 8.75: Ejecución de la tarea *undeploy*

8.9. Sesiones

Realiza las siguientes configuraciones:

- Importa la aplicación **compras** en *Eclipse* (sobre **DesarrolloW7XX**), genera un archivo **.war** y despliégala en **ServidorLinuxXX**.
- Accede a la aplicación, añade varios articulos a la cesta y prueba cómo *Tomcat* gestiona las sesiones.
  - Parando el servidor "ordenadamente".
  - Simulando que se produce un fallo en el servidor y se para de forma inesperada.

1. Importar la aplicación compras en *Eclipse*.
   1.1. Inicia sesión en **DesarrolloW7XX**.
   1.2. Inicia *Eclipse*.
   1.3. Importa la aplicación compras (***File, Import, Existing Projects into Workspace*** y selecciona **compras.zip**, Figura 8.76.
2. Crear el fichero **compras.war** y desplegar la aplicación.
   2.1. Crea el fichero **compras.war** (puedes usar la tarea de *Ant* o los menús de *Eclipse*).
   2.2. Despliega la aplicación en **ServidorLinuxXX** usando **Tomcat Web Manager**, Figura 8.77. Observa que el tiempo para que expiren la sesiones es de 15 minutos.

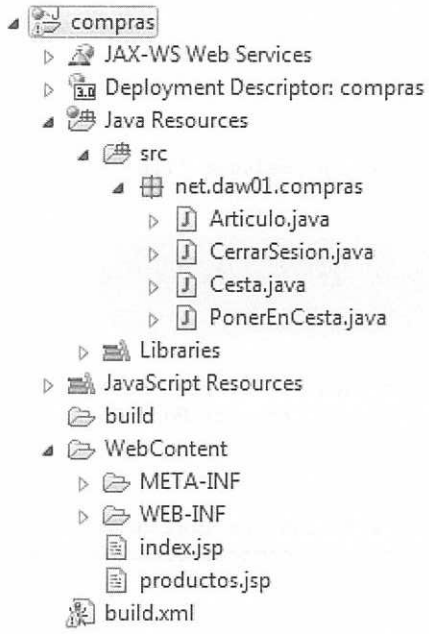


Figura 8.76: Aplicación compras

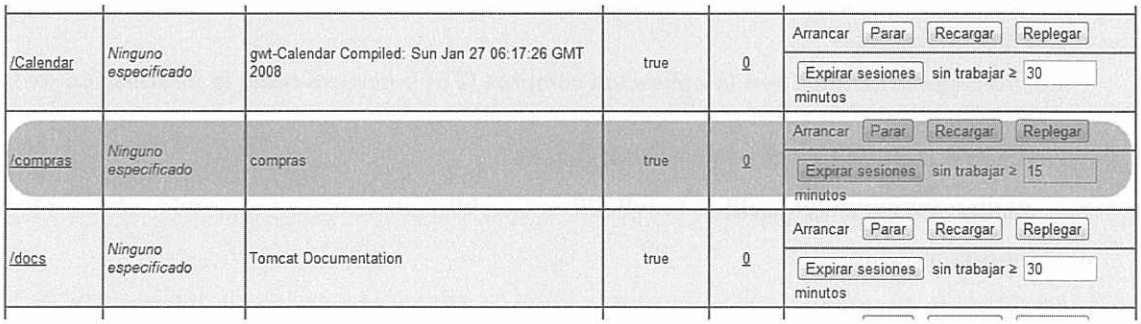


Figura 8.77: Aplicación compras desplegada

3. **Tiempo de vida de las sesiones.**

   3.1. Observa que el tiempo para que expiren la sesiones en la aplicación compras es de 15 minutos, Figura 8.77.

   3.2. Consulta el descriptor de despliegue (**web.xml**) de compras, Figura 8.78.

```
<display-name>compras</display-name>

<welcome-file-list>
    <welcome-file>index.jsp</welcome-file>
</welcome-file-list>

<session-config>
    <session-timeout>15</session-timeout>
</session-config>

<servlet>
    <servlet-name>PonerEnCesta</servlet-name>
    <servlet-class>net.daw01.compras.PonerEnCesta</servlet-class>
</servlet>

<servlet>
    <servlet-name>CerrarSesion</servlet-name>
    <servlet-class>net.daw01.compras.CerrarSesion</servlet-class>
</servlet>
```

Figura 8.78: Descriptor de despliegue de compras

4. **Sesiones (StandarManager).**

   4.1. Accede a la aplicación compras `http://192.168.1.X7:8080/compras` y compra tres artículos, Figura 8.79.

   4.2. Sin cerrar el navegador, para que la sesión siga activa, para el servidor *Tomcat* en **ServidorLinuxXX**.

   4.3. En **ServidorLinuxXX** consulta el directorio **/var/lib/tomcat7/work/CATALINA/localhost/compras** y observa que se ha creado el fichero **SESSIONS.ser**, Figura 8.80.

   4.4. Inicia Tomcat.

   4.5. Recarga la ventana con la aplicación compras (F5) y observa como la información de la sesión sigue activa (los productos siguen en la cesta).

   4.6. Ahora mata el proceso que ejecuta *Tomcat*.

```
sudo killall -9 java
```

   4.7. Inicia Tomcat.

   4.8. Recarga la ventana con la aplicación compras (F5) y observa que la información de la sesión se ha perdido (la cesta eatá vacía).

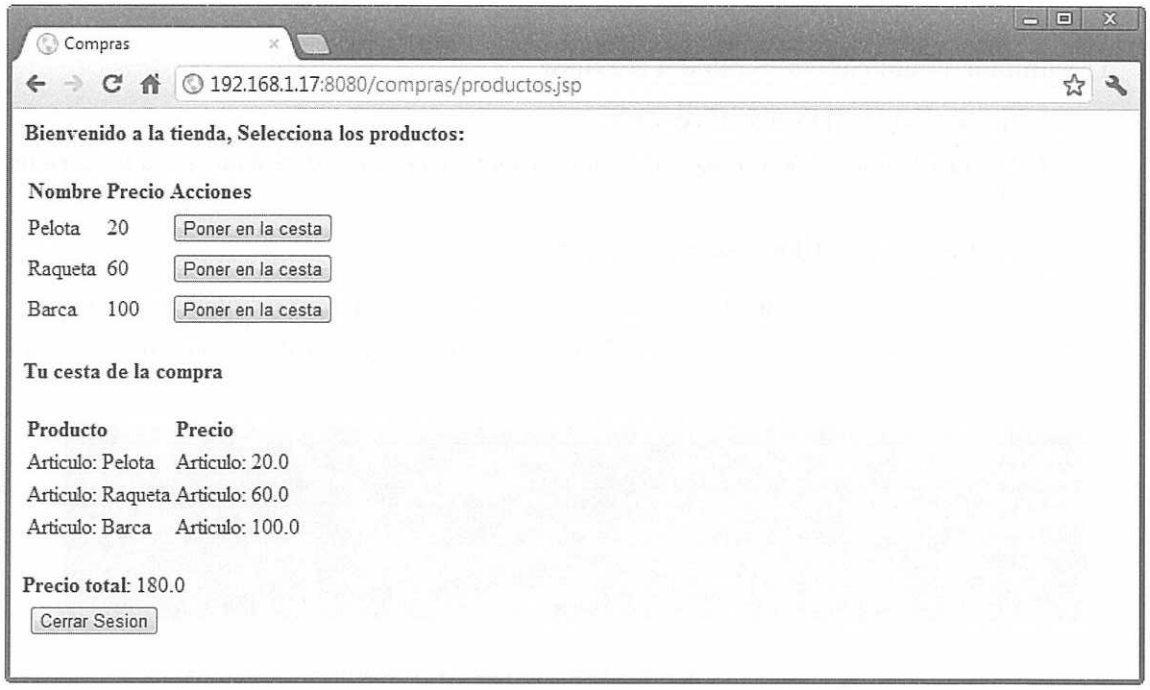


Figura 8.79: Compra de artículos

```
alumno@ServidorLinux01:/var/lib/tomcat7/work/Catalina/localhost/compras$ ls
org  SESSIONS.ser
alumno@ServidorLinux01:/var/lib/tomcat7/work/Catalina/localhost/compras$ _
```

Figura 8.80: Directorio **/var/lib/tomcat7/work/CATALINA/localhost/compras**

8.10. Autenticación y autorización (*MemoryRealm*)

Protege el acceso a la aplicación **curso** utilizando un *MemoryRealm* con autenticación BASIC para que solo puedan acceder los usuarios **profesor** y **alumno** (pertenecientes al **role curso**). La información sobre usuarios y roles se almacenará en el fichero **$CATALINA_BASE/conf/tomcat-users.xml**.

1. **Eliminar la aplicación curso del servidor.**

   1.1. Inicia sesión en **DesarrolloW7XX**.

   1.2. Elimina la aplicación **curso** del servidor *Tomcat* (puedes usar el *manager* o la tarea de *Ant*).

2. **Crear el fichero con los usuarios y roles.**

   2.1. Inicia sesión en **ServidorLinuxXX** con un usuario privilegios de administración.

   2.2. Edita el fichero **/var/lib/tomcat7/conf/tomcat-users.xml** y añade los siguientes elementos, Figura 8.81.

```
<role rolename="manager-gui"/>
<role rolename="manager-script"/>
<role rolename="curso"/>
<user username="tomcat" password="despliegue" roles="manager-gui"/>
<user username="tomcatscript" password="despliegue" roles="manager-script"/>
<user username="alumno" password="alumno" roles="curso"/>
<user username="profesor" password="profesor" roles="curso"/>
```

Figura 8.81: Fichero **/var/lib/tomcat7/conf/tomcat-users.xml**

   2.3. Reinicia *Tomcat* para que se lea de nuevo el fichero.

3. **Configurar el *Realm*.**

   3.1. En **DesarrolloW7XX** crea un fichero **context.xml** para la aplicación **curso** (**META-INF/context.xml**) y añade los siguientes elementos, Figura 8.82.

```
<Context>
    <Realm className="org.apache.catalina.realm.MemoryRealm" />
</Context>
```

Figura 8.82: Fichero **META-INF/context.xml**

4. **Proteger la aplicación con el *MemoryRealm* (Autenticación BASIC).**

   4.1. En **DesarrolloW7XX** edita el descriptor de despliegue (**web.xml**) de la aplicación **curso** y añade los siguientes elementos, Figura 8.83.

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="http://java.sun.com/xml/ns/javaee"
    xmlns:web="http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
        http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
    id="WebApp_ID" version="3.0">

    <display-name>curso</display-name>
    <welcome-file-list>
        <welcome-file>index.html</welcome-file>
    </welcome-file-list>

    <security-constraint>
        <web-resource-collection>
            <web-resource-name>MemoryRealm</web-resource-name>
            <url-pattern>/*</url-pattern>
        </web-resource-collection>
        <auth-constraint>
            <role-name>curso</role-name>
        </auth-constraint>
    </security-constraint>

    <login-config>
        <auth-method>BASIC</auth-method>
        <realm-name>Acceso al curso</realm-name>
    </login-config>

    <servlet>
        <servlet-name>HolaServlet</servlet-name>
        <servlet-class>net.daw01.curso.HolaServlet</servlet-class>
    </servlet>
```

Figura 8.83: Fichero **web.xml** de la aplicación **curso**

4.2. Elimina la aplicación **curso** del servidor *Tomcat*.

4.3. Genera el fichero **curso.war** con los cambios.

4.4. Despliega el nuevo fichero **curso.war**.

5. **Proteger la aplicación con el *MemoryRealm* y configurar la autenticación (BASIC).**

5.1. Accede a la aplicación y observa que se pide usuario y contraseña. Solo puedes acceder como alumno o profesor, Figuras 8.84 y 8.85.

Figura 8.84: Acceso a la aplicación curso

Figura 8.85: Acceso a la aplicación curso

8.11. Autenticación y autorización (*UserDatabaseRealm*)

Analiza la configuración del *UserDatabaseRealm* definido por defecto en *Tomcat* y cómo se utiliza en la aplicación ***manager***.

1. Inicia sesión en **ServidorLinuxXX** con un usuario privilegios de administración.
2. Consulta el fichero **/var/lib/tomcat7/server.xml** y observa el recurso JNDI que define un *UserDatabase*, Figura 8.86.

```
<!-- Global JNDI resources
     Documentation at /docs/jndi-resources-howto.html
-->
<GlobalNamingResources>
  <!-- Editable user database that can also be used by
       UserDatabaseRealm to authenticate users
  -->
  <Resource name="UserDatabase" auth="Container"
            type="org.apache.catalina.UserDatabase"

            description="User database that can be updated and saved"
            factory="org.apache.catalina.users.MemoryUserDatabaseFactory"
            pathname="conf/tomcat-users.xml" />
</GlobalNamingResources>
```

Figura 8.86: Fichero **/var/lib/tomcat7/server.xml**

3. Consulta el fichero **/var/lib/tomcat7/server.xml** y observa *UserDatabaseRealm* definido a nivel de `<Engine>`, Figura 8.87.

```
<!-- This Realm uses the UserDatabase configured in the global JNDI
     resources under the key "UserDatabase".  Any edits
     that are performed against this UserDatabase are immediately
     available for use by the Realm.  -->
      <Realm className="org.apache.catalina.realm.UserDatabaseRealm"
       resourceName="UserDatabase"/>
```

Figura 8.87: Fichero **/var/lib/tomcat7/server.xml**

4. Consulta del descriptor de despliegue de la aplicación manager en **/usr/share/tomcat7-admin /manager/WEB-INF/web.xml** y observa cómo se protege el acceso a la misma utilizando el *UserDatabaseRealm*, Figuras 8.88 y 8.89.

```
<security-constraint>
  <web-resource-collection>
    <web-resource-name>HTML Manager interface (for humans)</web-resource-name>
    <url-pattern>/html/*</url-pattern>
  </web-resource-collection>
  <auth-constraint>
     <role-name>manager-gui</role-name>
  </auth-constraint>
</security-constraint>
<security-constraint>
  <web-resource-collection>
    <web-resource-name>Text Manager interface (for scripts)</web-resource-nam$
    <url-pattern>/text/*</url-pattern>
  </web-resource-collection>
  <auth-constraint>
     <role-name>manager-script</role-name>
  </auth-constraint>
</security-constraint>
<security-constraint>
```

Figura 8.88: Fichero **/usr/share/tomcat7-admin/manager/WEB-INF/web.xml**

```
<!-- Define the Login Configuration for this Application -->
<login-config>
  <auth-method>BASIC</auth-method>
  <realm-name>Tomcat Manager Application</realm-name>
</login-config>
```

Figura 8.89: Fichero **/usr/share/tomcat7-admin/manager/WEB-INF/web.xml**

8.12. Autenticación y autorización (*JDBCRealm*)

Protege el acceso a la aplicación **compras** utilizando un *JDBCRealm* con autenticación FORM para que solo puedan acceder los usuarios **mortadelo** y **filemon** (pertenecientes al *role* **compras**).

1. **Eliminar la aplicación curso del servidor**.
   1.1. Inicia sesión en **DeasarrolloWindowsXX**.
   1.2. Elimina la aplicación **compras** del servidor *Tomcat* (puedes usar el *manager* o la tarea de *Ant*).
2. **Configuración de la base de datos**.
   2.1. Inicia sesión en **DeasarrolloWindowsXX**.
   2.2. Inicia una navegador y accede a `http://192.168.1.X7/phpmyadmin`.
   2.3. Inicia sesión con el usuario **root**.
   2.4. Accede a **Privilegios**, pincha en **Agregar un nuevo usuario**. Introduce **tomcatusers** como nombre de usuario, **localhost** como servidor y **tomcatusers** como clave del usuario. Marca la opción **Crear base de datos con el mismo nombre y otorgar todos los privilegios** y pincha en **Crear Usuario**, Figura 8.90.

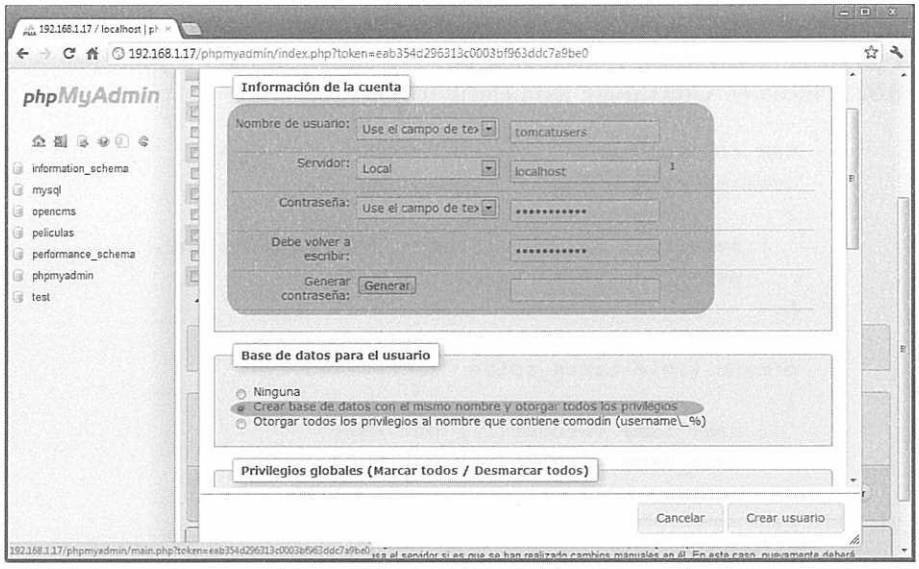


Figura 8.90: Crear el usuario y la base de datos **tomcatusers**

2.5. Se han creado el usuario **tomcatusers**, la base de datos **tomcatusers** y se han otorgado todos los privilegios al usuario sobre la base de datos, Figura 8.91.

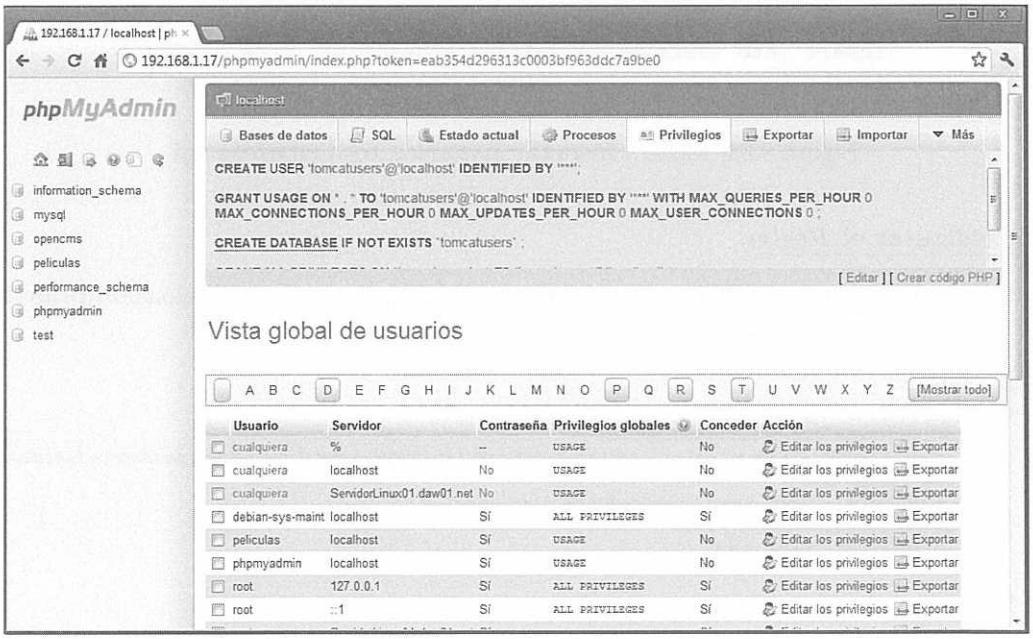


Figura 8.91: Usuario y la base de datos creados

2.6. Cierra la sesión de *phpmyadmin* del usuario **root.**

2.7. Inicia sesión en *phpmyadmin* con el usuario **tomcatusers.**

2.8. En la parte izquierda selecciona la base de datos **tomcatusers.**

2.9. Pincha en **Importar**.

2.10. Pincha en **Examinar** y selecciona el script **crear_tablas_tomcatusers.sql**, Figura 8.92. Pincha en **Continuar** para ejecutar el *script*.

```
use tomcatusers;
create table users
(
    user_name varchar(12) not null primary key,
    user_pass varchar(12) not null
);

create table users_roles
(
    user_name varchar(12) not null,
    role_name varchar(12) not null,
    primary key(user_name, role_name)
);
```

Figura 8.92: Fichero **crear_tablas_tomcatusers.sql**

2.11. Pincha en **Examinar** y selecciona el script **insertar_usuarios_tomcatusers.sql**, Figura 8.93. Pincha en **Continuar** para ejecutar el *script*.

```
insert into users values("mortadelo", "mortadelo");
insert into users values("filemon", "filemon");
insert into users_roles values("mortadelo", "compras");
insert into users_roles values("filemon", "compras");
```

Figura 8.93: Fichero **insertar_usuarios_tomcatusers.sql**

3. Configurar el *Realm*.

3.1. En **DesarrolloW7XX** crea un fichero **context.xml** para la aplicación **compras** (**META-INF/context.xml**) y añade los siguientes elementos, Figura 8.94.

```
<Context>
    <Realm  className="org.apache.catalina.realm.JDBCRealm"
            driverName="com.mysql.jdbc.Driver"
            connectionURL="jdbc:mysql://localhost/tomcatusers?user=tomcatusers&amp;password=tomcatusers"
            userTable="users"
            userNameCol="user_name" userCredCol="user_pass"
            userRoleTable="users_roles"
            roleNameCol="role_name"
    />
</Context>
```

Figura 8.94: Fichero **META-INF/context.xml**

3.2. Para que la aplicación se pueda conectar a *MySQL* usando JDBC es necesario que el driver JDBC esté disponible. Podemos añadirlo en el directorio **WEB-INF/lib** de la aplicación o en **$CATALINA_HOME/lib**. Vamos a optar por la segunda opción y así estará disponible para todas las aplicaciones del servidor.

a. Accede a `http://dev.mysql.com/downloads/connector/j/` y descarga el driver JDBDC para *MySQL*.

b. Descomprime el fichero descargado para obtener el fichero (en el momento de realizar la práctica **mysqlconnectorjava5.1.25bin.jar**).

c. Usa *Filezilla* para subir el fichero **mysqlconnectorjava5.1.25bin.jar** a **/home/alumno**.

d. En **ServidorLinuxXX** cambia el propietario y el grupo del fichero al usuario **root** y grupo **root** y concédele permisos de lectura y escritura para todos los usuarios.

```
sudo chown root:root /home/alumno/mysql-connector-java-5.1.25-bin.jar
sudo chmod 777 /home/alumno/mysql-connector-java-5.1.25-bin.jar
```

e. Mueve el fichero desde del directorio **home** del usuario **alumno** al directorio **/usr/share/tomcat7/lib**.

```
sudo mv /home/alumno/mysql-connector-java-5.1.20-bin.jar /usr/share/tomcat7/lib
```

3.3. Reinicia *Tomcat*.

4. **Proteger la aplicación con el *JDBCRealm* (Autenticación FORM).**

4.1. En **DesarrolloW7XX** edita el descriptor de despliegue (**web.xml**) de la aplicación **compras** y añade los siguientes elementos, Figura 8.95.

4.2. Crea los ficheros **WEB-INF/seguro/login.jsp** y **WEB-INF/seguro/login-error.jsp** con el contenido que se muestra en las Figuras 8.96 y 8.97.

4.3. Elimina la aplicación **compras** del servidor *Tomcat*.

4.4. Genera el fichero **compras.war** con los cambios.

4.5. Despliega el nuevo fichero **compras.war**.

5. **Acceso a al aplicación.**

5.1. Accede a la aplicación y observa que se pide usuario y contraseña. Solo puedes acceder como **mortadelo** o como **filemon**, Figuras 8.98 y 8.99.

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="http://java.sun.com/xml/ns/javaee" xmlns:web="http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
    id="WebApp_ID" version="3.0">

    <display-name>compras</display-name>

    <welcome-file-list>
        <welcome-file>index.jsp</welcome-file>
    </welcome-file-list>

    <security-constraint>
        <web-resource-collection>
            <web-resource-name>JDBCRealm</web-resource-name>
            <url-pattern>/*</url-pattern>
        </web-resource-collection>
        <auth-constraint>
            <role-name>compras</role-name>
        </auth-constraint>
    </security-constraint>

    <login-config>
        <auth-method>FORM</auth-method>
        <realm-name>Compras</realm-name>
        <form-login-config>
            <form-login-page>/WEB-INF/seguro/login.jsp</form-login-page>
            <form-error-page>/WEB-INF/seguro/login-error.jsp</form-error-page>
        </form-login-config>
    </login-config>

    <session-config>
        <session-timeout>15</session-timeout>
    </session-config>
```

Figura 8.95: Fichero **web.xml** de la aplicación compras

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Insert title here</title>
</head>
<body>

    <form method="POST"
        action='<%=response.encodeURL("j_security_check")%>'>
        Usuario:<input type="text" name="j_username">
        Password:<input type="password" name="j_password">
        <input type="submit" value="Login">
    </form>

</body>
</html>
```

Figura 8.96: Fichero **WEB-INF/seguro/login.jsp**

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Página de error de login</title>
</head>
<body>

Usuario o password icorrectos, prueba <a href="<%= response.encodeURL("index.jsp") %>"> de nuevo</a>.

</body>
</html>
```

Figura 8.97: Fichero **WEB-INF/seguro/login.jsp**

Figura 8.98: Acceso a la aplicación compras(1)

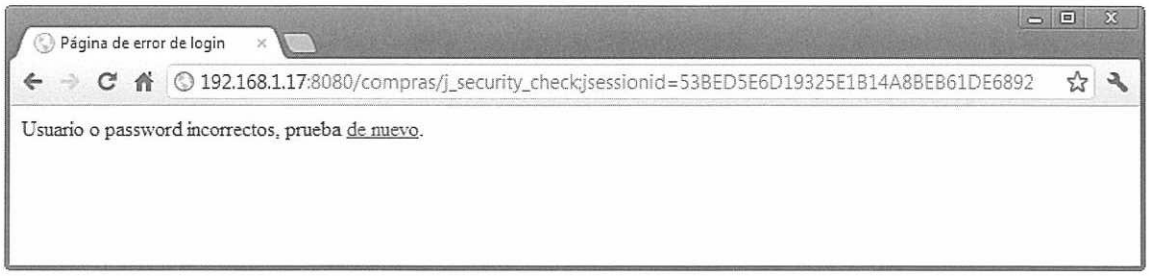


Figura 8.99: Acceso a la aplicación compras(2)

8.13. *Valves*

- Implementa y prueba el *Valve* que se propone en la práctica.
- Observa el funcionamiento del **AccessLogValve** definido por defecto en *Tomcat*.
- Configura el *Valve* predefinido **RemoteAddrValve** para que solo se pueda acceder a las aplicaciones desde **DesarrolloW7XX** (`192.168.1.X6`).

1. ***LogginValve***.

   1.1. Inicia sesión en **DesarrolloW7XX**.

   1.2. Crea un proyecto Java (***File, New, Project, Java, Java Project***).

   1.3. Llámalo *valve* y pincha en ***Next***.

   1.4. Accede a la pestaña ***Library*** y añade (***Add External Jar***) las librerías **catalina.jar** y **servlet-api.jar** (la puedes encontrar en la instalación local de *Tomcat*), Figura 8.100. Pincha en ***Finish*** para crear el proyecto.

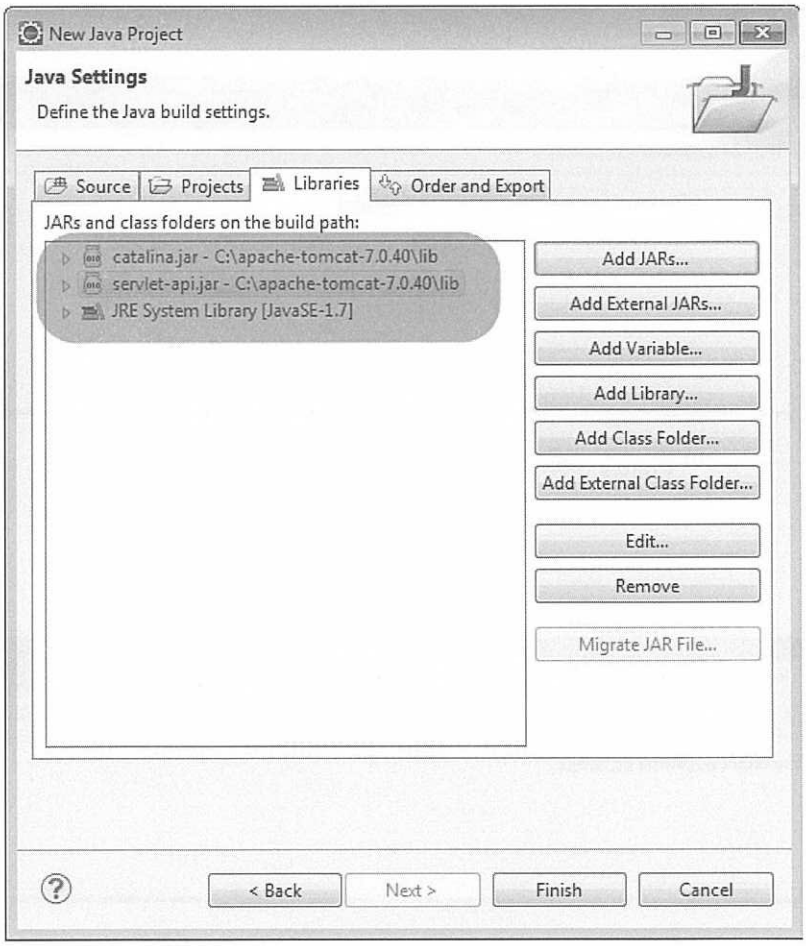


Figura 8.100: Librerías en el proyecto **valve**

1.5. Crea la clase **LogginValve** en el paquete **net.daw01.valves** con el siguiente contenido, Figura 8.101.

```
package net.daw01.valves;

import java.io.IOException;

import javax.servlet.*;

import org.apache.catalina.Valve;
import org.apache.catalina.valves.*;
import org.apache.catalina.connector.Request;
import org.apache.catalina.connector.Response;


public class LogginValve extends ValveBase {

    @Override
    public void invoke(Request request, Response response)
            throws IOException, ServletException {

        String IPRemota = request.getRemoteAddr();
        String URI = request.getRequestURI();
        System.out.println("URI " + URI + " accedida desde: "+IPRemota);
        Valve nextValve = getNext();
        if(nextValve!=null){
            nextValve.invoke(request, response);
        }
    }
}
```

Figura 8.101: Clase **LogginValve**

1.6. Exporta la clase **LogginValve.class** al fichero **despliegue-valves.jar** (sobre el proyecto **valve**, **botón derecho de ratón**, ***Export, Java, JAR File***).

1.7. Sube el fichero **despliegue-valves.jar** a **$CATALINA_HOME/lib** en **ServidorLinuxXX**.

a. Usa *Filezilla* para subir el fichero **despliegue-valves.jar** a **/home/alumno**.

b. En **ServidorLinuxXX** cambia el propietario y el grupo del fichero al usuario **root** y grupo **root** y concédele permisos de lectura y escritura para todos los usuarios.

```
sudo chown root:root /home/alumno/despliegue-valves.jar
sudo chmod 777 //home/alumno/despliegue-valves.jar
```

c. Mueve el fichero desde del directorio **home** del usuario **alumno** al directorio **/usr/share/tomcat7/lib**.

```
sudo mv /home/alumno/despliegue-valves.jar /usr/share/tomcat7/lib
```

1.8. Edita el fichero **/var/lib/tomcat7/conf/server.xml** y añade los siguientes elementos para configurar el *valve* en todas las aplicaciones de Host **localhost**, Figura 8.102.

```
<Host name="localhost"  appBase="webapps"
      unpackWARs="true" autoDeploy="true">

  <!-- SingleSignOn valve, share authentication between web applications
       Documentation at: /docs/config/valve.html -->
  <!--
  <Valve className="org.apache.catalina.authenticator.SingleSignOn" />
  -->

  <!-- Access log processes all example.
       Documentation at: /docs/config/valve.html
       Note: The pattern used is equivalent to using pattern="common" -->
  <Valve className="org.apache.catalina.valves.AccessLogValve" directory=$
         prefix="localhost_access_log." suffix=".txt"
         pattern="%h %l %u %t &quot;%r&quot; %s %b" />

  <Valve className="net.daw01.valves.LogginValve" />
```

Figura 8.102: Configuración de **LogginValve** en el fichero **server.xml**

1.9. Reinicia *Tomcat*.

1.10. Desde **DesarrolloW7XX** accede a varias aplicaciones del servidor.

1.11. En **ServidorLinuxXX** consulta el fichero **/var/log/tomcat7/catalina.out** y observa lo que ha escrito el *valve* que hemos configurado, Figura 8.103.

```
alumno@ServidorLinux01:/var/log/tomcat7$ tail -f catalina.out
URI /compras/ accedida desde: 192.168.1.16
URI /favicon.ico accedida desde: 192.168.1.16
URI /compras/productos.jsp accedida desde: 192.168.1.16
URI /favicon.ico accedida desde: 192.168.1.16
URI /curso/ accedida desde: 192.168.1.16
URI /curso/ accedida desde: 192.168.1.16
URI /curso/ accedida desde: 192.168.1.16
URI /favicon.ico accedida desde: 192.168.1.16
URI /curso/buenas.jsp accedida desde: 192.168.1.16
URI /favicon.ico accedida desde: 192.168.1.16
```

Figura 8.103: Fichero **/var/log/tomcat7/catalina.out**

.

2. ***LogginValve***.

2.1. Edita el fichero **var/lib/tomcat7/conf/server.xml** y observa el *valve* creado por defecto. Analiza su configuración, Figura 8.104.

2.2. Consulta el directorio **/var/log/tomcat7** y observa los ficheros **localhost_access_log**

.

3. ***RemoteAddrValve***.

3.1. Consulta la documentación del valve `http://tomcat.apache.org/tomcat-7.0-doc/config/valve.html\#Remote_Address_Filter`

```
<!-- Access log processes all example.
     Documentation at: /docs/config/valve.html
     Note: The pattern used is equivalent to using pattern="common" -->
<Valve className="org.apache.catalina.valves.AccessLogValve" directory=
       prefix="localhost_access_log." suffix=".txt"
       pattern="%h %l %u %t &quot;%r&quot; %s %b" />
```

Figura 8.104: Configuración de **AccessLogValveValve** en el fichero **server.xml**

3.2. Edita el fichero **/var/lib/tomcat7/conf/server.xml** y añade los siguientes elementos para configurar un *RemoteAddrValve* que solo permita las conexiones desde la ip `192.168.1.X6`, Figura 8.105.

```
<Host name="localhost"  appBase="webapps"
      unpackWARs="true" autoDeploy="true">

  <!-- SingleSignOn valve, share authentication between web applications
       Documentation at: /docs/config/valve.html -->
  <!--
  <Valve className="org.apache.catalina.authenticator.SingleSignOn" />
  -->

  <!-- Access log processes all example.
       Documentation at: /docs/config/valve.html
       Note: The pattern used is equivalent to using pattern="common" -->
  <Valve className="org.apache.catalina.valves.AccessLogValve" directory=
         prefix="localhost_access_log." suffix=".txt"
         pattern="%h %l %u %t &quot;%r&quot; %s %b" />

  <Valve className="net.daw01.valves.LogginValve" />

  <Valve className="org.apache.catalina.valves.RemoteAddrValve"
         allow="192.168.1.16" />

</Host>
```

Figura 8.105: Configuración de **RemoteAddrValve** en el fichero **server.xml**

3.3. Reinicia *Tomcat*.

3.4. Prueba a accede a alguna aplicación desde **DesarrolloW7XX**.

3.5. Prueba a accede a alguna aplicación desde **ServidorWindowXX**, Figura 8.106.

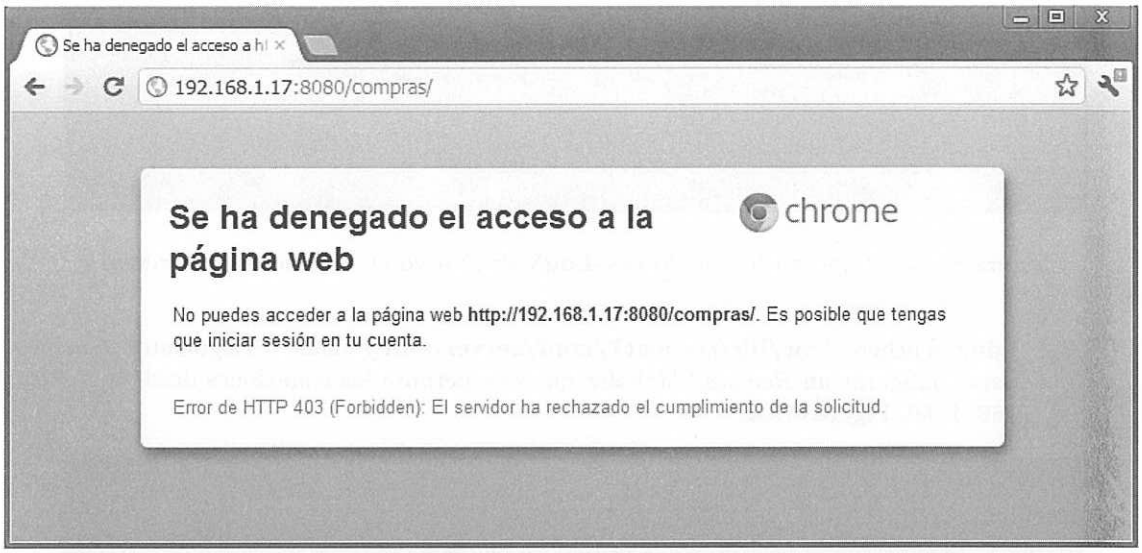


Figura 8.106: Denegación de acceso

8.14. Filtros

Implementa un filtro que cuente el número de accesos a la aplicación **curso** y deje constancia en fichero **/var/log/catalina.out**.

1. Inicia sesión en **DesarrolloW7XX**.
2. Crea un nuevo filtro (FiltroContador) en el proyecto **curso** con el siguiente contenido, Figura 8.107.
3. Edita el descriptor de despliegue de la aplicación **curso** y configura el filtro, Figura 8.108.
4. Elimina la aplicación **curso** del servidor *Tomcat*.
5. Genera el fichero **curso.war** con los cambios.
6. Despliega el nuevo fichero **curso.war**.
7. Desde **DesarrolloW7XX** accede a varias páginas de la aplicacíón curso.
8. En **ServidorLinuxXX** consulta el fichero **/var/log/tomcat7/catalina.out** y observa lo que ha escrito el filtro que hemos configurado, Figura 8.109.

```
package net.daw01.curso;

import java.io.IOException;
import javax.servlet.Filter;
import javax.servlet.FilterChain;
import javax.servlet.FilterConfig;
import javax.servlet.ServletContext;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;

public class FiltroContador implements Filter {

    private FilterConfig config;

    public void init(FilterConfig config) {
        this.config = config;
    }

    public void doFilter(ServletRequest request, ServletResponse response,
            FilterChain chain) throws IOException, ServletException {
        // Contexto de sesión de almacenará el contador
        ServletContext contexto = config.getServletContext();
        // Se comprueba si el atributo contador existe dentro del contexto
        Integer contador = (Integer) contexto.getAttribute("contador");
        if (contador == null) {
            contador = new Integer(0);
        }
        contador = new Integer(contador.intValue() + 1);
        contexto.setAttribute("contador", contador);
        System.out.println();
        System.out.println("Número de accesos a curso : " + contador.intValue());
        // Para que se invoque al siguiente filtro
        chain.doFilter(request, response);
    }

    public void destroy() {
        this.config = null;
    }
}
```

Figura 8.107: Clase **FiltroContador.java**

```
<filter>
    <filter-name>FiltroContador</filter-name>
    <filter-class>net.daw01.curso.FiltroContador</filter-class>
</filter>
<filter-mapping>
    <filter-name>FiltroContador</filter-name>
    <url-pattern>/*</url-pattern>
</filter-mapping>
```

Figura 8.108: Fichero **web.xml**

```
alumno@ServidorLinux01:/var/log/tomcat7$ tail -f catalina.out
Número de accesos a curso : 1
URI /favicon.ico accedida desde: 192.168.1.16
URI /curso/hola accedida desde: 192.168.1.16

Número de accesos a curso : 2
URI /favicon.ico accedida desde: 192.168.1.16
URI /curso/buenas.jsp accedida desde: 192.168.1.16

Número de accesos a curso : 3
URI /favicon.ico accedida desde: 192.168.1.16
```

Figura 8.109: Fichero **/var/log/tomcat7/catalina.out**

8.15. Ficheros de registros (*logs*)

Consulta los ficheros de configuración de *Tomcat* y observa las directivas utilizadas para definir la configuración de los ficheros de registros (*logs*). Posteriormente consulta estos ficheros.

1. Inicia sesión en **DesarrolloW7XX**.
2. Consulta el fichero **/var/lib/tomcat7/conf/logging.properties** y busca las directivas que definen la localización de los ficheros de registros (*logs*) de errores, Figura 8.110.

```
# limitations under the License.

handlers = 1catalina.org.apache.juli.FileHandler, 2localhost.org.apache.juli.Fi$

.handlers = 1catalina.org.apache.juli.FileHandler, java.util.logging.ConsoleHan$

############################################################
# Handler specific properties.
# Describes specific configuration info for Handlers.
############################################################

1catalina.org.apache.juli.FileHandler.level = FINE
1catalina.org.apache.juli.FileHandler.directory = ${catalina.base}/logs
1catalina.org.apache.juli.FileHandler.prefix = catalina.

2localhost.org.apache.juli.FileHandler.level = FINE
2localhost.org.apache.juli.FileHandler.directory = ${catalina.base}/logs
2localhost.org.apache.juli.FileHandler.prefix = localhost.

java.util.logging.ConsoleHandler.level = FINE
java.util.logging.ConsoleHandler.formatter = java.util.logging.SimpleFormatter
```

Figura 8.110: Fichero **/var/lib/tomcat7/conf/logging.properties**

3. Consulta el fichero **/var/lib/tomcat7/conf/server.xml** y busca el Valve que definen la localización y el formato de los ficheros de registros (logs) de accesos, Figura 8.111.

```
<!-- Access log processes all example.
     Documentation at: /docs/config/valve.html
     Note: The pattern used is equivalent to using pattern="common" -->
<Valve class Name="org.apache.catalina.valves.AccessLogValve"
       directory="logs"
       prefix="localhost_access_log." suffix=".txt"
       pattern="%h %l %u %t &quot;%r&quot; %s %b" />
```

Figura 8.111: Fichero **/var/lib/tomcat7/conf/server.xml**

4. Consulta el directorio **/var/lib/tomcat7/logs**, Figura 8.112.

```
alumno@ServidorLinux01:/var/log/tomcat7$ ls
catalina.2015-06-16.log    localhost.2015-06-17.log
catalina.2015-06-17.log    localhost_access_log.2015-06-16.txt
catalina.out               localhost_access_log.2015-06-17.txt
localhost.2015-06-16.log
```

Figura 8.112: Directorio **/var/lib/tomcat7/logs**

5. Consulta el contenido de los ficheros de *logs*.

8.16. Alojamiento virtual

Realiza la siguiente configuración en el servidor *Tomcat* instalado en **ServidorLinuxXX**.

- Configura el servidor DNS de **ServidorW2008XX** para que resuelva los nombres `europa.dawXX.net` y `africa.dawXX.net`.
- Crea y habilita un *host* virtual para el dominio `europa.dawXX.net`.
  - Directorio de aplicaciones **/var/lib/tomcat7/webapps-europa**.
  - El fichero de *logs* de acceso será **/var/log/tomcat7/europa_access_log.*fecha*.txt**.
  - Habilita *Tomcat Web Manager*.
  - Despliega la aplicación **curso**.
- Crea y habilita un *host* virtual para el dominio `africa.dawXX.net`.
  - Directorio de aplicaciones **/var/lib/tomcat7/webapps-africa**.
  - El fichero de *logs* de acceso será **/var/log/tomcat7/africa_access_log.*fecha*.txt**.
  - Habilita *Tomcat Web Manager*.
  - Despliega la aplicación **compras**.

1. **Configuración del servidor DNS**

   1.1. Configura el servidor DNS de **ServidorW2008XX** para que resuelva los nombres `europa.dawXX.net` y `africa.dawXX.net`. La dirección IP asociada a los nombres será la IP de **ServidoLinuxXX**, es decir, `192.168.1.X7`, Figura 8.113.

   1.2. Asegúrate que **DesarrolloW7XX** utiliza el servidor DNS que has configurado, Figura 8.114.

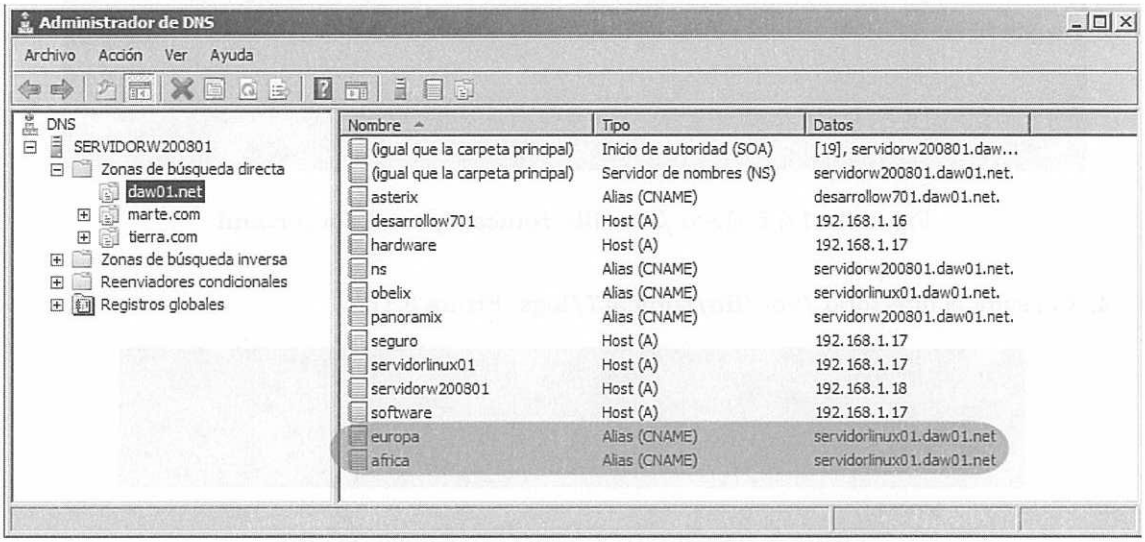


Figura 8.113: Configuración del servidor DNS en **ServidorW2008XX**

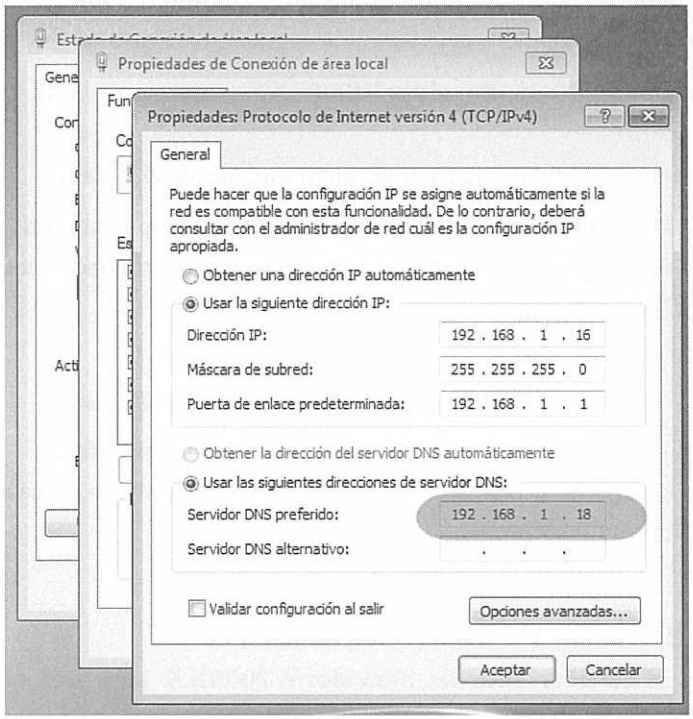


Figura 8.114: Servidor DNS en **DesarrolloW7XX**

2. ***Host* virtual para el dominio** `europa.dawXX.net`

   2.1. Inicia sesión en **ServidorLinuxXX** con un usuario con privilegios de administración.

   2.2. Crea el directorio **/var/lib/tomcat7/webapps-europa** y asígnale los permisos adecuados, Figura 8.115.

```
sudo mkdir  /var/lib/tomcat7/webapps-europa
sudo chown tomcat7:tomcat7  /var/lib/tomcat7/webapps-europa
sudo chmod 775 /var/lib/tomcat7/webapps-europa
```

```
drwxrwxr-x 7 tomcat7 tomcat7 4096 jun  9 13:46 webapps
drwxrwxr-x 2 tomcat7 tomcat7 4096 jun  9 14:17 webapps-europa
```

Figura 8.115: Directorio **/var/lib/tomcat7/webapps-europa**

   2.3. Edita el fichero **/var/lib/tomcat7/conf/server.xml** y habilita el *Host* virtual, Figura 8.116.

```
<Host name="europa.daw01.net"  appBase="webapps-europa"
      unpackWARs="true" autoDeploy="true">

  <Valve className="org.apache.catalina.valves.AccessLogValve" directory="logs"
         prefix="europa_access_log." suffix=".txt"
         pattern="%h %l %u %t &quot;%r&quot; %s %b" />

</Host>
```

Figura 8.116: Fichero **/var/lib/tomcat7/conf/server.xml**

   2.4. Reinicia *Tomcat*.

   2.5. Observa que se ha creado el fichero **/var/lib/tomcat7/conf/Catalina/europa.daw01.net/**, Figura 8.117.

```
alumno@ServidorLinux01:/var/lib/tomcat7/conf/Catalina$ ls -l
total 8
drwxr-xr-x 2 tomcat7 tomcat7 4096 jun  9 14:13 europa.daw01.net
drwxrwxr-x 2 root    tomcat7 4096 jun  4 16:45 localhost
alumno@ServidorLinux01:/var/lib/tomcat7/conf/Catalina$
```

Figura 8.117: Directorio**/var/lib/tomcat7/conf/Catalina/europa.daw01.net/**

   2.6. Copia el fichero **/var/lib/tomcat7/conf/Catalina/localhost/manager.xml** a **/var/lib/tomcat7/conf/Catalina/europa.daw01.net** para habilitar *Tomcat Web Manager* el nuevo *Host virtual*.

```
cd /var/lib/tomcat7/conf/Catalina/
sudo cp localhost/manager.xml europa.daw01.net
```

   2.7. Desde **DesarrolloWindosXX** accede a `http://europa.dawXX.net:8080/manager/html` y despliega la aplicación **curso**, Figuras 8.118 y 8.119.

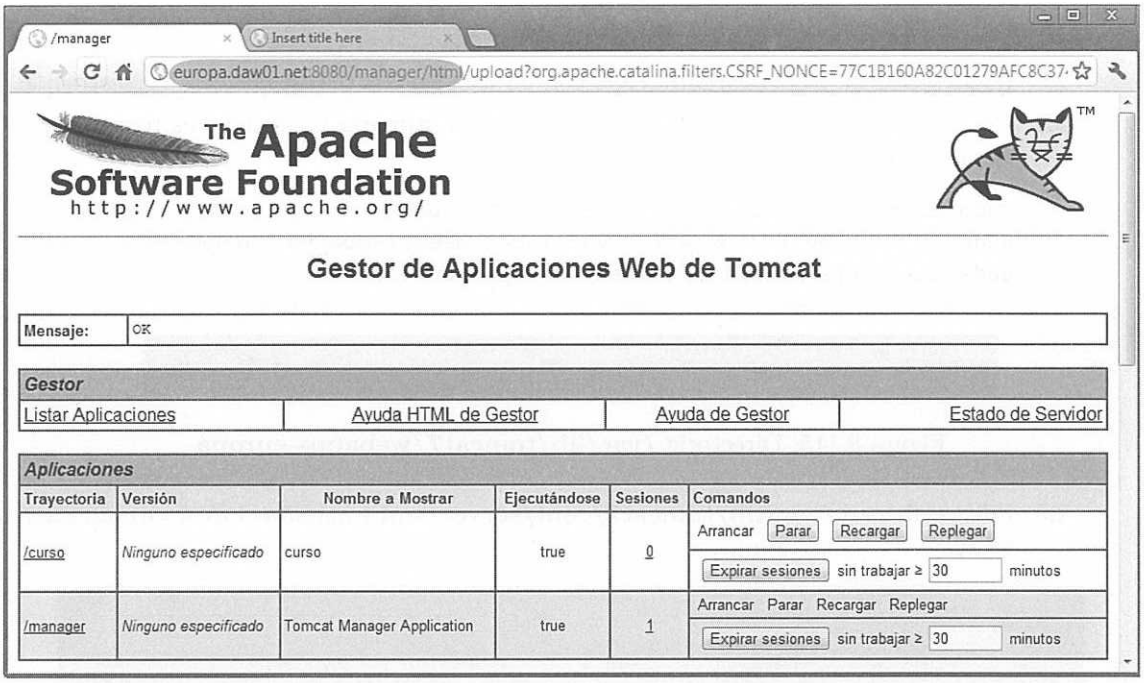


Figura 8.118: Acceso al servidor virtual europa.daw01.net

Figura 8.119: Acceso al servidor virtual europa.daw01.net

3. ***Host* virtual para el dominio** `africa.dawXX.net`

3.1. Inicia sesión en **ServidorLinuxXX** con un usuario con privilegios de administración.

3.2. Crea el directorio **/var/lib/tomcat7/webapps-africa** y asígnale los permisos adecuados, Figura 8.120.

```
sudo mkdir  /var/lib/tomcat7/webapps-africa
sudo chown tomcat7:tomcat7  /var/lib/tomcat7/webapps-africa
sudo chmod 775 /var/lib/tomcat7/webapps-africa
```

```
drwxrwxr-x 7 tomcat7 tomcat7 4096 jun  9 13:46 webapps
drwxrwxr-x 2 tomcat7 tomcat7 4096 jun  9 15:00 webapps-africa
drwxrwxr-x 3 tomcat7 tomcat7 4096 jun  9 14:49 webapps-europa
```

Figura 8.120: Directorio **/var/lib/tomcat7/webapps-africa**

3.3. Edita el fichero **/var/lib/tomcat7/conf/server.xml** y habilita el *Host* virtual, Figura 8.121.

```
<Host name="africa.daw01.net"  appBase="webapps-africa"
      unpackWARs="true" autoDeploy="true">

  <Valve  className="org.apache.catalina.valves.AccessLogValve" directory="logs"
          prefix="africa_access_log." suffix=".txt"
          pattern="%h %l %u %t &quot;%r&quot; %s %b" />

</Host>
```

Figura 8.121: Fichero **/var/lib/tomcat7/conf/server.xml**

3.4. Reinicia *Tomcat.*

3.5. Observa que se ha creado el fichero **/var/lib/tomcat7/conf/Catalina/africa.daw01.net/**, Figura 8.122.

```
alumno@ServidorLinux01:/var/lib/tomcat7/conf/Catalina$ ls -l
total 12
drwxr-xr-x 2 tomcat7 tomcat7 4096 jun  9 15:03 africa.daw01.net
drwxr-xr-x 2 tomcat7 tomcat7 4096 jun  9 14:42 europa.daw01.net
drwxrwxr-x 2 root    tomcat7 4096 jun  4 16:45 localhost
alumno@ServidorLinux01:/var/lib/tomcat7/conf/Catalina$
```

Figura 8.122: Directorio**/var/lib/tomcat7/conf/Catalina/africa.daw01.net/**

3.6. Copia el fichero **/var/lib/tomcat7/conf/Catalina/localhost/manager.xml** a **/var/lib/tomcat7/conf/Catalina/africa.daw01.net** para habilitar *Tomcat Web Manager* el nuevo *Host virtual.*

```
cd /var/lib/tomcat7/conf/Catalina/
sudo cp localhost/manager.xml africa.daw01.net
```

3.7. Desde **DesarrolloWindosXX** accede a `http://africa.dawXX.net:8080/manager/html` y despliega la aplicación **compras**, Figuras 8.123 y 8.124.

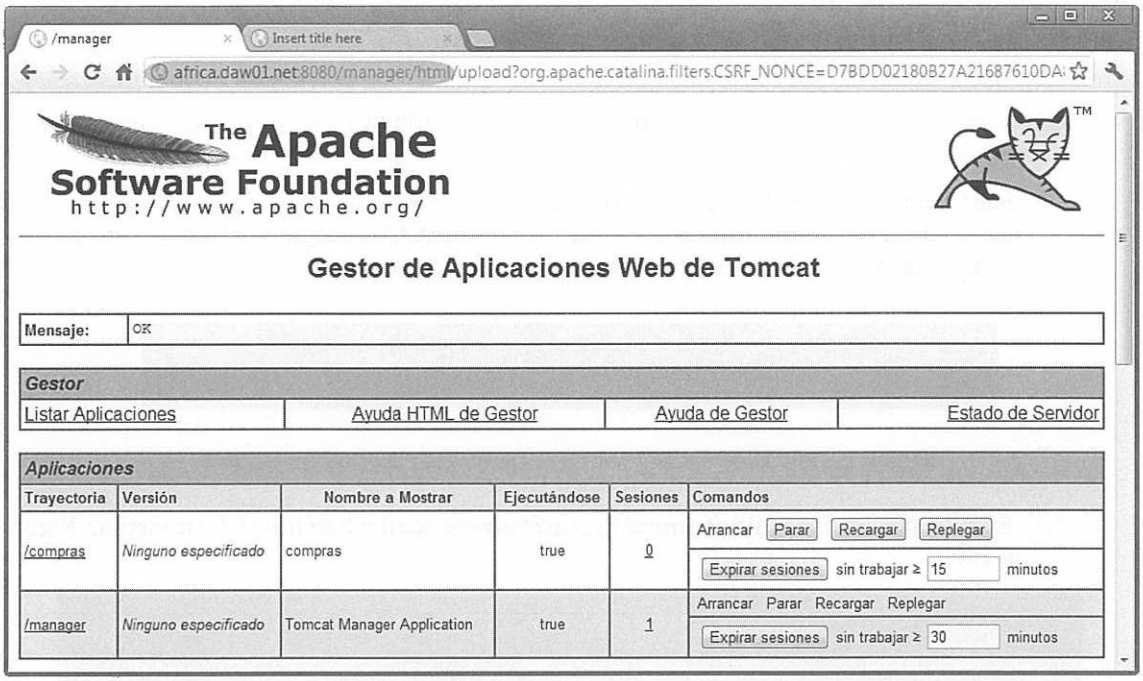


Figura 8.123: Acceso al servidor virtual africa.daw01.net

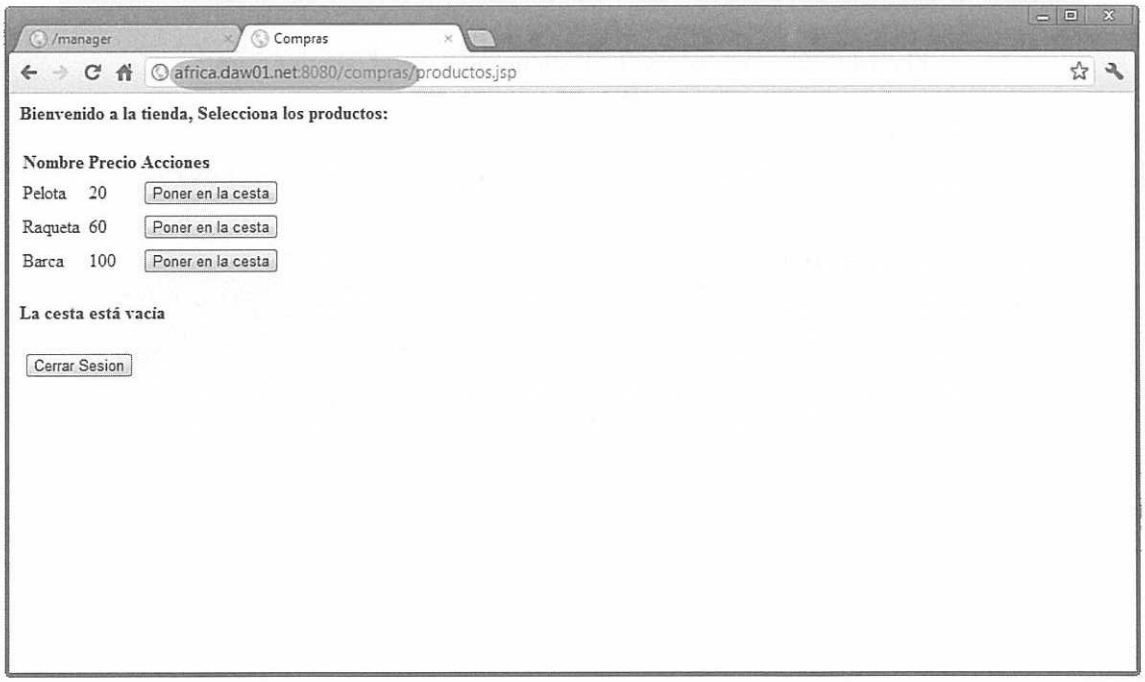


Figura 8.124: Acceso al servidor virtual africa.daw01.net

8.17. *Tomcat Host Manager*

Configura el usuario **tomcat** con el rol **admin-gui** para que pueda utilizar el acceso web y administrar *Host* Virtuales.

1. Instalación.

   1.1. Inicia sesión en **ServidorLinuxXX** con un usuario con privilegios de administración.

   1.2. En la **práctica 8.6** se instaló el **tomcat7-admin** que incluía la aplicación **host-manager**.

   1.3. Consulta el contenido del directorio **/var/lib/tomcat7/conf/Catalina/localhost/** y observa que existe el fichero **host-manager.xml**.

2. Acceso.

   2.1. Desde el equipo **DesarrolloW7XX** accede `http://192.168.1.X7:8080/host-manager` y observa que no puedes acceder. Pincha en **Cancelar** y observa las instrucciones, Figura 8.125.

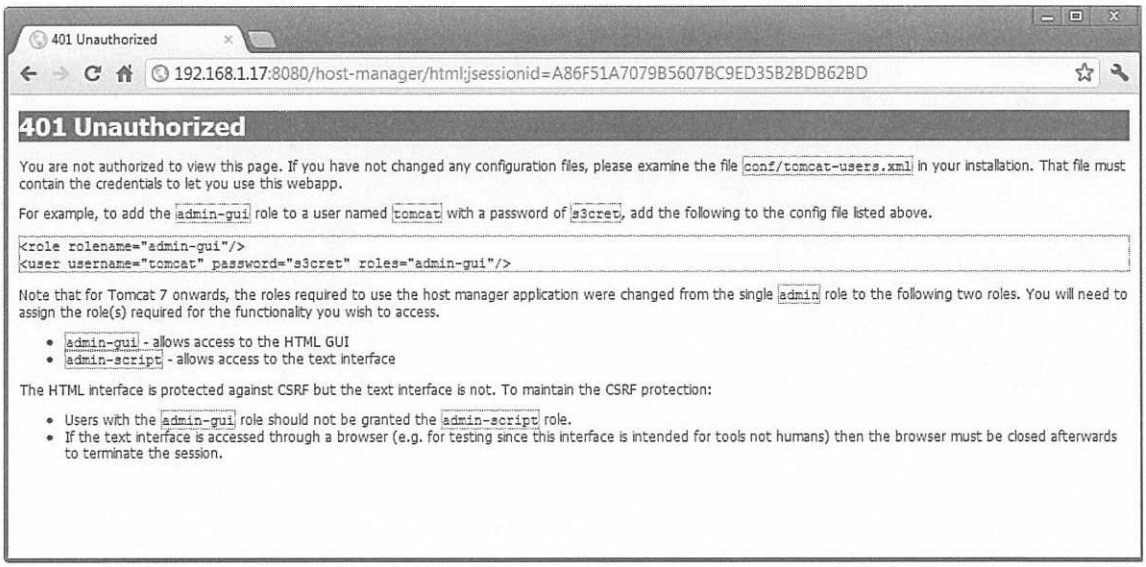


Figura 8.125: Acceso prohibido al *hots-manager*

   2.2. En **ServidorLinuxXX** edita el fichero **/var/lib/tomcat7/conf/tomcat-users.xml/** y realiza la configuración que se muestra en la Figura 8.126.

```
<role rolename="manager-gui"/>
<role rolename="manager-script"/>
<role rolename="admin-gui"/>
<role rolename="curso"/>
<user username="tomcat" password="despliegue" roles="manager-gui, admin-gui"/>
<user username="tomcatscript" password="despliegue" roles="manager-script"/>
<user username="alumno" password="alumno" roles="curso"/>
<user username="profesor" password="profesor" roles="curso"/>
```

Figura 8.126: Fichero **/var/lib/tomcat7/conf/tomcat-users.xml/**

2.3. Reinicia *Tomcat*.

2.4. Desde el equipo **DesarrolloW7XX** accede `http://192.168.1.X7:8080/host-manager` e introduce el usuario creado, Figuras 8.127 y 8.128.

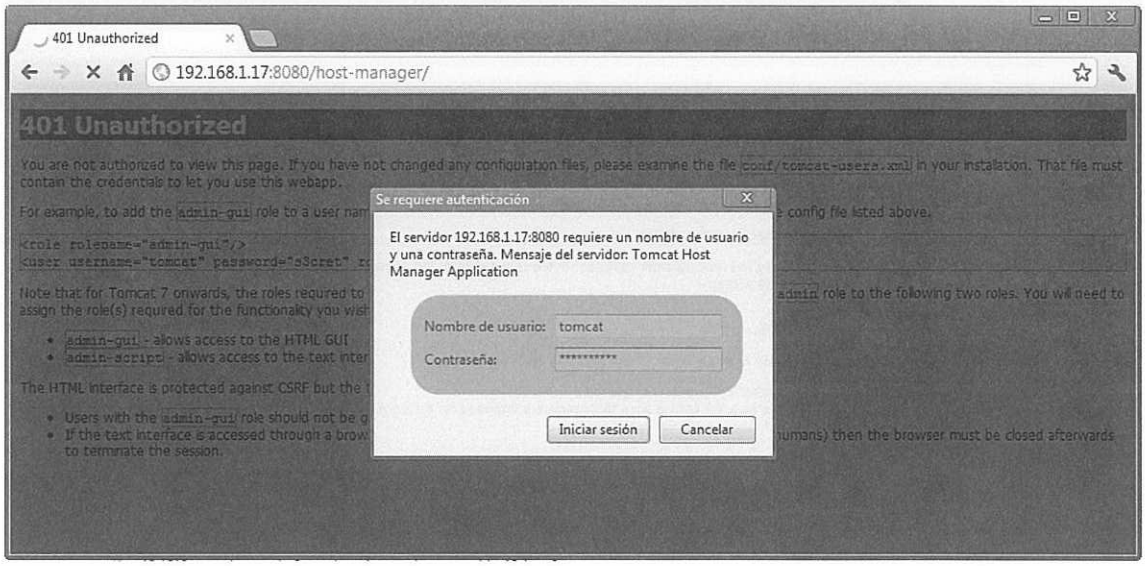


Figura 8.127: Acceso a *Tomcat Host Manager*

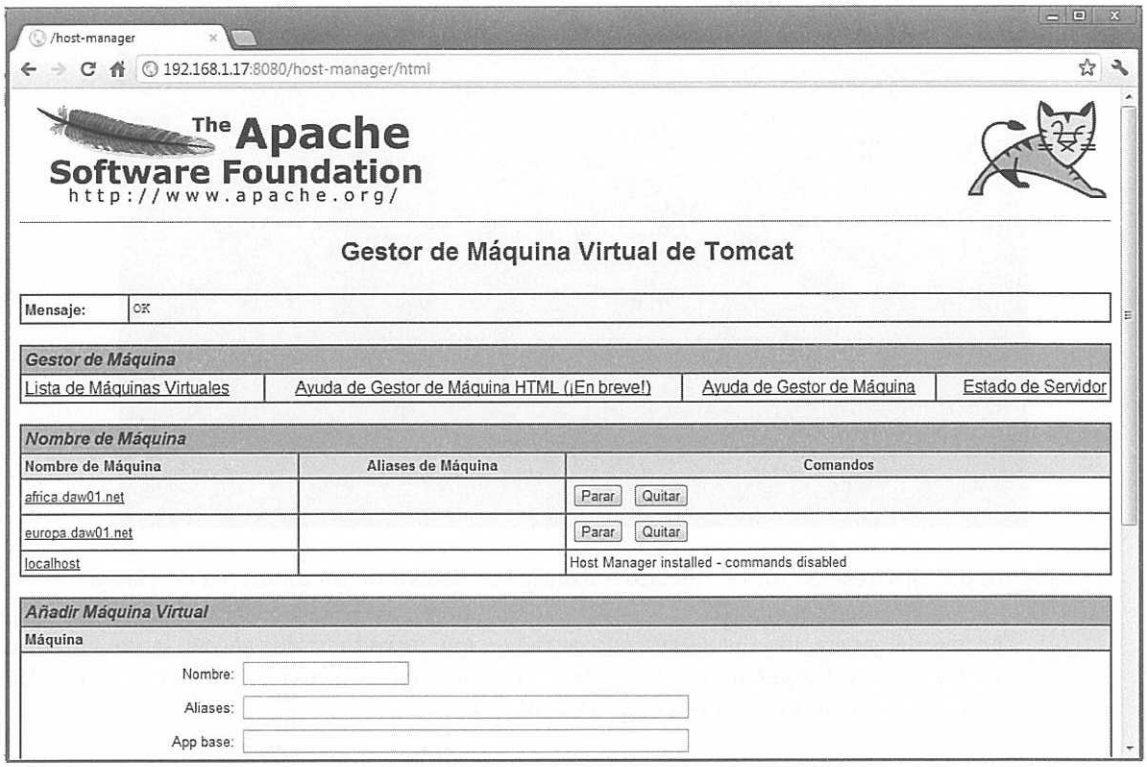


Figura 8.128: Acceso a *Tomcat Host Manager*

8.18. Seguridad SSL

- Habilita SSL en el *Host* por defecto de *Tomcat* (*localhost*).
- Configura la aplicación **curso** para que solo permita conexiones https.

1. <u>**Crear un almacén de claves con un certificado SSL**</u>.

   1.1. Inicia sesión en **ServidorLinuxXX** con un usuario con privilegios de administración.

   1.2. Utiliza la herramienta `keytool` para crear un certificado autofirmado y almacenarlo en un almacén de claves JKS.

```
sudo keytool -genkey -alias tomcat -keyalg RSA \
-keystore /var/lib/tomcat7/daw01keystore
```

   Introduce los datos del certificado, Figura 8.129

```
alumno@ServidorLinux01:/var/lib/tomcat7$ sudo keytool -genkey -alias tomcat -key
alg RSA -keystore /var/lib/tomcat7/daw01keystore
Introduzca la contraseña del almacén de claves:
Volver a escribir la contraseña nueva:
¿Cuáles son su nombre y su apellido?
  [Unknown]:  servidorlinux01.daw01.net
¿Cuál es el nombre de su unidad de organización?
  [Unknown]:  tomcat
¿Cuál es el nombre de su organización?
  [Unknown]:  daw01.net
¿Cuál es el nombre de su ciudad o localidad?
  [Unknown]:  Madrid
¿Cuál es el nombre de su estado o provincia?
  [Unknown]:  Madrid
¿Cuál es el código de país de dos letras de la unidad?
  [Unknown]:  ES
¿Es correcto CN=servidorlinux01.daw01.net, OU=tomcat, O=daw01.net, L=Madrid, ST=
Madrid, C=ES?
  [no]:  si

Introduzca la contraseña de clave para <tomcat>
        (INTRO si es la misma contraseña que la del almacén de claves):
alumno@ServidorLinux01:/var/lib/tomcat7$
```

Figura 8.129: Creación un certificado autofirmado dentro de un alamecen de claves

Durante la generación del certificado se tienen que introducir dos claves, la del almacén de claves (**daw1keystore**) y la de las claves asociadas a alias (**tomcat**) creado. En ambos casos pondremos como clave **despliegue**.

2. **Configurar un conector SSL en *Tomcat*.**

   2.1. En **ServidorLinuxXX** edita el fichero **/var/lib/tomcat7/conf/server.xml/** y realiza la configuración que se muestra en la Figura 8.130.

```
<!-- Define a SSL HTTP/1.1 Connector on port 8443
     This connector uses the JSSE configuration, when using APR, the
     connector should be using the OpenSSL style configuration
     described in the APR documentation -->

<Connector
    port="8443"
    protocol="HTTP/1.1"
    SSLEnabled="true"
    maxThreads="150"
    scheme="https" secure="true"
    clientAuth="false"
    sslProtocol="TLS"
    keystoreFile="/var/lib/tomcat7/daw01keystore" keystorePass="despliegue"
    keyAlias="tomcat" keyPass="despliegue"
  />
```

Figura 8.130: Fichero **/var/lib/tomcat7/conf/server.xml/**

   2.2. Reinicia *Tomcat*.

   2.3. Comprueba que el servidor está iniciado y escuchando en los puerto 8080/TCP y 8443/TCP.

```
ps -ef | grep tomcat
netstat -ltn
```

2.4. Desde **DesarrolloW7XX** abre el navegador y establece una conexión a `https://servidorlinux01.daw01.net:8443/curso`, Figuras 8.131, 8.132 y 8.133.

Figura 8.131: Conexión https

Figura 8.132: Conexión https

3. **Configurar la aplicación curso para que solo permita conexiones https.**

3.1. Desde **DesarrolloW7XX** abre el navegador y establece una conexión a `http://servidorlinux01.daw01.net:8080/curso` y observa que permite conexiones http, Figura 8.134.

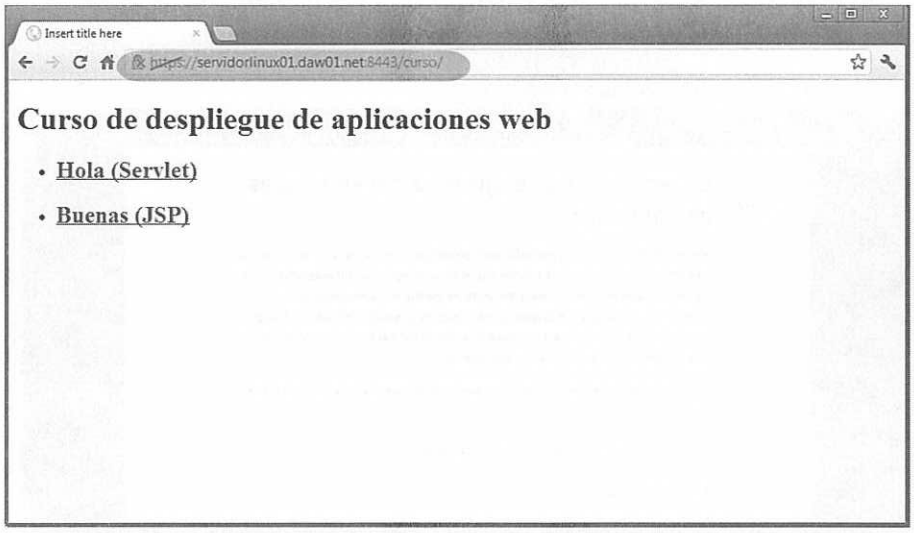


Figura 8.133: Conexión https

Figura 8.134: Conexión http

3.2. Edita el descriptor de despliegue de la aplicación **curso** y realiza la siguiente configuración, Figura 8.135.

```
<security-constraint>
    <web-resource-collection>
        <web-resource-name>MemoryRealm</web-resource-name>
        <url-pattern>/*</url-pattern>
    </web-resource-collection>
    <auth-constraint>
        <role-name>curso</role-name>
    </auth-constraint>
    <user-data-constraint>
        <transport-guarantee>CONFIDENTIAL</transport-guarantee>
    </user-data-constraint>
</security-constraint>
```

Figura 8.135: Fichero **web.xml**

3.3. Elimina la aplicación **curso** del servidor *Tomcat*.

3.4. Genera el fichero **curso.war** con los cambios.

3.5. Despliega el nuevo fichero **curso.war**.

3.6. Desde **DesarrolloW7XX** abre el navegador y establece una conexión a `http://servidorlinux01.daw01.net:8443/curso` y observa la petición se redirige a `https://servidorlinux01.daw01.net:8443/curso`.

8.19. Despliegue en la nube (OpenShift) - 3

En esta práctica vamos a desplegar la aplicación **curso** (la primera versión desarrollada) utilizando las **herramientas en línea de comandos** de *OpenShift*. Se presupone que se han realizado previamente las prácticas 6.8 y 6.9 en donde se ha creado una cuenta en *OpenShift* y se han instalado y configurado las herramientas necesarias.

1. **Introducción al despliegue de aplicaciones JavaEE en *OpenShift***

   OpenShift permite desplegar y ejecutar aplicaciones Java fácilmente. Cosnulta el siguiente enlace `https://www.openshift.com/developers/java` para conocer las tecnologías que soporta. Observa que está disponible el *cartridge* ***Tomcat 6 and 7 (JBoss EWS 1.0 and 2.0)*** para desplegar aplicaciones en *Tomcat*.

   Los siguientes enlaces contiene información útil de cómo desplegar aplicaciones Java:

   - https://www.openshift.com/blogs/run-your-java-tomcat-application-for-free-on-openshifts-paas
   - https://access.redhat.com/site/documentation/en-US/OpenShift_Online/2.0/html/User_Guide/sect-OpenShift-User_Guide-Deploying_JBoss_Applications.html

2. **Crear la aplicación**

   2.1. Inicia sesión en **DesarrolloW7XX** con un usuario con privilegios de administrador.

   2.2. Ejecuta el siguiente comando para mostrar los ***cartridges*** disponibles, Figura 8.136.

   ```
   rhc cartridge list
   ```

```
php-5.3                PHP 5.3                                    web
python-2.6             Python 2.6                                 web
python-2.7             Python 2.7                                 web
python-3.3             Python 3.3                                 web
ruby-1.8               Ruby 1.8                                   web
ruby-1.9               Ruby 1.9                                   web
jbossews-1.0           Tomcat 6 (JBoss EWS 1.0)                   web
jbossews-2.0           Tomcat 7 (JBoss EWS 2.0)                   web
zend-5.6               Zend Server 5.6                            web
diy-0.1                Do-It-Yourself 0.1                         web
10gen-mms-agent-0.1    10gen Mongo Monitoring Service Agent       addon
cron-1.4               Cron 1.4                                   addon
jenkins-client-1.4     Jenkins Client 1.4                         addon
mongodb-2.2            MongoDB NoSQL Database 2.2                 addon
mysql-5.1              MySQL Database 5.1                         addon
```

Figura 8.136: *Cartridges* disponibles

2.3. Ejecuta el siguiente comando para crear la aplicación **curso** con el *cartrige* **jbossews-2.0 (Tomcat7)**. Figura 8.137.

```
rhc app create curso jbossews-2.0
```

Observa la información que ofrece la salida del comando. Puedes observar que se ha creado la aplicación y se ha clonado el repositorio git (recuerda que la práctica anterior lo hicimos nosotros con el comando **git clone**).

2.4. Si accedes a tu cuenta con la consola de administración web puedes observar la información de tu nueva aplicación, Figuras 8.138 y 8.139.

3. **Desplegar la aplicación**

3.1. Accede a **Inicio, Todos los programas, Git, Git Bash** para abrir una consola de Git.

3.2. Accede al directorio de la aplicación (**cd curso**), Figuras 8.139 .

Observa que existen los directorios **src** y **webapps**, y el fichero **pom.xml**. El directorio **src** contiene la estructura en la que incluir el código fuente de la aplicación, y el fichero **pom.xml** es un fichero de configuración de **Maven** (`http://maven.apache.org/`) que describe la estructura de la aplicación y cómo construirla. El directorio **webapps** contendrá la aplicación desplegada.

En nuestro caso vamos a realizar el despliegue de una aplicación ya creada (fichero **war**) por lo que vamos a eliminar el directorio **src** y el fichero **pom.xml**. De esta forma le decimos a *OpenShift* que despliegue la aplicación a partir de nuestro fichero **war** y no a partir del código fuente.

En el siguiente enlace `https://github.com/openshift/origin-server/blob/master/cartridges/openshift-origin-cartridge-jbossews/README.md` puedes leer información sobre las diferentes forma de desplegar la aplicación.

3.3. Sobre el directorio raíz de la aplicación (curso) ejecuta el siguiente comando para borrar del repositorio Git el directorio src y el fichero pom.xml.

```
git rm -r src/ pom.xml
```

3.4. Copia el fichero **curso.war** en el directorio **webapps** con el nombre **ROOT.war**.

```
C:\Users\alumno>rhc app create curso jbossews-2.0
DL is deprecated, please use Fiddle
Application Options
-------------------
Domain:     daw01
Cartridges: jbossews-2.0
Gear Size:  default
Scaling:    no

Creating application 'curso' ... done


Waiting for your DNS name to be available ... done

Cloning into 'curso'...
The authenticity of host 'curso-daw01.rhcloud.com (23.22.151.214)' can't be esta
blished.
RSA key fingerprint is cf:ee:77:cb:0e:fc:02:d7:72:7e:ae:80:c0:90:88:a7.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'curso-daw01.rhcloud.com,23.22.151.214' (RSA) to the
list of known hosts.
remote: Counting objects: 41, done.
remote: Compressing objects: 100% (31/31), done.
rRemote: Total 41 (delta 2), reused 41 (delta 2)eceiving objects:  97% (40/41)
Receiving objects: 100% (41/41), 51.24 KiB | 0 bytes/s, done.

Resolving deltas: 100% (2/2), done.
Checking connectivity... done.

Your application 'curso' is now available.

  URL:        http://curso-daw01.rhcloud.com/
  SSH to:     53a54363e0b8cdfaf8000ba5@curso-daw01.rhcloud.com
  Git remote: ssh://53a54363e0b8cdfaf8000ba5@curso-daw01.rhcloud.com/~/git/curso
.git/
  Cloned to:  C:/Users/alumno/curso

Run 'rhc show-app curso' for more details about your app.

C:\Users\alumno>_
```

Figura 8.137: Creación de la aplicación **curso** con el *cartrige* jbossews-2.0

Figura 8.138: Aplicaciones

Figura 8.139: Aplicación **curso**

```
MINGW32:~/curso
Welcome to Git (version 1.8.3-preview20130601)

Run 'git help git' to display the help index.
Run 'git help <command>' to display help for specific commands.

alumno@DESARROLLOW701 ~
$ cd curso

alumno@DESARROLLOW701 ~/curso (master)
$ ls
README.md  pom.xml  src  webapps

alumno@DESARROLLOW701 ~/curso (master)
$
```

Figura 8.140: Directorio de la aplicación

3.5. Sobre el directorio raíz de la aplicación (curso) ejecuta los siguientes comandos para subir los cambios al repositorio git de *OpenShift*, Figura 8.141.

```
git add -A
git commit -m "Primer despliegue de curso"
git push
```

3.6. Accede a la url de la aplicación **curso** para probar su funcionamiento, Figura 8.141.

OpenShift by Red Hat | Insert title here

curso-daw01.rhcloud.com

Curso de despliegue de aplicaciones web

- Hola (Servlet)
- Buenas (JSP)

Figura 8.141: Acceso a la aplicación **curso**

8.20. Despliegue en la nube (OpenShift) - 4

En esta práctica veremos cómo integrar ***Eclipse*** con ***OpenShift*** para poder desarrollar y desplegar aplicaciones.

1. **Introducción al desarrollo con IDEs en *OpenShift***

   En prácticas anteriores hemos utilizado ***OpenShift Web Interface*** y ***OpenShift Command Line Interface*** para desplegar aplicaciones en *OpenShift*. En esta práctica utilizaremos ***Eclipse***.

   Los siguientes enlaces contienen información útil de cómo integrar un IDE con *OpenShift*:

   - https://www.openshift.com/get-started/
   - https://www.openshift.com/page/install-jboss-developer-studio
   - https://www.openshift.com/blogs/getting-started-with-eclipse-paas-integration

2. **Instalación de *JBoss Tools***

   2.1. Inicia sesión en **DesarrolloW7XX** con un usuario con privilegios de administrador.

   2.2. Inicia *Eclipse* y accede a ***Help, Eclipse Marketplace***. Busca *JBoss Tools*.

   2.3. Instala la versión adecuada para tu **Eclipse**, Figura 8.142.

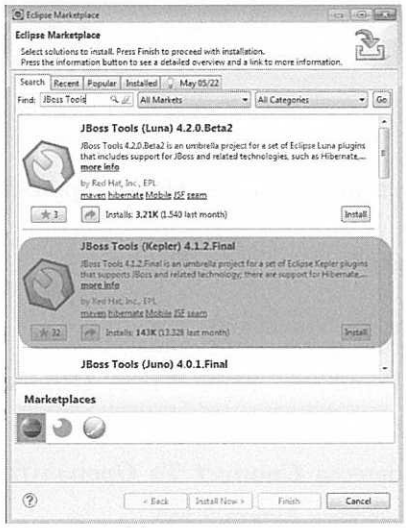


Figura 8.142: Instalación de ***JBoss Tools***

2.4. Selecciona solo ***JBoss OpenShift Tools*** y pincha en ***Next***, Figura 8.143.

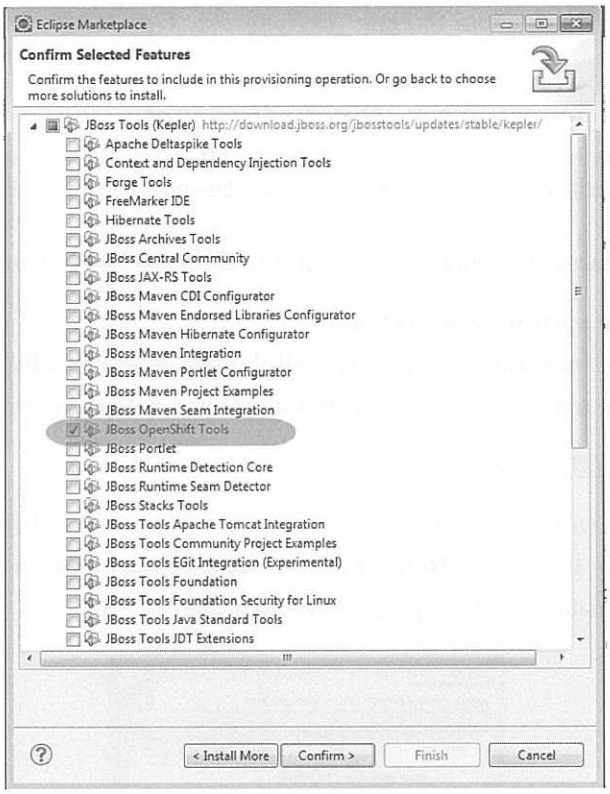


Figura 8.143: Instalación de ***JBoss OpenShift Tools***

2.5. Cuando termine la instalación reinicia ***Eclipse***.

3. **Conexión con la cuenta de *OpenShift***

3.1. En Eclipse accede a ***Windows, Show View, Others***.

3.2. Despliega ***OpenShift*** y selecciona ***OpenShift Explorer***.

3.3. En la parte superior derecha ***Connect To Openshift*** e introduce tus credenciales.

3.4. Accederás a tus aplicaciones, Figura 8.144.

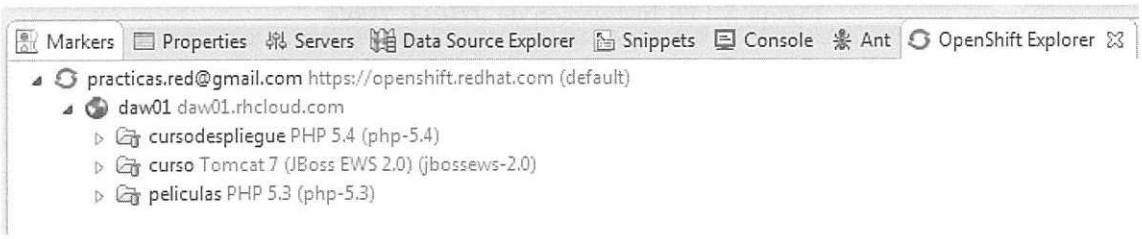


Figura 8.144: Aplicaciones en ***OpenShift***

4. Crear una aplicación

   4.1. Accede a ***File, New, Others.***

   4.2. Despliega ***OpenShift*** y selecciona ***OpenShift Application.*** Pincha en ***Next*** para continuar.

   4.3. Introduce los datos de tu cuenta de ***OpenShift*** y pincha en ***Next*** para continuar.

   4.4. Aquí podrías crear una nueva aplicación o importar alguna de tus aplicaciones, Figura 8.145.

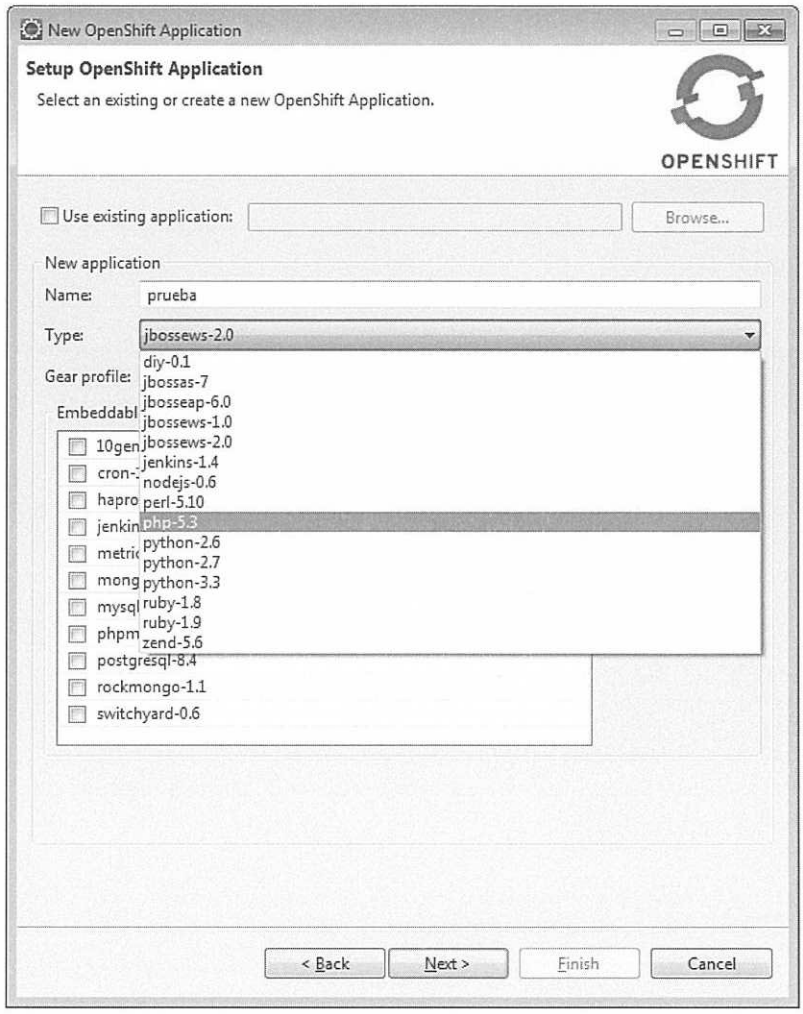


Figura 8.145: Crear una aplicación ***OpenShift***

Capítulo 9

Integración de Apache y Tomcat

9.1. Integración de *Apache* y *Tomcat* usando mod_proxy

- Configura *Apache* y *Tomcat* en **ServidorLinuxXX** como se indica a continuación para posteriormente realizar pruebas de integración.
  - *Apache* solo tendrá habilitados los servidores virtuales **default** y **default-ssl**.
  - *Tomcat* tendrá habilitados los siguientes servidores virtuales:
    - **localhost** con las aplicaciones **Calendar**, **sample**, **docs**, **examples**, **OpenCMS**, **host-manager** y **manager**.
    - **europa.dawXX.net** con las aplicaciones **curso** y **manager**.
    - **africa.dawXX.net** con las aplicaciones **compras** y **manager**.
- Utiliza **mod_proxy** para integrar *Apache* y *Tomcat* en la máquina **ServidorLinuxXX** y prueba las siguientes configuraciones.
  - Configuración 1
    - Permite que la aplicación **sample** del *Host* **localhost** en **Tomcat** sea accesible a través de *Apache*.
  - Configuración 2
    - Permite que las aplicación **curso** del *Host* **europa.dawXX.net** en **Tomcat** sea accesible a través de *Apache*.
    - Permite que las aplicación **compras** del *Host* **africa.dawXX.net** en **Tomcat** sea accesible a través de *Apache*.
  - Configuración 3
    - Configura el servidor virtual https de *Apache* para acceder de forma segura a la aplicación **manager/html** del *Host* **localhost** en **Tomcat**.

1. **Configuración inicial de *Apache*.**

   1.1. Inicia sesión en **ServidorLinuxXX** con un usuario con privilegios de administración.

   1.2. Deshabilita el servidor **seguro** y habilita el servidor **default-ssl**.

   ```
   sudo a2dissite seguro
   sudo a2ensite default-ssl
   ```

   1.3. Reinicia *Apache*.

2. **Configuración inicial de *Tomcat*.**

   2.1. Repliega las aplicaciones **curso** y **compras** del *Host* **localhost**.

   2.2. Asegúrate de que en el *Host* **europa.dawXX.net** está desplegada la aplicación **curso**.

   2.3. Asegúrate de que en el *Host* **africa.dawXX.net** está desplegada la aplicación **compras**.

3. **Habilitar mod_proxy en *Apache*.**

   3.1. Inicia sesión en **ServidorLinuxXX** con un usuario con privilegios de administración.

   3.2. Habilita los módulos **mod_proxy** y **mod_proxy_http** en *Apache*.

   ```
   sudo a2enmod proxy
   sudo a2enmod proxy_http
   ```

```
#ProxyRequests On
<Proxy *>
   AddDefaultCharset off
   Require all granted
#   #Require local
</Proxy>
```

Figura 9.1: Fichero **/etc/apache2/mods-enabled/proxy.conf**

3.3. Edita el fichero **/etc/apache2/mods-enabled/proxy.conf** y habilita las directivas que se muestran en la Figura 9.1

4. **Configuración 1**.

4.1. Reinicia *Apache*.

4.2. Edita el fichero **/etc/apache2/sites-availables/000-default.comf** y añade las siguientes directivas para redirigir las peticiones de la aplicación **sample** desde *Apache* a *Tomcat*, Figura 9.2.

```
<VirtualHost *:80>
        # The ServerName directive sets the request scheme, hostname and port that
        # the server uses to identify itself. This is used when creating
        # redirection URLs. In the context of virtual hosts, the ServerName
        # specifies what hostname must appear in the request's Host: header to
        # match this virtual host. For the default virtual host (this file) this
        # value is not decisive as it is used as a last resort host regardless.
        # However, you must set it for any further virtual host explicitly.
        #ServerName www.example.com

        ServerAdmin webmaster@localhost
        DocumentRoot /var/www/html

        ProxyPreserveHost on
        ProxyPass /sample http://localhost:8080/sample
        ProxyPassReverse /sample http://localhost:8080/sample
```

Figura 9.2: Fichero **/etc/apache2/sites-availables/000-default.conf**

4.3. Reinicia *Apache*.

4.4. Edita el fichero **/var/lib/tomcat7/conf/server.xml/**.En este caso el conector HTTP recibe las peticiones de Apache por HTTP, hay que indicar el puerto del proxy usado en *Apache*, Figura 9.3.

```
<Connector port="8080" protocol="HTTP/1.1"
           connectionTimeout="20000"
           URIEncoding="UTF-8"
           redirectPort="8443"
           proxyPort="80"
    />
```

Figura 9.3: Fichero **/var/lib/tomcat7/conf/server.xml/**

4.5. Recuerda que habiamos configurado un *Valve* que limitaba las conexiones al *Host localhost* solo desde `192.168.1.16`. Coméntalo para que *Apache* pueda realizarle peticiones al *Host localhost* de *Tomcat*, Figura 9.4.

```
<Host name="localhost"  appBase="webapps"
      unpackWARs="true" autoDeploy="true">

  <!-- SingleSignOn valve, share authentication between web applications
       Documentation at: /docs/config/valve.html -->
  <!--
  <Valve className="org.apache.catalina.authenticator.SingleSignOn" />
  -->

  <!-- Access log processes all example.
       Documentation at: /docs/config/valve.html
       Note: The pattern used is equivalent to using pattern="common" -->
  <Valve className="org.apache.catalina.valves.AccessLogValve" directory=$
         prefix="localhost_access_log." suffix=".txt"
         pattern="%h %l %u %t &quot;%r&quot; %s %b" />

  <Valve className="net.daw01.valves.LogginValve" />

  <!-- <Valve className="org.apache.catalina.valves.RemoteAddrValve"
         allow="192.168.1.16" /> -->

</Host>
```

Figura 9.4: Fichero **/var/lib/tomcat7/conf/server.xml/**

4.6. Reinicia *Apache* y *Tomcat*.

4.7. Desde **DesarrolloW7XX** abre el navegador y establece una conexión a `http://192.168.1.X7/sample/` o `http://servidorlinuxXX.dawXX.net/sample/`, Figura 9.5

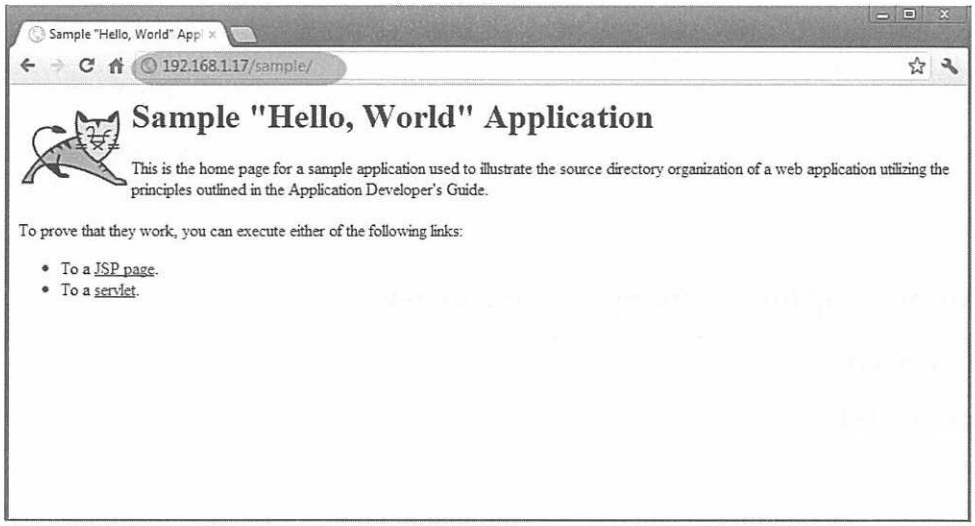


Figura 9.5: Conexión a la aplicación **sample**

5. **Configuración 2.**

   5.1. Edita el fichero **/etc/apache2/sites-enable/000-default.conf** y añade las siguientes directivas para redirigir las peticiones de las aplicaciones **curso** y **compras** desde *Apache* a *Tomcat*, Figura 9.6. La directiva `ProxyPreserveHost on` permite que se mantenga el nombre DNS de la petición que se redirige a Tomcat, en lugar de cambiarlo por el definido en la directiva `ProxyPass`.

```
<VirtualHost *:80>
        # The ServerName directive sets the request scheme, hostname and port that
        # the server uses to identify itself. This is used when creating
        # redirection URLs. In the context of virtual hosts, the ServerName
        # specifies what hostname must appear in the request's Host: header to
        # match this virtual host. For the default virtual host (this file) this
        # value is not decisive as it is used as a last resort host regardless.
        # However, you must set it for any further virtual host explicitly.
        #ServerName www.example.com

        ServerAdmin webmaster@localhost
        DocumentRoot /var/www/html

        ProxyPreserveHost on
        ProxyPass /sample http://localhost:8080/sample
        ProxyPassReverse /sample http://localhost:8080/sample
        ProxyPass /curso http://localhost:8080/curso
        ProxyPassReverse /curso http://localhost:8080/curso
        ProxyPass /compras http://localhost:8080/compras
        ProxyPassReverse /compras  http://localhost:8080/compras
```

Figura 9.6: Fichero **/etc/apache2/sites-availables/000-default.conf**

5.2. Reinicia *Apache*.

5.3. Desde **DesarrolloW7XX** abre el navegador y establece conexiónes a `http://europa.dawXX.net/curso` y `http://africa.dawXX.net/compras`, Figuras 9.7 y 9.8.

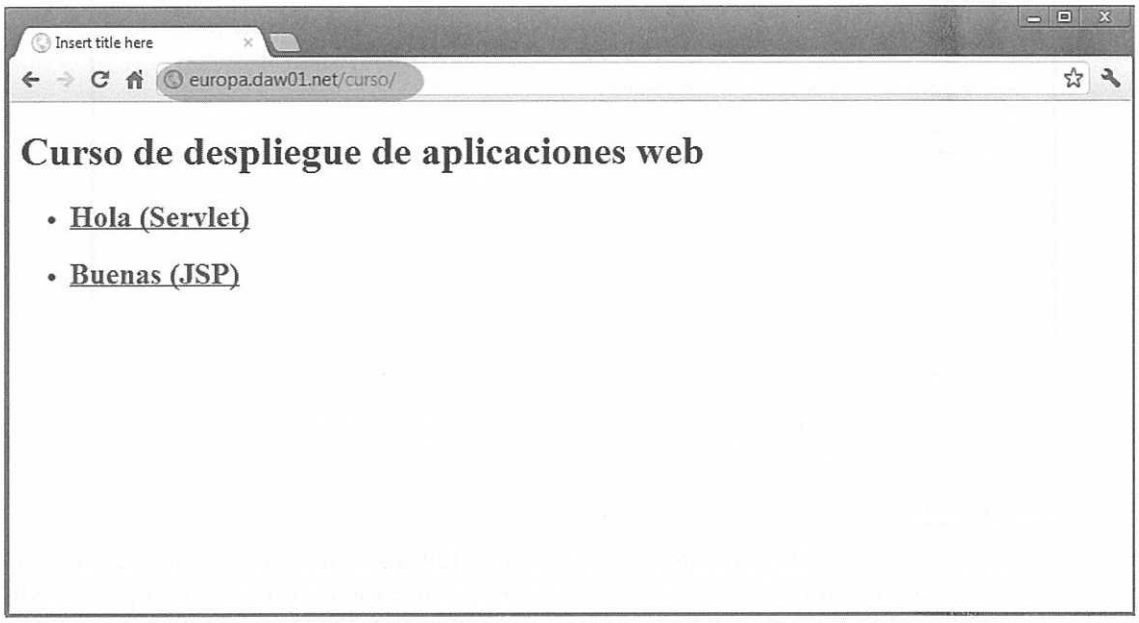


Figura 9.7: Conexión a la aplicación **curso**

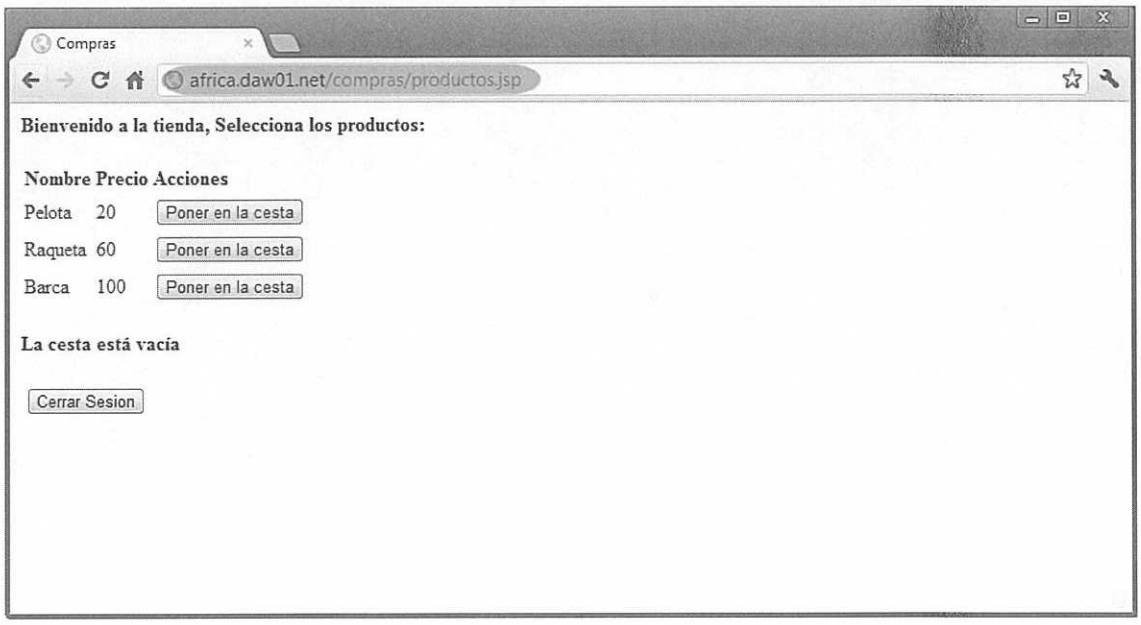


Figura 9.8: Conexión a la aplicación **compras**

6. **Configuración 3.**

   6.1. Edita el fichero **/etc/apache2/sites-availables/default-ssl.conf** y añade las siguientes directivas (observa que la comunicación entre *Apache* y *Tomcat* es http), Figura 9.9.

```
<IfModule mod_ssl.c>
        <VirtualHost _default_:443>
                ServerAdmin webmaster@localhost

                DocumentRoot /var/www/html

                SSLProxyEngine on
                ProxyPreserveHost on
                ProxyPass /manager/html http://localhost:8080/manager/html
                ProxyPassReverse /manager/html http://localhost:8080/manager/html

                # Available loglevels: trace8, ..., trace1, debug, info, notice, warn,
                # error, crit, alert, emerg.
                # It is also possible to configure the loglevel for particular
                # modules, e.g.
                #LogLevel info ssl:warn

                ErrorLog ${APACHE_LOG_DIR}/error.log
                CustomLog ${APACHE_LOG_DIR}/access.log combined
```

Figura 9.9: Fichero **/etc/apache2/sites-availables/default-ssl.conf**

   6.2. Reinicia *Apache*.

   6.3. Edita el fichero **/var/lib/tomcat7/conf/server.xml/**. Observa que en el conector HTTP está definido el parámetro **redirectPort** que redirige las peticiones HTTPS al puerto 8443, Figura 9.10.

```
<Connector port="8080" protocol="HTTP/1.1"
           connectionTimeout="20000"
           URIEncoding="UTF-8"
           redirectPort="8443"
           proxyPort="80"
       />
```

Figura 9.10: Fichero **/var/lib/tomcat7/conf/server.xml/**

   6.4. En el conector HTTPS (escucha en el puerto 8443), hay que indicar el puerto del proxy usado en *Apache*, Figura 9.11.

```
<Connector
    port="8443"
    protocol="HTTP/1.1"
    SSLEnabled="true"
    maxThreads="150"
    scheme="https" secure="true"
    clientAuth="false"
    sslProtocol="TLS"
    keystoreFile="/var/lib/tomcat7/daw01keystore" keystorePass="despliegue"
    keyAlias="tomcat" keyPass="despliegue"
    proxyPort="443"
  />
```

Figura 9.11: Fichero **/var/lib/tomcat7/conf/server.xml/**

6.5. Reinicia *Tomcat*.

6.6. Desde **DesarrolloW7XX** abre el navegador y establece una conexión a `https://192.168.1.X7/manager/html`, Figura 9.12

Figura 9.12: Conexión https

9.2. Integración de *Apache* y *Tomcat* usando mod_jk

Utiliza **mod_jk** para integrar *Apache* y *Tomcat* en la máquina **ServidorLinuxXX** y permitir que la aplicación **Calendar** del *Host* **localhost** en **Tomcat** sea accesible a través de *Apache*.

1. **Instalación de *mod_jk*.**

   1.1. Inicia sesión en **ServidorLinuxXX** con un usuario con privilegios de administración.

   1.2. Descarga el módulo desde los repositorios oficiales de *Ubuntu*.

   ```
   sudo apt-get update
   sudo apt-get install libapache2-mod-jk
   ```

   1.3. Observa que al instalar el paquete se habilita el módulo.

   1.4. Reinicia *Apache*.

2. **Configuración de mod_jk en *Apache*.**

   2.1. Edita el fichero **/etc/apache2/mods-enabled/jk.conf** y observa las directivas de configuración, Figura 9.13.

```
# limitations under the License.

# Configuration Example for mod_jk
# used in combination with Apache 2.2.x

<IfModule jk_module>

    # We need a workers file exactly once
    # and in the global server
    JkWorkersFile /etc/libapache2-mod-jk/workers.properties

    # Our JK error log
    # You can (and should) use rotatelogs here
    JkLogFile /var/log/apache2/mod_jk.log

    # Our JK log level (trace,debug,info,warn,error)
    JkLogLevel info

    # Our JK shared memory file
    JkShmFile /var/log/apache2/jk-runtime-status
```

Figura 9.13: Fichero **/etc/apache2/mods-enabled/jk.conf**

   2.2. Edita el fichero **/etc/libapache2-mod-jk/workers-properties** y configura la siguiente directiva, Figura 9.14.

```
#
# workers.tomcat_home should point to the location where you
# installed tomcat. This is where you have your conf, webapps and lib
# directories.
#
workers.tomcat_home=/usr/share/tomcat7
```

Figura 9.14: Fichero **/etc/libapache2-mod-jk/workers-properties**

2.3. Observa la configuración de los `workers` (instancias de *Tomcat* en ejecución) definidos, Figura 9.15.

```
#
#------ worker list ----------------------------------------
#----------------------------------------------------------------------
#_
#
# The workers that your plugins should create and work with
#
worker.list=ajp13_worker

#
#------ ajp13_worker WORKER DEFINITION ------------------------------
#----------------------------------------------------------------------
#


#
# Defining a worker named ajp13_worker and of type ajp13
# Note that the name and the type do not have to match.
#
worker.ajp13_worker.port=8009
worker.ajp13_worker.host=localhost
worker.ajp13_worker.type=ajp13
```

Figura 9.15: Fichero **/etc/libapache2-mod-jk/workers-properties**

2.4. Reinicia *Apache*.

3. <u>**Habilitar el conector AJP en *Tomcat***</u>.

3.1. Edita el fichero **/var/lib/tomcat7/conf/server.xml/**. Habilita el conector AJP descomentando las directivas que se muestran en la Figura 9.16.

```
<!-- Define an AJP 1.3 Connector on port 8009 -->

<Connector port="8009" protocol="AJP/1.3" redirectPort="8443" />
```

Figura 9.16: Fichero **/var/lib/tomcat7/conf/server.xml/**

3.2. Reinicia *Tomcat*.

4. **Configuración de *Apache*.**

   4.1. Edita el fichero **/etc/apache2/sites-availables/000-default** y añade las siguientes directivas para redirigir las peticiones de la aplicación **Calendar** desde *Apache* a *Tomcat*, Figura 9.17.

```
<VirtualHost *:80>
        # The ServerName directive sets the request scheme, hostname and port that
        # the server uses to identify itself. This is used when creating
        # redirection URLs. In the context of virtual hosts, the ServerName
        # specifies what hostname must appear in the request's Host: header to
        # match this virtual host. For the default virtual host (this file) this
        # value is not decisive as it is used as a last resort host regardless.
        # However, you must set it for any further virtual host explicitly.
        #ServerName www.example.com

        ServerAdmin webmaster@localhost
        DocumentRoot /var/www/html

        ProxyPreserveHost on
        ProxyPass /sample http://localhost:8080/sample
        ProxyPassReverse /sample http://localhost:8080/sample
        ProxyPass /curso http://localhost:8080/curso
        ProxyPassReverse /curso http://localhost:8080/curso
        ProxyPass /compras http://localhost:8080/compras
        ProxyPassReverse /compras  http://localhost:8080/compras

        JkMount /Calendar ajp13_worker
        JkMount /Calendar/* ajp13_worker
```

Figura 9.17: Fichero **/etc/apache2/sites-availables/000-default**

   4.2. Reinicia *Apache*.

   4.3. Desde **DesarrolloW7XX** abre el navegador y establece una conexión a `http://192.168.1.X7/Calendar/` o `http://servidorlinuxXX.dawXX.net/Calendar/`, Figura 9.18

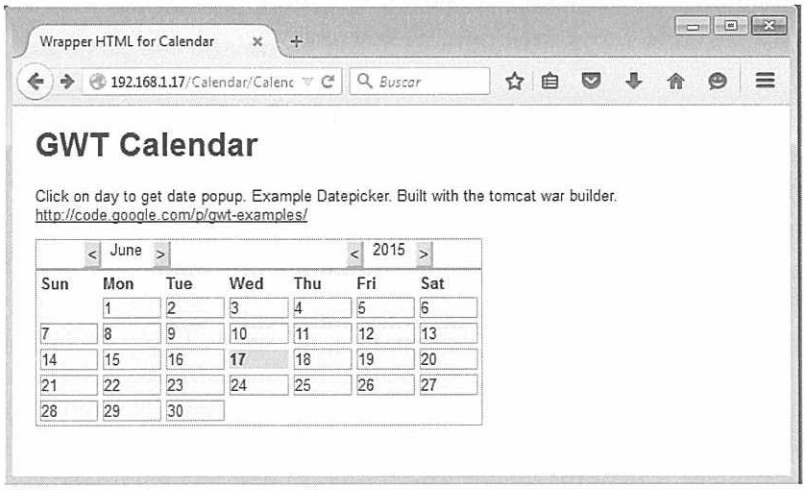


Figura 9.18: Conexión a la aplicación **Calendar**

9.3. Cluster con *Apache* y *Tomcat* (balanceo de carga)

En esta práctica vamos a crear un cluster con dos servidores *Tomcat*. Utilizaremos *Apache* y el módulo **mod_jk** para realizar un balanceo de carga entre los dos servidores (probaremos con la aplicación **Calendar**). El entorno que crearemos es el siguiente:

- *Apache* (**ServidorLinuxXX**) recibirá peticiones HTTP y realizará un balanceo de carga entre dos servidores *Tomcat*. Para ello se usará el módulo **mod_jk**.
- Los servidores *Tomcat* en **ServidorLinuxXX** y *Tomcat* en **ServidorWindowsXX**.

Nota: Lo ideal sería instalar *Apache* en otra máquina diferente.

1. **Configuración de *mod_jk* en *Apache*.**
   1.1. Inicia sesión en **ServidorLinuxXX** con un usuario con privilegios de administración.
   1.2. Edita el fichero **/etc/libapache2-mod-jk/workers-properties** y realiza la siguiente configuración, Figura 9.19.
   1.3. Reinicia *Apache*.
   1.4. Edita el fichero **/etc/apache2/sites-availables/000-default** y añade las siguientes directivas para redirigir las peticiones de la aplicación **Calendar** desde *Apache* a balanceador Figura 9.20.
   1.5. Reinicia *Apache*.
2. **Configuración de *Tomcat* en ServidorLinuxXX (nodo1).**
   2.1. Inicia sesión en **ServidorLinuxXX** con un usuario con privilegios de administración.
   2.2. *Tomcat* en **ServidorLinuxXX** ya está configurado. Revisemos su configuración.
   2.3. Edita el fichero **/var/lib/tomcat7/conf/server.xml/** y observa que está habilitado el conector AJP, Figura 9.21.
   2.4. Verifica que la aplicación **Calendar** está desplegada.
3. **Configuración de *Tomcat* en ServidorWindowsXX (nodo2).**
   3.1. Inicia sesión en **ServidorWindowsXX** con un usuario con privilegios de administración.
   3.2. Descarga *Java SE Development Kit* de `http://www.oracle.com/technetwork/java/javase/downloads/index.html`. En el momento de realizar este manual estaba disponible la versión 7.
   3.3. Instala la versión de JDK descargada previamente.
   3.4. Descarga de `http://tomcat.apache.org/` la versión **32-bit/64-bit Windows Service Installer**.
   3.5. Instala la versión descargada anteriormente, marcando la instalación completa. Configura el usuario **tomcat** para usar la consola de administración web (manager) y observa que se habilita el conector AJP, Figura 9.22. Deja el resto de opciones por defecto.
   3.6. Consulta el contenido del fichero **C:\Program Files\Apache Software Foundation\Tomcat 7.0\conf\server.xml** y observa que está habilitado el conector AJP.
   3.7. Habilita una regla *Firewall* para que se permitan conexiones a a los conectores HTTP (8080/TCP) AJP (8009/TCP) de *Tomcat*.

```
#------ worker list ----------------------------------------
#-----------------------------------------------------------------
#
# The workers that your plugins should create and work with
#
worker.list=loadbalancer
#
#------ ajp13_worker WORKER DEFINITION ----------------------------
#-----------------------------------------------------------------
# Defining a worker named ajp13_worker and of type ajp13
# Note that the name and the type do not have to match.
#
worker.nodo1.port=8009
worker.nodo1.host=localhost
worker.nodo1.type=ajp13
#
# Specifies the load balance factor when used with
# a load balancing worker.
# Note:
#  ----> lbfactor must be > 0
#  ----> Low lbfactor means less work done by the worker.
worker.nodo1.lbfactor=5

worker.nodo2.port=8009
worker.nodo2.host=192.168.1.18
worker.nodo2.type=ajp13
#
# Specifies the load balance factor when used with
# a load balancing worker.
# Note:
#  ----> lbfactor must be > 0
#  ----> Low lbfactor means less work done by the worker.
worker.nodo2.lbfactor=5

#
# Specify the size of the open connection cache.
#worker.ajp13_worker.cachesize
#
#------ DEFAULT LOAD BALANCER WORKER DEFINITION ----------------------
#-----------------------------------------------------------------
# The loadbalancer (type lb) workers perform wighted round-robin
# load balancing with sticky sessions.
# Note:
#  ----> If a worker dies, the load balancer will check its state
#        once in a while. Until then all work is redirected to peer
#        workers.
worker.loadbalancer.type=lb
worker.loadbalancer.balance_workers=nodo1,nodo2
worker.loadbalancer.sticky_session=True
```

Figura 9.19: Fichero **/etc/libapache2-mod-jk/workers-properties**

```
<VirtualHost *:80>
        ServerAdmin webmaster@localhost
        DocumentRoot /var/www

        ProxyPreserveHost on
        ProxyPass /sample http://localhost:8080/sample
        ProxyPassReverse  /sample http://localhost:8080/sample
        ProxyPass /curso http://localhost:8080/curso
        ProxyPassReverse  /curso http://localhost:8080/curso
        ProxyPass /compras http://localhost:8080/compras
        ProxyPassReverse  /compras http://localhost:8080/compras

        JkMount /Calendar loadbalancer
        JkMount /Calendar/* loadbalancer
```

Figura 9.20: Fichero **/etc/apache2/sites-availables/000-default**

```
<!-- Define an AJP 1.3 Connector on port 8009 -->

<Connector port="8009" protocol="AJP/1.3" redirectPort="8443" />
```

Figura 9.21: Fichero **/var/lib/tomcat7/conf/server.xml/**

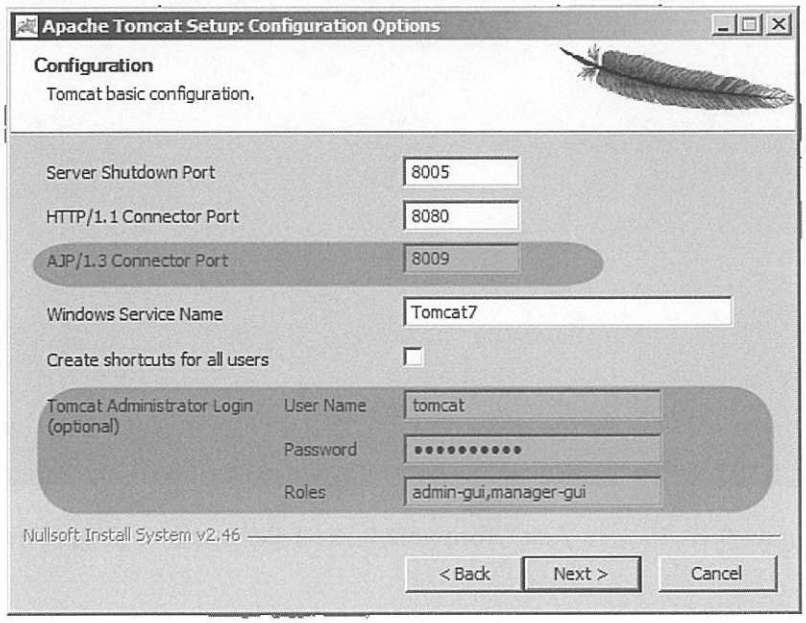


Figura 9.22: Instalación y configuración de *Tomcat* en **ServidorWindowsXX**

a. Accede a **Inicio, Panel de control, Seguridad, Dejar pasar un programa a través de *firewall* de *Windows***.
b. Pincha en **Agregar Programa**.
c. Pincha en **Examina** y selecciona **C:\Program Files\Apache Software Foundation\Tomcat7.0\bin\tomcat7.exe**.
d. Acepta los cambios.

3.8. Despliega la aplicación **Calendar**.

3.9. Verifica que puedes acceder a ella desde **DesarrolloW7XX**.

4. **Prueba la configuración**.

4.1. Desde **DesarrolloW7XX** abre el navegador y establece una conexión a `http://192.168.1.X7/Calendar/` o `http://servidorlinuxXX.dawXX.net/Calendar/`.

4.2. Prueba a parar algunos de los servidores *Tomcat* y verifica que se puede acceder a la aplicación.

4.3. Prueba a iniciar el servidor que has parado previamente y para el otro. Verifica que se puede acceder a la aplicación.

4.4. Consulta los *logs* de acceso de los servidores **Tomcat** y observa que los dos están respondiendo.

5. **Estadísticas de balanceo de carga**.

5.1. Inicia sesión en **ServidorLinuxXX** con un usuario con privilegios de administración.

5.2. Edita el fichero **/etc/libapache2-mod-jk/workers-properties** y realiza la siguiente configuración, Figura 9.23.

```
#
#------ worker list ----------------------------------------
#-----------------------------------------------------------------
#
#
# The workers that your plugins should create and work with
#
worker.list=loadbalancer,stat

#
#------ ajp13_worker WORKER DEFINITION ----------------------------
#-----------------------------------------------------------------
#
worker.stat.type=status
#
```

Figura 9.23: Fichero **/etc/libapache2-mod-jk/workers-properties**

5.3. Reinicia *Apache*.

5.4. Edita el fichero **/etc/apache2/sites-availables/000-default** y añade las siguientes directivas para redirigir las peticiones al ***worker*** que nos permite ver las estadísticas Figura 9.24.

5.5. Reinicia *Apache*.

5.6. Desde **DesarrolloW7XX** abre el navegador y establece una conexión a `http://192.168.1.X7/status/`, Figura 9.25.

```
<VirtualHost *:80>
        ServerAdmin webmaster@localhost
        DocumentRoot /var/www

        ProxyPreserveHost on
        ProxyPass /sample http://localhost:8080/sample
        ProxyPassReverse  /sample http://localhost:8080/sample
        ProxyPass /curso http://localhost:8080/curso
        ProxyPassReverse  /curso http://localhost:8080/curso
        ProxyPass /compras http://localhost:8080/compras
        ProxyPassReverse  /compras http://localhost:8080/compras

        JkMount /Calendar loadbalancer
        JkMount /Calendar/* loadbalancer
        JkMount /status stat
```

Figura 9.24: Fichero **/etc/apache2/sites-availables/000-default**

/manager | /manager | /manager | JK Status Manager

192.168.1.17/status?cmd=show&w=loadbalancer

JK Status Manager for 192.168.1.17:80

Server Version: Apache/2.2.22 (Ubuntu) DAV/2 mod_jk/1.2.32 PHP/5.3.10-1ubuntu3.6 with Suhosin-Patch mod_ssl/2.2.22 OpenSSL/1.0.1

JK Version: mod_jk/1.2.32 ()

Server Time: Fri, 07 Jun 2013 00:19:14 CEST

Unix Seconds: 1370557154

Start auto refresh (every 10 seconds) | Change format XML

[Back to worker list] [Read Only] [Dump] [E=Edit worker, R=Reset worker state, T=Try worker recovery]

[S|E|R] Worker Status for loadbalancer

Type	Sticky Sessions	Force Sticky Sessions	Retries	LB Method	Locking	Recover Wait Time	Error Escalation Time	Max Reply Timeouts [Hide]
lb	True	False	2	Request	Optimistic	60	30	0

Good	Degraded	Bad/Stopped	Busy	Max Busy	Next Maintenance	Last Reset [Hide]
1	0	1	0	2	47/109	482

Balancer Members [Hide]

Name	Type	Hostname	Address:Port	Connection Pool Timeout	Connect Timeout	Prepost Timeout	Reply Timeout	Retries	Recovery Options	Max Packet Size [Hide]
nodo1	ajp13	localhost	127.0.0.1:8009	0	0	0	0	2	0	8192
nodo2	ajp13	192.168.1.18	192.168.1.18:8009	0	0	0	0	2	0	8192

	Name	Act	State	D	F	M	V	Acc	Err	CE	RE	Wr	Rd	Busy	Max	Con	Route	RR	Cd	Rs	LR	LE
[S|E|R]	nodo1	ACT	OK	0	5	1	1	6 (0/sec)	1	0	0	3.1K (6 /sec)	748 (1 /sec)	0	1	4	nodo1		0/0		482	
[S|E|R|T]	nodo2	ACT	ERR	0	5	1	0	5 (0/sec)	3	0	0	1.2K (2 /sec)	305 (0 /sec)	0	2	0	nodo2		47/60		482	Fri, 07 Jun 2013 00:14:53 CEST

Edit this attribute for all members: Activation Go

Figura 9.25: Estadísticas de balanceo de carga

Capítulo 10

Servicios de directorio. LDAP

10.1. Instalación de *OpenLDAP 2.4* en *Linux*

Instala el servidor *OpenLDAP* (`http://www.openldap.org/`) en la máquina **ServidorLinuxXX** y configúralo con las siguientes opciones:

- La entidad raíz de nuestro DIT usará como DN `dc=dawXX, dc=net`
- Establece la clave del usuario **admin** (`dn: cn=admin, dc=dawXX, dc=net`).

1. **Instalación**

   1.1. Inicia una sesión en **ServidorLinuxXX** con un usuario con privilegios de administración.

   1.2. Instala el servidor desde los repositorios oficiales de *Ubuntu*.

   ```
   sudo apt-get update
   sudo apt-get install slapd
   ```

   1.3. Introduce la contraseña del administrador (`dn: cn=admin, dc=dawXX, dc=net`) (usaremos **despliegue**), Figura 10.1.

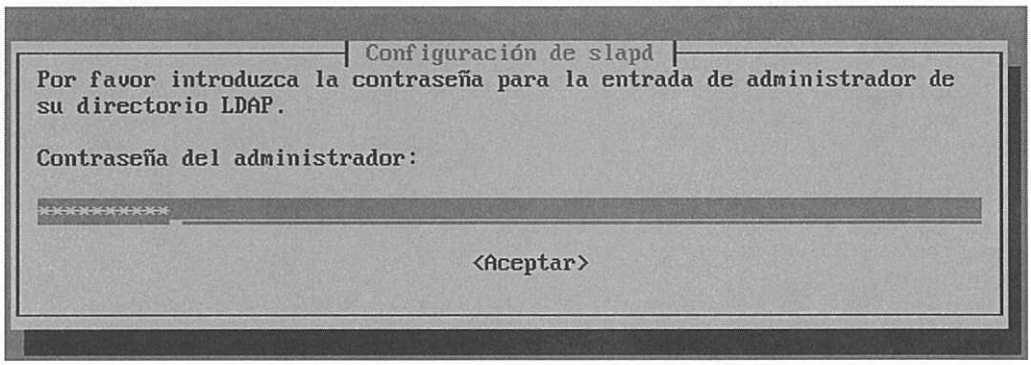


Figura 10.1: Contraseña del usuario admin

   1.4. Vuelve a introducir la contraseña.

   Al instalar el servidor se crean:

   - Los archivos de configuración.
   - Dos DIT
     - Uno con la configuración del servidor (*slapd-conf*).
     - Otro con nuestra configuración (`dc=dawXX dc=net`) (utiliza como **dawXX.net** porque está definido como nombre de dominio en el fichero **/etc/hosts** .
   - Esquemas usados por el servidor.

   1.5. Comprueba que el servidor está iniciado y escuchando peticiones en el puero 389/TCP.

   ```
   ps -ef | grep slapd
   netstat -ltn
   ```

2. **Utilidades *OpenLDAP***

   2.1. Instala el paquete **ldap-utils** que contiene un conjunto de utilidades cliente (`ldapadd, ldapsearch, ldapdelete, ldapmodify`, ...) y servidor (`slapcat, slaptest`).

```
sudo apt-get update
sudo apt-get install ldap-utils
```

3. **Configuración por defecto**

   3.1. Consulta el directorio **/etc/ldap** y observa los ficheros y directorios de configuración.

   3.2. Consulta el directorio **/etc/ldap/slapd.d** donde se almacena el DIT de configuración del servidor, Figura 10.2.

```
/etc/ldap/slapd.d/

    ├── cn=config
    │   ├── cn=module{0}.ldif
    │   ├── cn=schema
    │   │   ├── cn={0}core.ldif
    │   │   ├── cn={1}cosine.ldif
    │   │   ├── cn={2}nis.ldif
    │   │   └── cn={3}inetorgperson.ldif
    │   ├── cn=schema.ldif
    │   ├── olcBackend={0}hdb.ldif
    │   ├── olcDatabase={0}config.ldif
    │   ├── olcDatabase={-1}frontend.ldif
    │   └── olcDatabase={1}hdb.ldif
    └── cn=config.ldif
```

Figura 10.2: DIT de configuración del servidor

   3.3. Consulta el directorio **/etc/ldap/schema** que contiene los *schemas* del servidor (en formato LDIF), Figura 10.3

```
root@ServidorLinux01:/etc/ldap/schema# ls
collective.ldif     cosine.schema          java.ldif       openldap.ldif
collective.schema   duaconf.ldif           java.schema     openldap.schema
corba.ldif          duaconf.schema         ldapns.schema   pmi.ldif
corba.schema        dyngroup.ldif          misc.ldif       pmi.schema
core.ldif           dyngroup.schema        misc.schema     ppolicy.ldif
core.schema         inetorgperson.ldif     nis.ldif        ppolicy.schema
cosine.ldif         inetorgperson.schema   nis.schema      README
root@ServidorLinux01:/etc/ldap/schema#
```

Figura 10.3: Directorio **/etc/ldap/schema**

   3.4. Utiliza el comando `ldapsearch` para mostrar el DIT `dc=dawXX, dc=net`, Figura 10.4.

```
sudo ldapsearch -x -LLL -H ldap://localhost -b "dc=dawXX, dc=net"

-x: Se usa autenticación simple en lugar SASL.
-L : muestra la salida en formato LDIFv1.
```

```
-LL no muestra los comentarios LDIF.
-LLL no muestra la versión de LDIF.
 Si no se usa L muestra la versión extendida de LDIF.
-H: URL del servidor LDAP.Si se omite se usa localhosst.
-b: Entrada del DIT a partir de donde se empieza a buscar.
```

```
dn: dc=daw01,dc=net
objectClass: top
objectClass: dcObject
objectClass: organization
o: daw01.net
dc: daw01

dn: cn=admin,dc=daw01,dc=net
objectClass: simpleSecurityObject
objectClass: organizationalRole
cn: admin
description: LDAP administrator
```

Figura 10.4: DIT dc=dawXX, dc=net

```
Prueba también

ldapsearch -x -H ldap://localhost -b "dc=dawXX, dc=net"
ldapsearch -x -H ldap:/// -b "dc=dawXX, dc=net"
ldapsearch -x -b "dc=dawXX, dc=net"
```

4. **Asistente de configuración**

   4.1. Puedes reconfigurar el servidor ldap ejecutando su asistente de configuración, vamos a probar.

   `sudo dpkg-reconfigure slapd`

   4.2. Selecciona que no quieres omitir la configuración.

   4.3. Como nombre de dominio dns introduce `daw01.net`.

   4.4. Como nombre de la organización introduce `daw01.net`.

   4.5. Introduce la contraseña del administrador (usaremos **despliegue**).

   4.6. Vuelve a introducir la contraseña.

   4.7. Selecciona el motor *HDB* (es el recomendado).

   4.8. Indica que no quieres que se borren los datos si se elimina el servidor.

   4.9. Indica que quieres borrar la base de datos antigua.

   4.10. No permitas el protocolo LDAPv2.

   4.11. Utiliza el comando `ldapsearch` para mostrar el DIT `dc=dawXX, dc=net`. Es igual que antes.

   `ldapsearch -x -LLL -H ldap://localhost -b "dc=dawXX, dc=net"`

10.2. Operaciones sobre LDAP

Realiza las operaciones que se indican en los siguientes apartados sobre el servidor **ldap** instalado en **ServidorLinuxXX** utilizando las utilidades del paquete **ldap-utils**.

1. Añadir entidades

   1.1. Inicia una sesión en **ServidorLinuxXX** con un usuario con privilegios de administración.

   1.2. Crea el fichero **add_entradas.ldif** en formato LDIF, Figura 10.5.

```
# Unidad organizativa usuarios
dn: ou=usuarios,dc=daw01,dc=net
objectClass: organizationalUnit
ou: usuarios

# Unidad organizativa grupos
dn: ou=grupos,dc=daw01,dc=net
objectClass: organizationalUnit
ou: grupos

# Usuario zipi en al unidad organizativa usuarios
dn: uid=zipi,ou=usuarios,dc=daw01,dc=net
objectClass: inetOrgPerson
uid: zipi
sn: zapatilla
cn: zipi
mail: zipi@daw01.net
userPassword: zipi

# Usuario zape en al unidad organizativa usuarios
dn: uid=zape,ou=usuarios,dc=daw01,dc=net
objectClass: inetOrgPerson
uid:zape
sn: zapatilla
cn: zape
mail: zape@daw01.net
userPassword: zape
```

Figura 10.5: Fichero **add_entradas.ldif**

   1.3. Utiliza el comando `ldapadd` para añadir al DIT las entradas definidas en el fichero.

```
ldapadd -x -D cn=admin,dc=dawXX,dc=net -W -f add_entradas.ldif
```

2. Buscar

   2.1. Utiliza el comando `ldapsearch` para consultar todo el DIT.

```
ldapsearch -x -b dc=dawXX,dc=net
```

2.2. Utiliza el comando `ldapsearch` para mostrar el atributo **mail** de todas las entradas

```
ldapsearch -x -b dc=dawXX,dc=net mail
```

2.3. Utiliza el comando `ldapsearch` para mostrar todos los atributos de las entidades **inetOrgPerson**.

```
ldapsearch -x -b dc=dawXX,dc=net "objectClass=inetOrgPerson"
```

2.4. Utiliza el comando `ldapsearch` para mostrar todos los atributos de las entidades **inetOrgPerson** cuyo **cn** es **zape**.

```
ldapsearch -x -b dc=dawXX,dc=net "(&(objectClass=inetOrgPerson)(cn=zape))"
```

3. <u>Modificar entidades</u>

3.1. Crea el fichero **modify_entradas.ldif** en formato LDIF, Figura 10.6.

```
dn: uid=zipi,ou=usuarios,dc=daw01,dc=net
add: mobile
mobile: 999999999

dn: uid=zipi,ou=usuarios,dc=daw01,dc=net
delete:mail

dn:uid=zape,ou=usuarios,dc=daw01,dc=net
changetype: modify
replace: mail
mail: zapezape@daw01.net
```

Figura 10.6: Fichero **modify_entradas.ldif**

3.2. Utiliza el comando `ldapmodify` para modificar las entradas definidas en el fichero.

```
ldapmodify -x -D cn=admin,dc=dawXX,dc=net -W -f modify_entradas.ldif
```

3.3. Utiliza el comando `ldapsearch` para consultar todo el DIT.

```
ldapsearch -x -b dc=dawXX,dc=net
```

4. <u>Eliminar entidades</u>

4.1. Crea el fichero **delete_entradas.ldif** en formato LDIF, Figura 10.7.

```
uid=zape,ou=usuarios,dc=daw01,dc=net
uid=zipi,ou=usuarios,dc=daw01,dc=net
ou=usuarios,dc=daw01,dc=net
ou=grupos,dc=daw01,dc=net
```

Figura 10.7: Fichero **delete_entradas.ldif**

4.2. Utiliza el comando `ldapdelete` para borrar las entradas definidas en el fichero.

```
ldapdelete -x -D cn=admin,dc=dawXX,dc=net -W -f delete_entradas.ldif
```

4.3. Utiliza el comando `ldapsearch` para consultar todo el DIT.

```
ldapsearch -x -b dc=dawXX,dc=net
```

10.3. Clientes LDAP

Instala los siguiente clientes LDAP y prueba a establecer conexiones con el servidor *OpenLDAP* de **ServidorLinuxXX**.

- *JXplorer* (`http://jxplorer.org/`) en **DesarrolloW7XX**.
- *Apache Directory Studio* (`http://directory.apache.org/studio/`) en **DesarrolloW7XX**.
- *phpLDAPadmin* (`http://phpldapadmin.sourceforge.net/`) en **ServidorLinuxXX**.

1. ***JXplorer***

1.1. Inicia una sesión en **DesarrolloW7XX** con un usuario con privilegios de administración.

1.2. Accede a `http://jxplorer.org/downloads` y descarga el instalador para *Windows*.

1.3. Instala el software con las opciones por defecto e inicia el programa.

1.4. Accede a ***File, Connect*** e introduce los datos necesarios para establecer una conexión con el servidor LDAP, Figura 10.8.

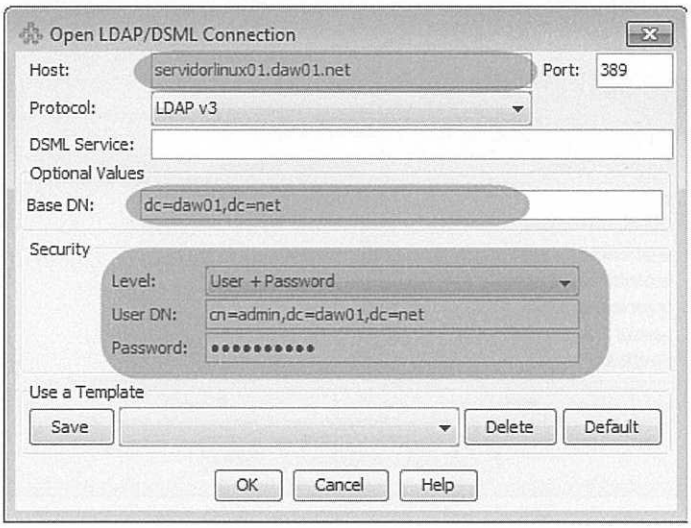


Figura 10.8: *JXplorer*. Conexión al servidor

1.5. Haz clic en OK para establecer la conexión y observa el DIT , Figura 10.9.

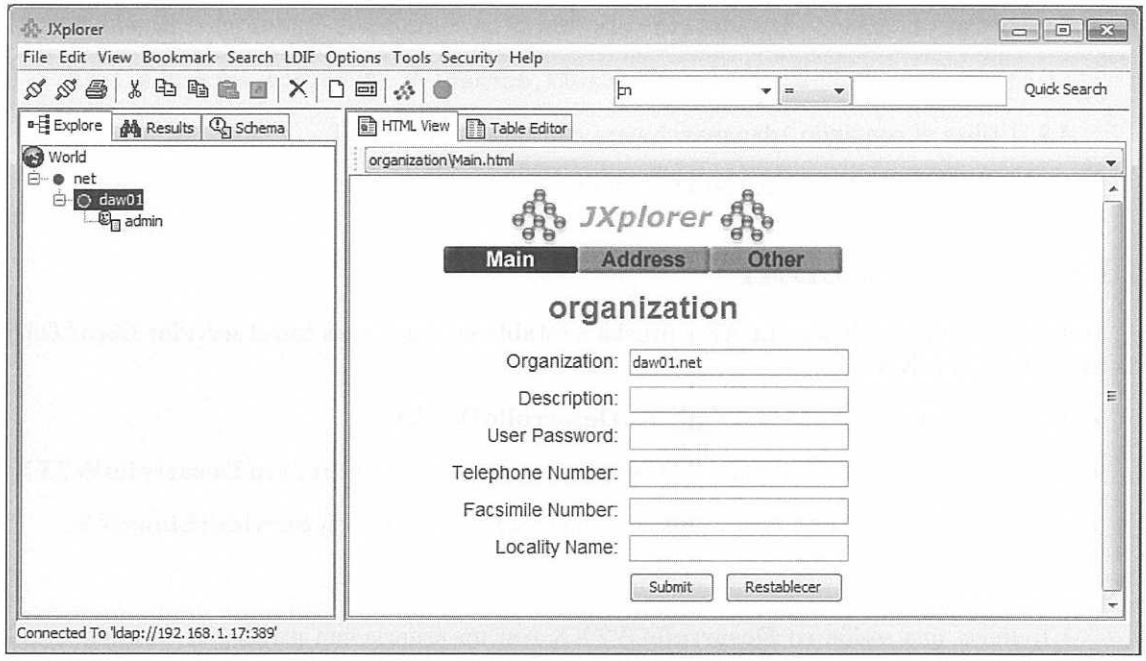


Figura 10.9: *JXplorer*. DIT

1.6. Crea una nueva entrada (**Edit, New**) para la unidad organizativa **usuarios**, Figuras 10.10 y 10.11.

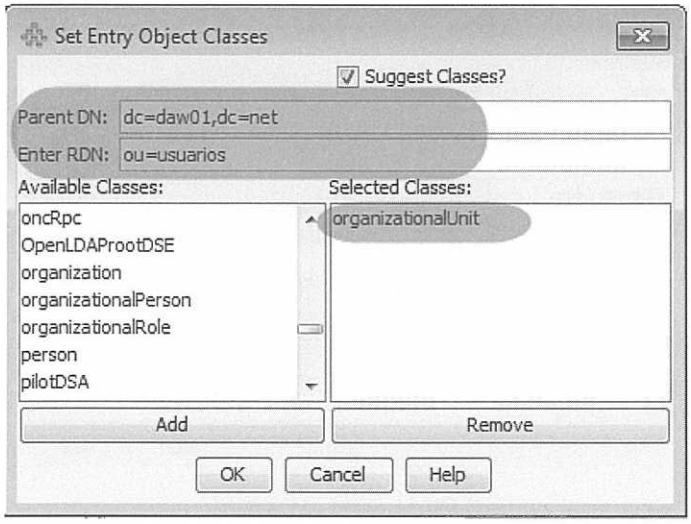


Figura 10.10: *JXplorer*. Crear una entrada

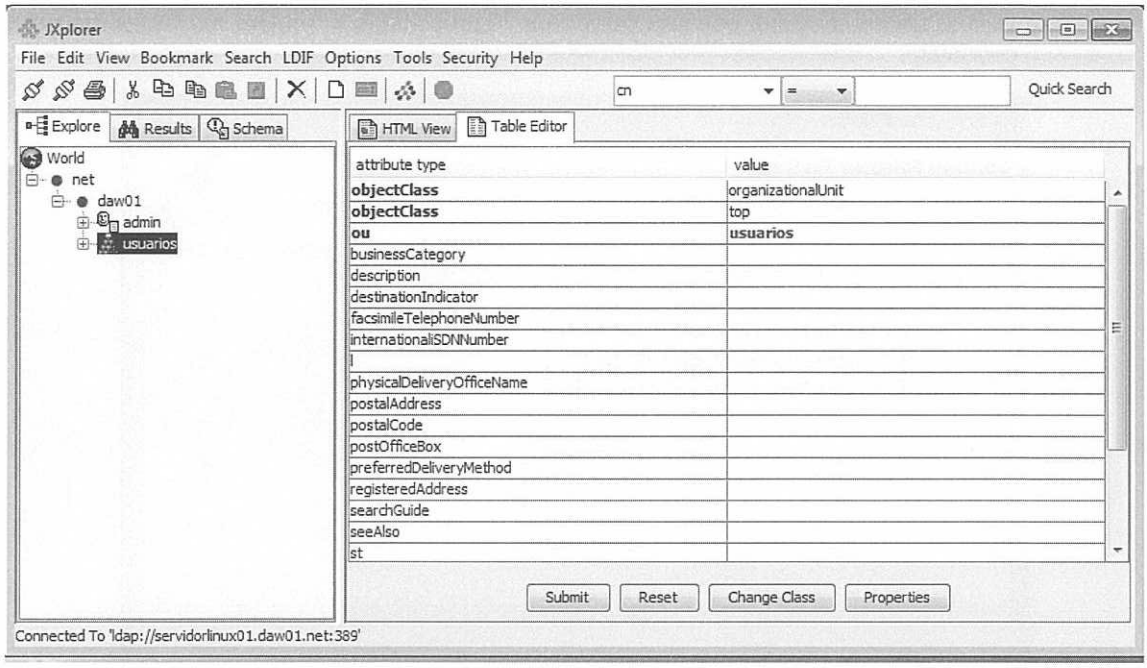


Figura 10.11: *JXplorer*. Entrada creada

Hasta que no pinches en **Submit** no se hacen efectivos los cambios en el servidor.

2. ***Apache Directory Studio***

   Apache Directory Studio consiste en un conjunto de *plugins* para *Eclipse*.

   2.1. Inicia *Eclipse*.

   2.2. Accede a ***Help, Eclipse MarketPlace***.

   2.3. Buscar ***Apache Directory Studio*** e instálalo, Figura 10.12.

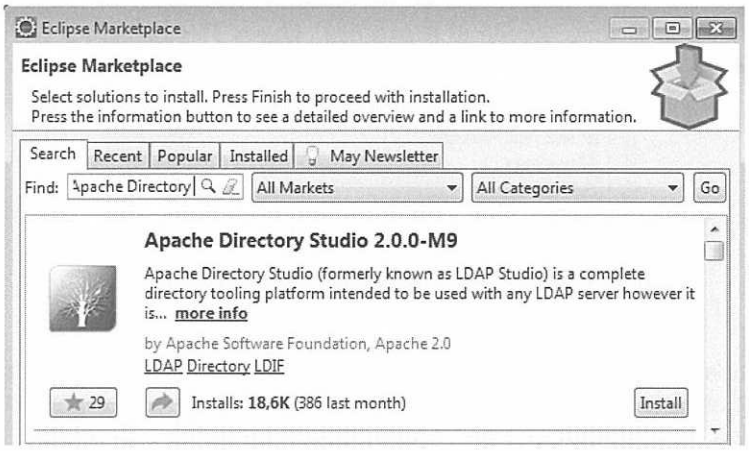


Figura 10.12: Instalación de Apache Directory Studio

2.4. Selecciona los *plug-ins* que nos interesan, Figura 10.13, y pincha en ***Confirm*** para continuar.

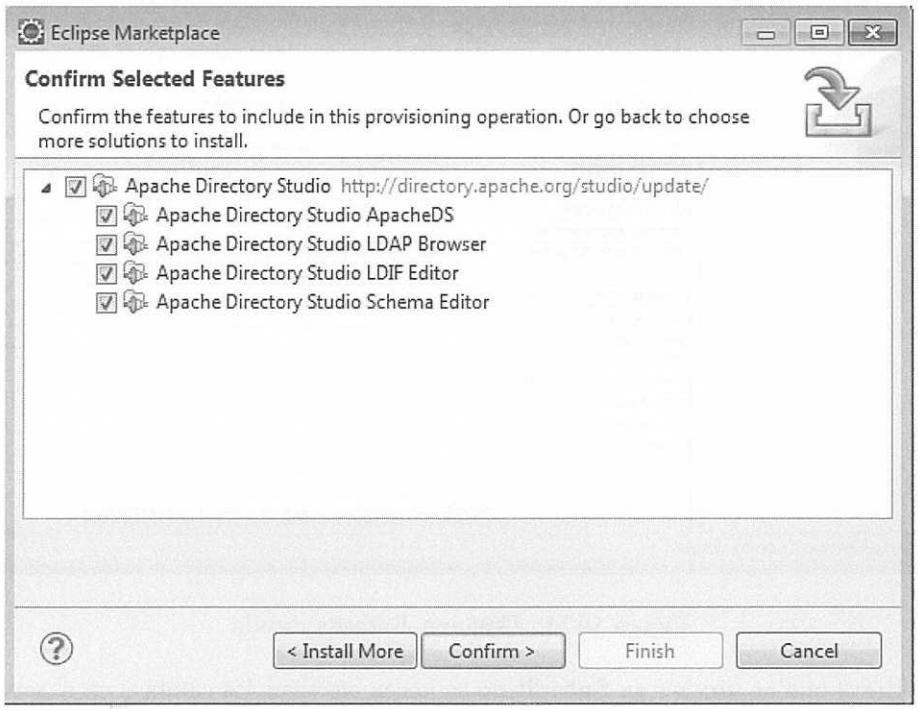


Figura 10.13: Seleccionar plug-ins

2.5. Observa el resumen de los *plugins* a instalar y pincha en ***Next*** para continuar.

2.6. Acepta la licencia y pincha en ***Finish*** para comenzar la instalación.

2.7. Acepta la advertencia de seguridad.

2.8. Pincha en ***Restart Now*** para reiniciar *Eclipse* cuando termine la instalación.

2.9. Cuando inicie *Eclipse* accede a ***Windows, Open Perspective, Other, LDAP*** para abrir la perspectiva de *Apache Directory Studio*, Figuras 10.14 y 10.15.

Figura 10.14: Abrir perspectiva de *Apache Directory Studio*

Si no aparece la perspectiva LDAP consulta este enlace `http://directory.apache.org/studio/installation-in-eclipse.html` y sigue las instrucciones para instalar la última versión y/o paquetes no instalados.

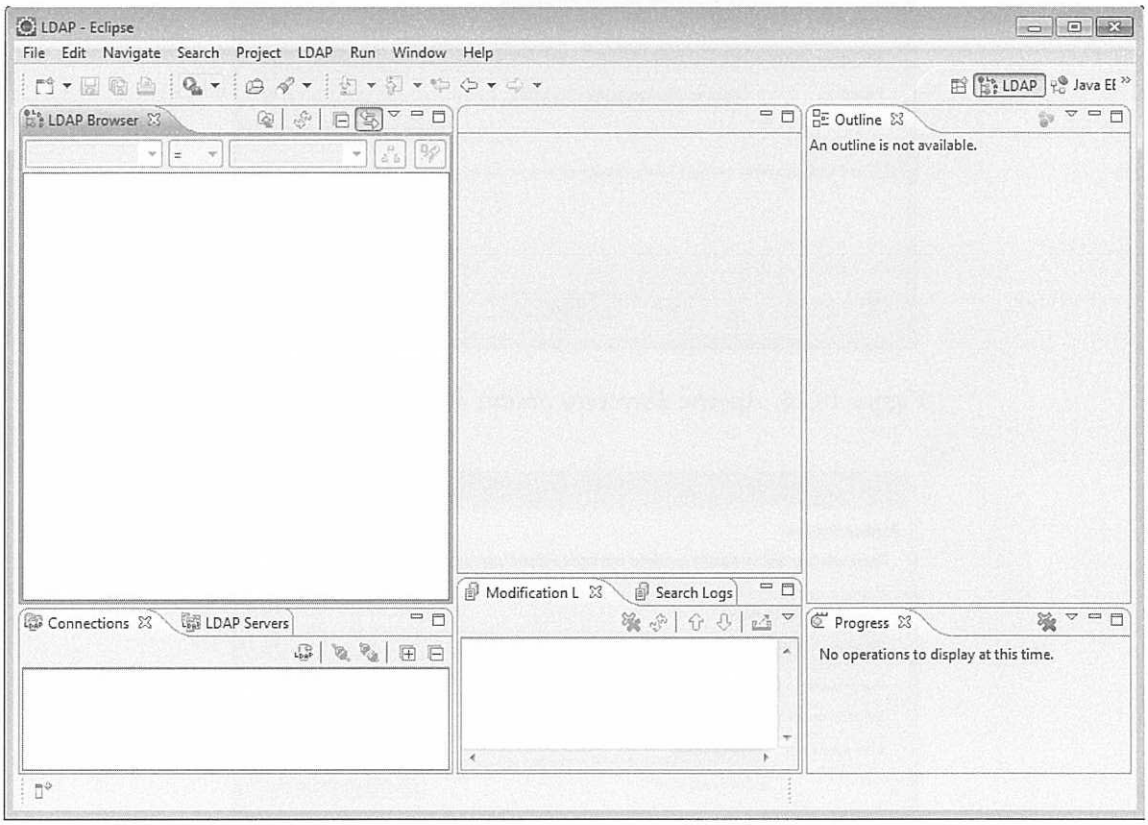


Figura 10.15: Perspectiva de *Apache Directory Studio*

2.10. En la parte inferior izquierda crea una nueva conexión (***New Connection***) e introduce los datos necesarios para establecer una conexión con el servidor LDAP, Figuras 10.16, 10.17, 10.18 y 10.19.

Figura 10.16: *Apache Directory Studio.* Conexión al servidor

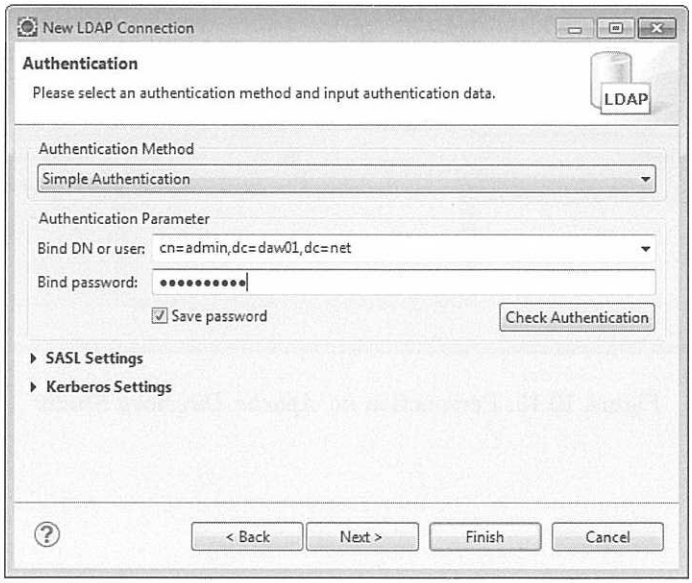


Figura 10.17: *Apache Directory Studio.* Conexión al servidor

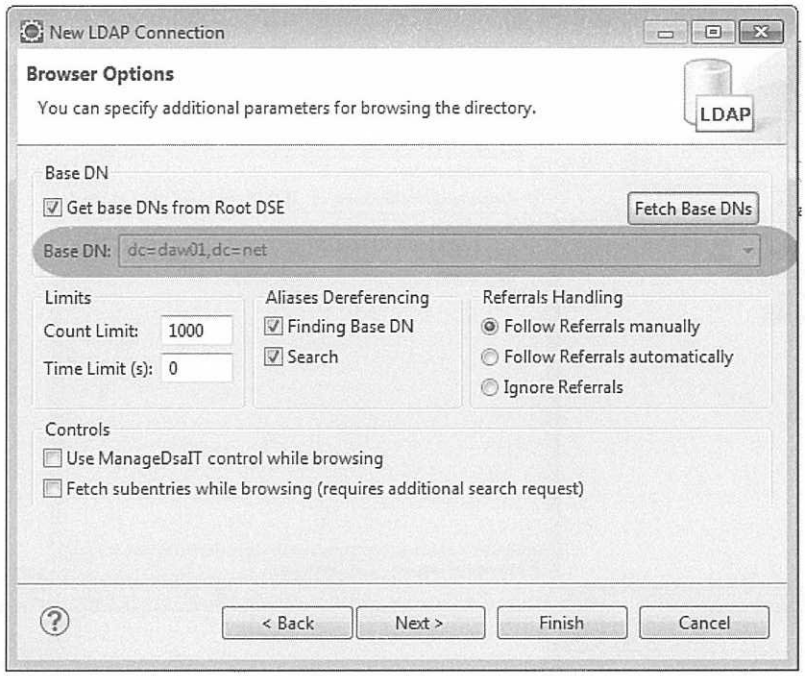


Figura 10.18: *Apache Directory Studio.* Conexión al servidor

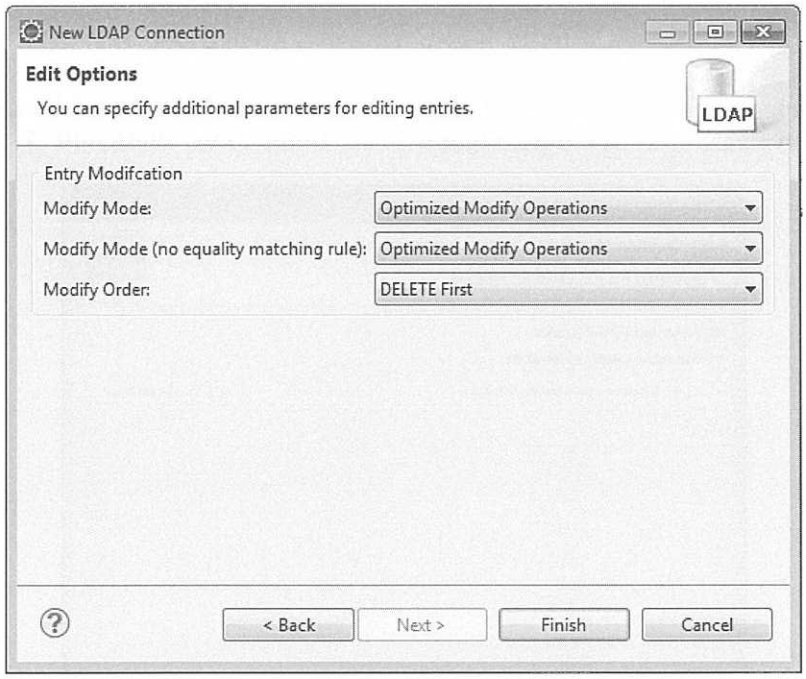


Figura 10.19: *Apache Directory Studio.* Conexión al servidor

2.11. Observa el DIT, Figura 10.20.

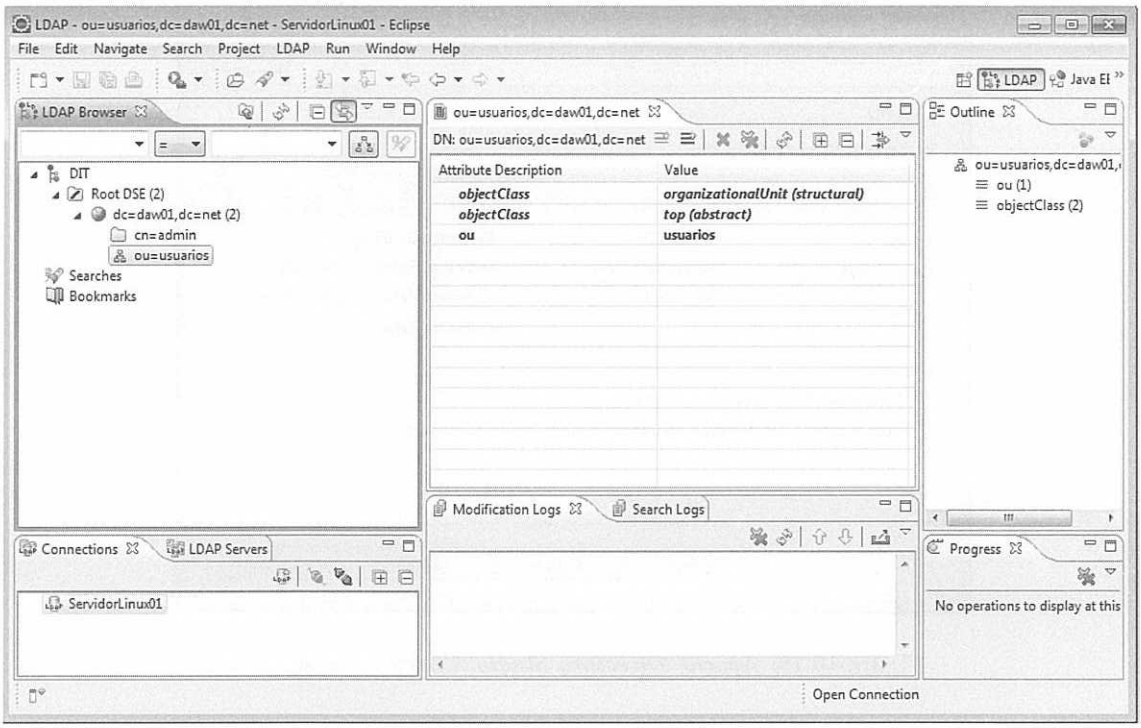


Figura 10.20: *Apache Directory Studio.* DIT

2.12. Crea una nueva entrada (sobre la entrada usuarios, botón derecho del ratón, **New, New Entry**) para el usuario **zipi**, Figuras 10.21, 10.22, 10.23, 10.24 y 10.25.

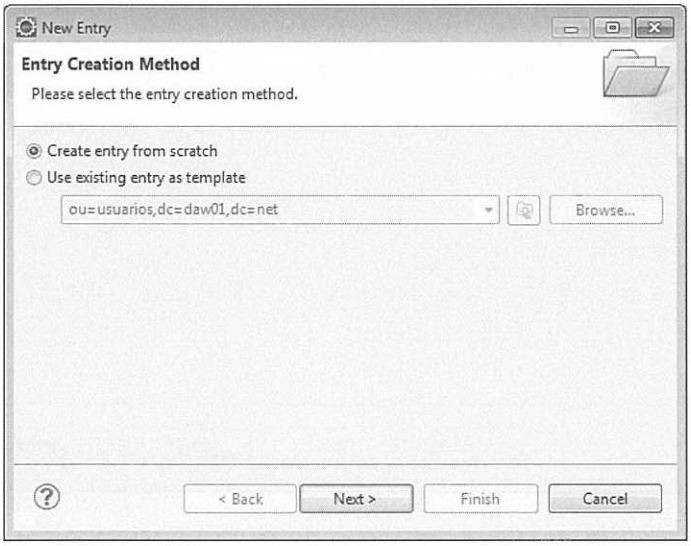


Figura 10.21: *Apache Directory Studio.* Crear una entrada

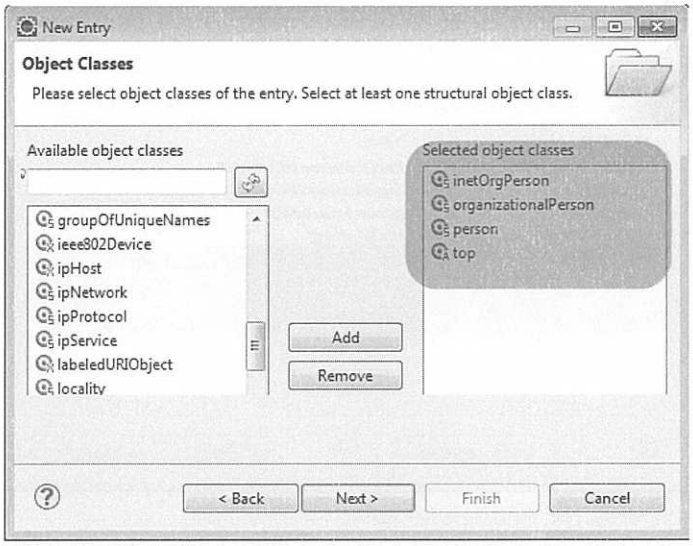


Figura 10.22: *Apache Directory Studio.* Crear una entrada

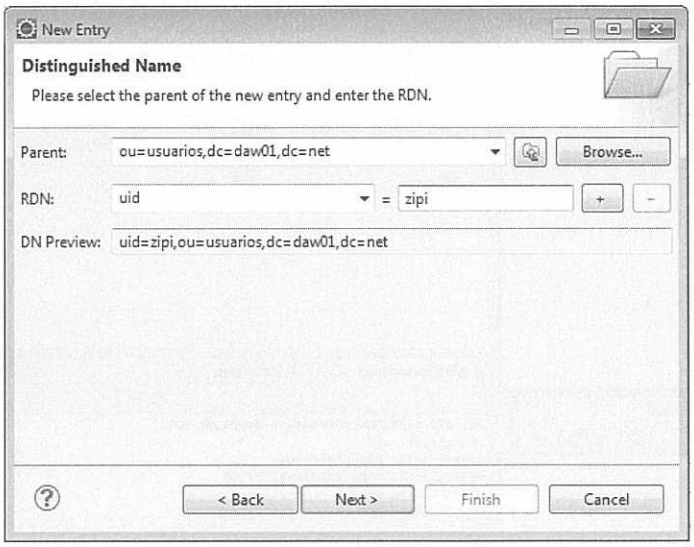


Figura 10.23: *Apache Directory Studio.* Crear una entrada

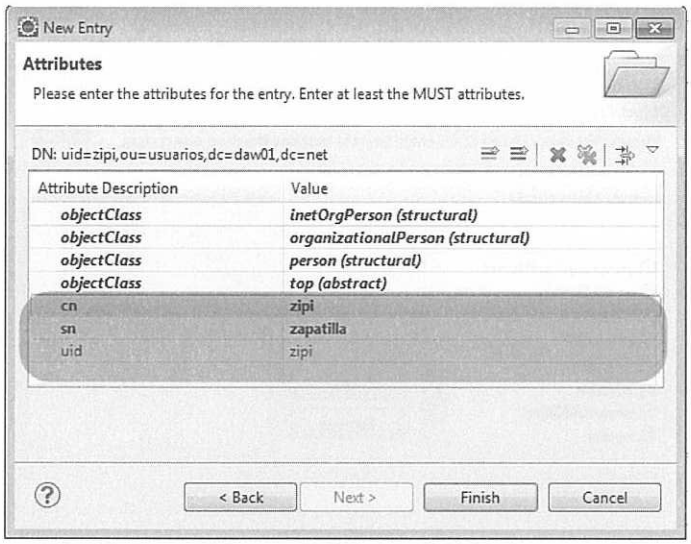


Figura 10.24: *Apache Directory Studio.* Crear una entrada

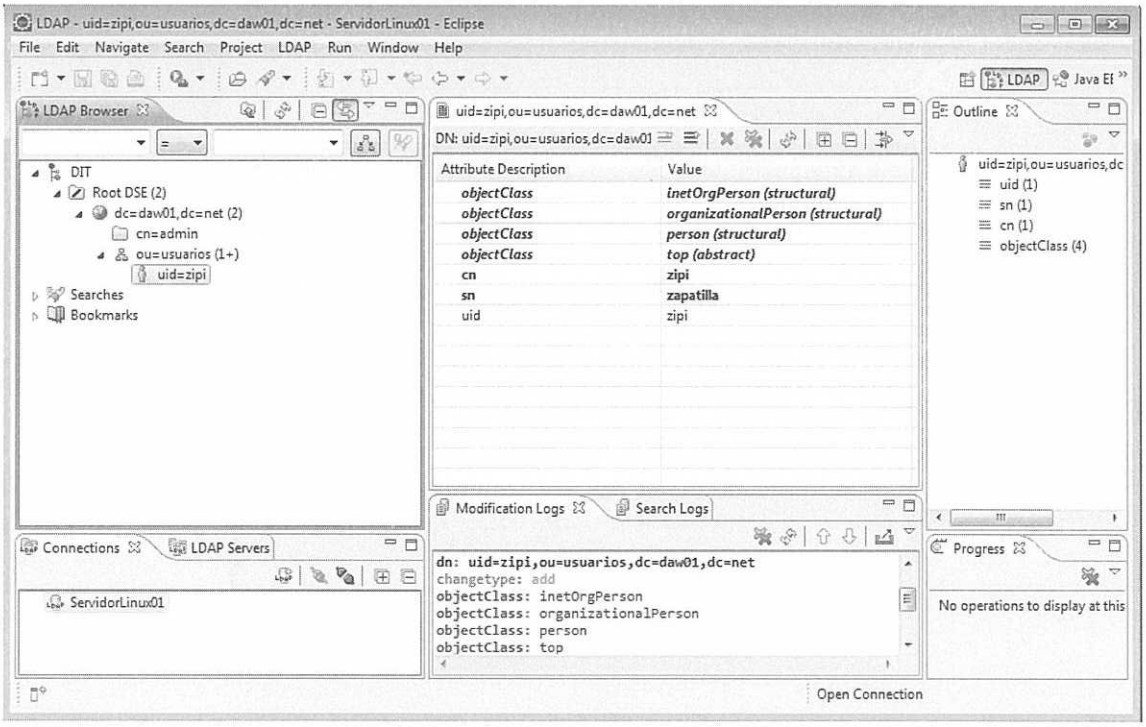


Figura 10.25: *Apache Directory Studio.* Entrada creada

3. ***phpLDAPadmin***

   3.1. Inicia una sesión en **ServidorLinuxXX** con un usuario con privilegios de administración.

   3.2. Instala *phpLDAPadmin* desde los repositorios oficiales de *Ubuntu*.

```
sudo apt-get update
sudo apt-get install phpldapadmin
```

   3.3. Edita el fichero **/etc/phpldapadmin/config.php** y modifica las siguientes directivas, Figuras 10.26 y 10.27.

```
/* Array of base DNs of your LDAP server. Leave this blank to have phpLDAPadmin
   auto-detect it for you. */
$servers->setValue('server','base',array('dc=daw01,dc=net'));
```

Figura 10.26: Fichero **/etc/phpldapadmin/config.php**

```
/* The DN of the user for phpLDAPadmin to bind with. For anonymous binds or
   'cookie','session' or 'sasl' auth_types, LEAVE THE LOGIN_DN AND LOGIN_PASS
   BLANK. If you specify a login_attr in conjunction with a cookie or session
   auth_type, then you can also specify the bind_id/bind_pass here for searching
   the directory for users (ie, if your LDAP server does not allow anonymous
   binds. */
$servers->setValue('login','bind_id','cn=admin,dc=daw01,dc=net');
```

Figura 10.27: Fichero **/etc/phpldapadmin/config.php**

   3.4. Desde **DesarrolloW7XX** accede a `http://servidorlinux01.daw01.net/phpldapadmin` pincha en **conectar** e inicia sesión, Figura 10.28.

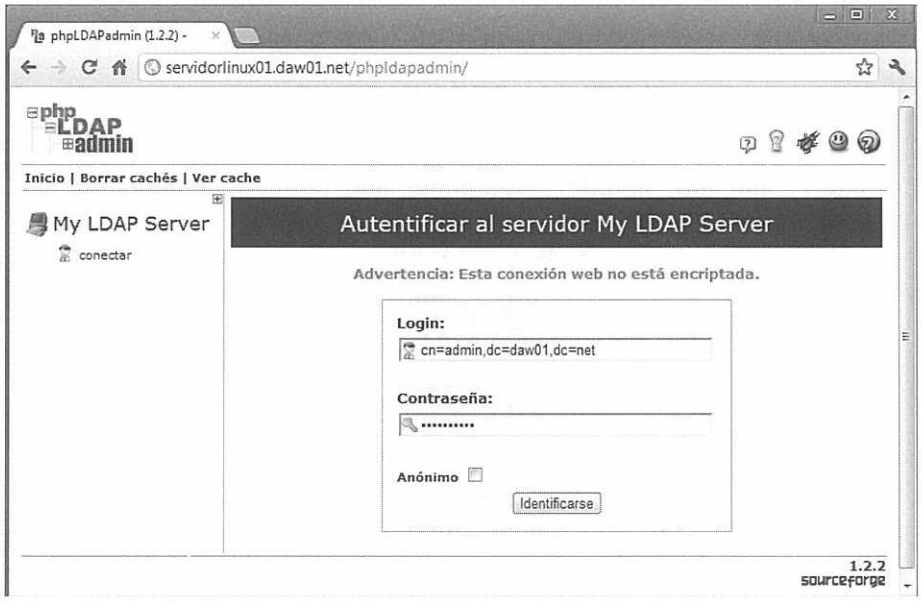


Figura 10.28: *phpLDAPadmin*. Conexión al servidor

3.5. Observa el DIT , Figura 10.29.

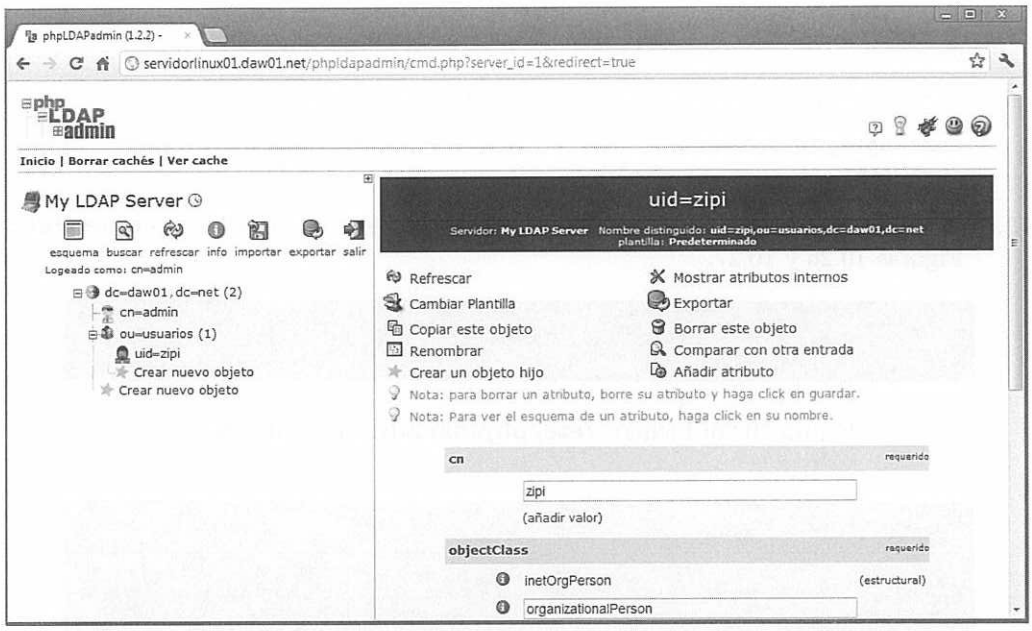


Figura 10.29: *phpLDAPadmin*. DIT

10.4. Configuración del servidor LDAP

Utiliza alguno de los clientes LDAP para importar el fichero **ldif** que proporciona el profesor en el servidor LDAP de **ServidorLinuXX** y crea un DIT como la que se muestra en la Figura 10.30:

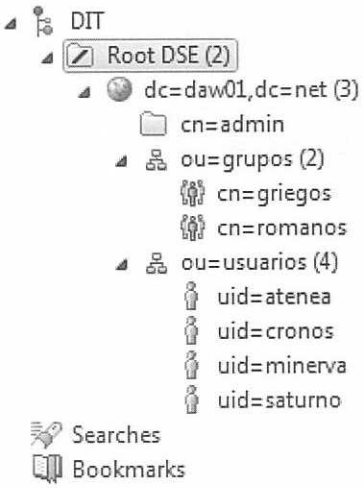


Figura 10.30: DIT

1. Inicia una sesión en **DesarrolloW7XX** con un usuario con privilegios de administración.
2. Obtén el archivo **ldif** que te proporcione el profesor.

```
dn: ou=usuarios,dc=daw01,dc=net
objectClass: organizationalUnit
objectClass: top
ou: usuarios

dn: ou=grupos,dc=daw01,dc=net
objectClass: organizationalUnit
objectClass: top
ou: grupos

dn: uid=minerva,ou=usuarios,dc=daw01,dc=net
objectClass: inetOrgPerson
objectClass: organizationalPerson
objectClass: person
objectClass: top
cn: minerva
sn: minerva
uid: minerva
userPassword:: bWl2ZXJ2YQ==

dn: uid=saturno,ou=usuarios,dc=daw01,dc=net
objectClass: inetOrgPerson
objectClass: organizationalPerson
objectClass: person
objectClass: top
cn: saturno
sn: saturno
uid: saturno
userPassword:

dn: uid=cronos,ou=usuarios,dc=daw01,dc=net
objectClass: inetOrgPerson
objectClass: organizationalPerson
objectClass: person
objectClass: top
cn: cronos
sn: cronos
uid: cronos
userPassword:: Y3Jvbm9z

dn: uid=atenea,ou=usuarios,dc=daw01,dc=net
objectClass: inetOrgPerson
objectClass: organizationalPerson
objectClass: person
objectClass: top
cn: atenea
```

```
sn: atenea
uid: atenea
userPassword:: YXRlbmVh

dn: cn=romanos,ou=grupos,dc=daw01,dc=net
objectClass: groupOfUniqueNames
objectClass: top
cn: romanos
uniqueMember: uid=miverva,ou=usuarios,dc=daw01,dc=net
uniqueMember: uid=saturno,ou=usuarios,dc=daw01,dc=net

dn: cn=griegos,ou=grupos,dc=daw01,dc=net
objectClass: groupOfUniqueNames
objectClass: top
cn: griegos
uniqueMember: uid=atenea,ou=usuarios,dc=daw01,dc=net
uniqueMember: uid=cronos,ou=usuarios,dc=daw01,dc=net
```

3. Edita el fichero y cambia **dawXX** por tu número.
4. Inicia *Eclipse* y abre la perspectiva de *Apache Directory Studio*.
5. Establece una conexión con el servidor ldap.
6. Elimina las entradas creadas en las prácticas anteriores.
7. Importa el fichero ldif.
8. Asegúrate de que se han creado las entradas correctamente.

10.5. Autenticación y autorización LDAP en *Apache*

En la máquina **ServidorLinuxXX** configura en *Apache* la autenticación contra el servidor LDAP para que al directorio **/var/www/profesor** solo puedan acceder los usuarios del grupo **griegos** y al directorio/**var/www/departamento** solo puedan acceder los usuarios del grupo **romanos**.

1. **Habilitar el módulo mod_authnz_ldap**
   1.1. Inicia una sesión en **ServidorLinuxXX** con un usuario con privilegios de administrador.
   1.2. Habilita el módulo ***mod_authnz_ldap***.

   ```
   sudo a2enmod authnz_ldap
   ```

   1.3. Reinicia *Apache*.
2. **Autenticación HTTP *Digest***
   2.1. Edita el fichero de configuración **/etc/apache2/sites-available/000-default.conf** y configura la autenticación/autorización LDAP sobre el directorio **/var/www/profesor**, Figura 10.31.

```
<Directory /var/www/html/profesor>
        Options Indexes FollowSymLinks
        AllowOverride None
        AuthType Basic
        AuthBasicProvider ldap
        AuthName "Introduce tu usuario y password"
        AuthLDAPURL "ldap://localhost/dc=daw01,dc=net?uid?sub?(objectClass=*)"
        AuthLDAPBindDN "cn=admin,dc=daw01,dc=net"
        AuthLDAPBindPassword despliegue
        <RequireALL>
                Require ldap-group cn=griegos,ou=grupos,dc=daw01,dc=net
                <RequireAny>
                        Require ip 127.0.0.1
                        Require ip 192.168.1.16
                </RequireAny>
        </RequireALL>
</Directory>
```

Figura 10.31: Autenticación/autorización LDAP

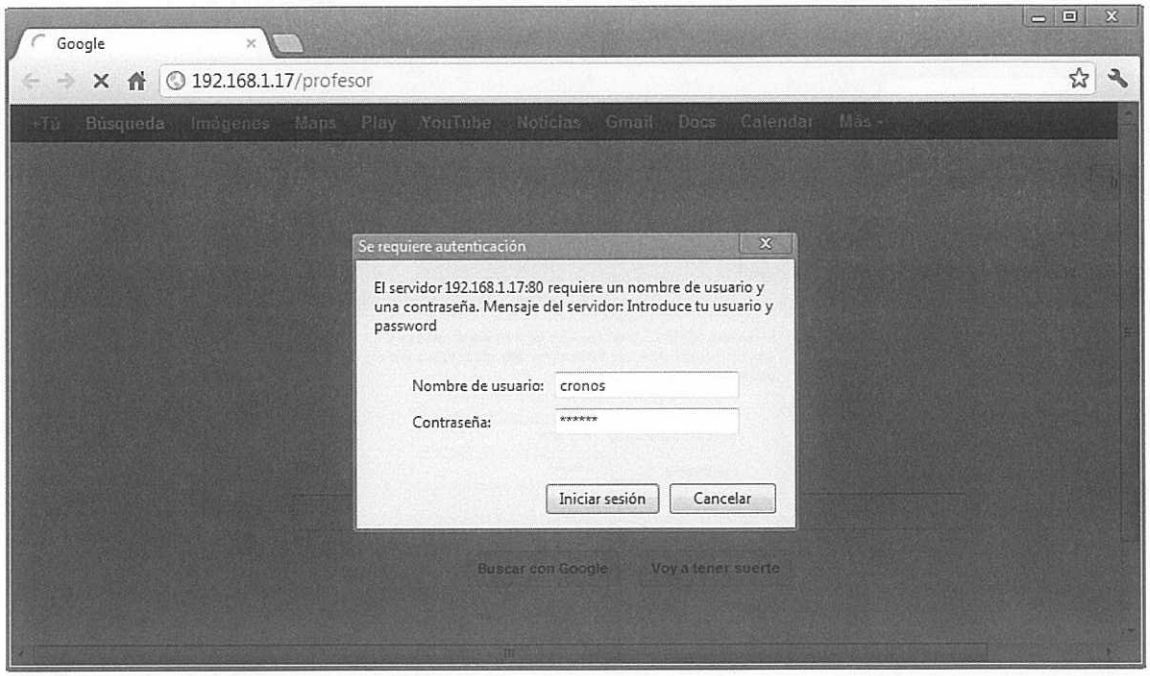


Figura 10.32: Conexión desde **DesarrrolloW7XX**

2.2. Reinicia *Apache*.

2.3. Desde **DesarrolloW7XX** accede a `http://192.168.1.X7/profesor/` con el usuario **cronos**, Figura 10.32. Intenta el acceso con **minerva**. ¿Es posible?.

2.4. Edita el fichero de configuración **/etc/apache2/sites-available/000-default.conf** y configura la autenticación/autorización LDAP sobre el directorio **/var/www/departamento**, Figura 10.33.

```
<Directory /var/www/html/departamento>
        Options Indexes FollowSymLinks
        AllowOverride None
        AuthType Basic
        AuthBasicProvider ldap
        AuthName "Introduce tu usuario y password"
        AuthLDAPURL "ldap://localhost/dc=daw01,dc=net?uid?sub?(objectClass=*)"
        AuthLDAPBindDN "cn=admin,dc=daw01,dc=net"
        AuthLDAPBindPassword despliegue
        Require ldap-group cn=romanos,ou=grupos,dc=daw01,dc=net
</Directory>
```

Figura 10.33: Autenticación/autorización LDAP

2.5. Reinicia *Apache*.

2.6. Desde **DesarrolloW7XX** accede a `http://192.168.1.X7/departamento/` con el usuario **miverva**, Figura 10.34. Intenta el acceso con **cronos** ¿Es posible?.

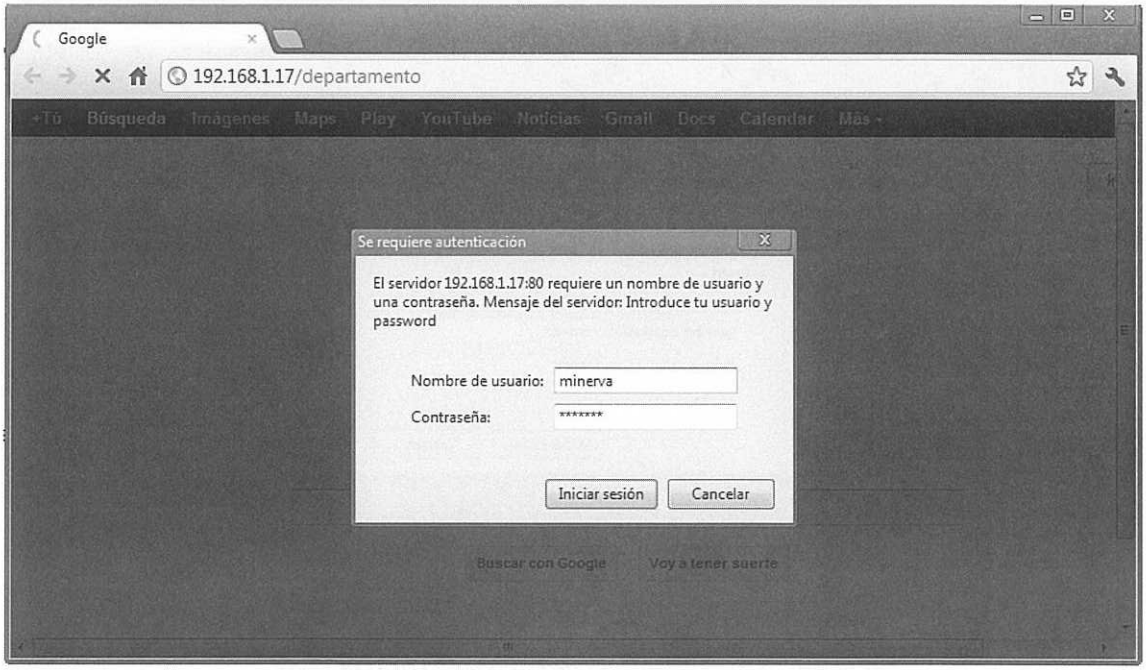


Figura 10.34: Conexión desde **DesarrrolloW7XX**

10.6. Autenticación y autorización LDAP en *Tomcat* (*JNDIRealm*)

Configura el acceso a la aplicación **curso** (en el *Host* **europa.dawXX.net**) utilizando un *JNDIRealm* para que solo puedan acceder los usuarios del grupo **griegos** y configura el acceso a la aplicación **compras** (en el *Host* **africa.dawXX.net**) utilizando un *JNDIRealm* para que solo puedan acceder los usuarios del grupo **romanos**.

1. **Eliminar las aplicaciones del servidor.**
   1.1. Inicia sesión en **DeasarrolloWindowsXX**.
   1.2. Elimina la aplicación **curso** del servidor *Tomcat* (europa).
   1.3. Elimina la aplicación **compras** del servidor *Tomcat* (africa).
2. **Configurar la aplicación curso.**
   2.1. Edita el fichero **context.xml** para la aplicación **curso** (**META-INF/context.xml**) y configura el *Realm*, Figura 10.35.

```
<Context>
    <Realm  className="org.apache.catalina.realm.JNDIRealm"
            connectionURL="ldap://localhost:389"
            userPattern="uid={0},ou=usuarios,dc=daw01,dc=net"
            roleBase="ou=grupos,dc=daw01,dc=net"
            roleName="cn"
            roleSearch="(uniqueMember={0})"
    />
</Context>
```

Figura 10.35: Fichero **META-INF/context.xml** de la aplicación **curso**

   2.2. Edita el descriptor de despliegue (**web.xml**) de la aplicación **curso y** configura la autenticación, Figura 10.36.

```
<security-constraint>
    <web-resource-collection>
        <web-resource-name>JNDIRealm</web-resource-name>
        <url-pattern>/*</url-pattern>
    </web-resource-collection>
    <auth-constraint>
        <role-name>griegos</role-name>
    </auth-constraint>
    <user-data-constraint>
        <transport-guarantee>CONFIDENTIAL</transport-guarantee>
    </user-data-constraint>
</security-constraint>
```

Figura 10.36: Fichero **web.xml** de la aplicación **curso**

   2.3. Elimina la aplicación **curso** del servidor *Tomcat* (en el *Host* **europa.dawXX.net**).
   2.4. Genera el fichero **curso.war** con los cambios.
   2.5. Despliega el nuevo fichero **curso.war**.

Figura 10.37: Acceso a la aplicación **curso**

2.6. Accede a la aplicación y observa que se pide usuario y contraseña y verifica que solo puedes acceder con los usuarios el grupo **griegos**, Figura 10.37

3. <u>**Configurar la aplicación compras**</u>.

3.1. Edita el fichero **context.xml** para la aplicación **compras** (**META-INF/context.xml**) y configura el *Realm*, Figura 10.38.

```
<Context>
    <Realm  className="org.apache.catalina.realm.JNDIRealm"
            connectionURL="ldap://localhost:389"
            userPattern="uid={0},ou=usuarios,dc=daw01,dc=net"
            roleBase="ou=grupos,dc=daw01,dc=net"
            roleName="cn"
            roleSearch="(uniqueMember={0})"
    />
</Context>
```

Figura 10.38: Fichero **META-INF/context.xml** de la aplicación **compras**

3.2. Edita el descriptor de despliegue (**web.xml**) de la aplicación **compras** y configura la autenticación, Figura 10.39.

3.3. Elimina la aplicación **compras** del servidor *Tomcat* (en el *Host* **africa.dawXX.net**).

3.4. Genera el fichero **compras.war** con los cambios.

3.5. Despliega el nuevo fichero **compras.war**.

```
<security-constraint>
    <web-resource-collection>
        <web-resource-name>JNDIRealm</web-resource-name>
        <url-pattern>/*</url-pattern>
    </web-resource-collection>
    <auth-constraint>
        <role-name>romanos</role-name>
    </auth-constraint>
</security-constraint>
```

Figura 10.39: Fichero **web.xml** de la aplicación **compras**

Figura 10.40: Acceso a la aplicación **compras**

3.6. Accede a la aplicación y observa que se pide usuario y contraseña y verifica que solo puedes acceder con los usuarios el grupo **romanos**, Figura 10.40

Capítulo 11

Administración de GlassFish

11.1. Instalación de *GlassFish 4.0* en *Windows*

Instala (configuración típica) el servidor de aplicaciones *GlassFish Server Open Source Edition*, en su versión 4.0, en la máquina **ServidorW2008XX**.

Algunos conceptos sobre *GlassFish*:

- **Nodo**: Equipo donde se ejecuta una instancia de *GlassFish*.
- **Instancia**: Una instancia en un servidor *GlassFish* se corresponde con una máquina virtual de java (JVM, *Java Virtual Machine*) en un único nodo. En una instancia se pueden crear varios servidores virtuales.
- **Dominio**: Un dominio consiste en una o varias instancias que se administran juntas. Una instancia puede pertenecer solo a un dominio. Cada dominio tiene sus propios ficheros de configuración, ficheros de *logs* y áreas de despliegue de aplicaciones independientes de otros dominios. En una instalación básica se crea un dominio por defecto (**domain1**) con una instancia por defecto que tiene configurado un servidor virtual por defecto (**server**) que escucha peticiones HTTP en el puerto 8080/TCP y peticiones HTTPS en el puerto 8181/TCP.
- **DAS (*Domain Administration Server*)**: Instancia especial que contiene aplicaciones de administración. Autentica al administrador, acepta peticiones de herramientas administrativas y se comunica con otras instancias del dominio para trasladarles las peticiones de las herramientas administrativas. Cada dominio tiene su propio DAS. Durante la instalación se crea un DAS con un servidor virtual por defecto (**__asadmin**) que escucha peticiones HTTP en el puerto defecto 4848/TCP.

1. **Instalación de *Java Standard Edition Development Kit (JDK)*.**

   GlassFish requiere para su funcionamiento tener instalado el *Java Standard Edition Development Kit (JDK)*.

   1.1. Inicia sesión en **ServidorWindowsXX** con usuario con privilegios de administrador.

   1.2. Descarga *Java SE Development Kit* de `http://www.oracle.com/technetwork/java/javase/downloads/index.html`. En el momento de realizar este manual estaba disponible la versión 7.

   1.3. Instala la versión de JDK descargada previamente.

2. **Instalación de *GlassFish* utilizando el instalador de *Windows*.**

   A la hora de instalar *GlassFish 4.0* existen 2 tipos de instaladores: **Zip** e **Instaladores Nativos.** El instalador Zip puede ejecutarse sobre cualquier plataforma mientras que el los instaladores nativos son específicos de cada plataforma. El instalador Zip es más simple y continente una configuración lista para usar, mientras que los instaladores nativos ofrecen una interfaz gráfica que permite personalizar la instalación de*GlassFish*. En esta práctica emplearemos el instalador nativo para *Windows*.

   2.1. Si has instalado **Tomcat** en **ServidorWindowsXX** detén Tomcat para que no entre en conflicto con *GlassFish*.

   2.2. Accede a `https://glassfish.java.net/download.html`. Pincha en **Earlier Releases**. Accede a **GlassFish Archives**.

   2.3. Observa que existen dos distribuciones: *Full Platform* y *Web Profile*. Como no hemos desarrollado aplicaciones con EJBs vamos a descargar la distribución *Web Profile*.

2.4. Descarga el instalador nativo para Windows **glassfish-4.0-web-windows.exe**, Figura 11.1.

Archived Releases

GlassFish Server Open Source Edition 4.0 (Java EE 7)

Platform	Java EE 7	Locale	
Windows	Web Profile	EN	glassfish-4.0-web-windows.exe
	Full Platform	EN	glassfish-4.0-windows.exe
Windows	Web Profile	ML	glassfish-4.0-web-windows-ml.exe
	Full Platform	ML	glassfish-4.0-windows-ml.exe
Linux, Mac, Solaris	Web Profile	EN	glassfish-4.0-web-unix.sh
	Full Platform	EN	glassfish-4.0-unix.sh
Linux, Mac, Solaris	Web Profile	ML	glassfish-4.0-web-unix-ml.sh
	Full Platform	ML	glassfish-4.0-unix-ml.sh
Zip (Win, Linux, Mac, Solaris)	Web Profile	EN	glassfish-4.0-web.zip
	Full Platform	EN	glassfish-4.0.zip
Zip (Win, Linux, Mac, Solaris)	Web Profile	ML	glassfish-4.0-web-ml.zip
	Full Platform	ML	glassfish-4.0-ml.zip

Figura 11.1: Descarga de *GlassFish*

2.5. Ejecuta el instalador y sigue los pasos del asistente.

a. Lee la introducción del asistente y pincha en **Siguiente**.

b. Selecciona **Instalación Típica** y pincha en **Siguiente**, Figura 11.2.

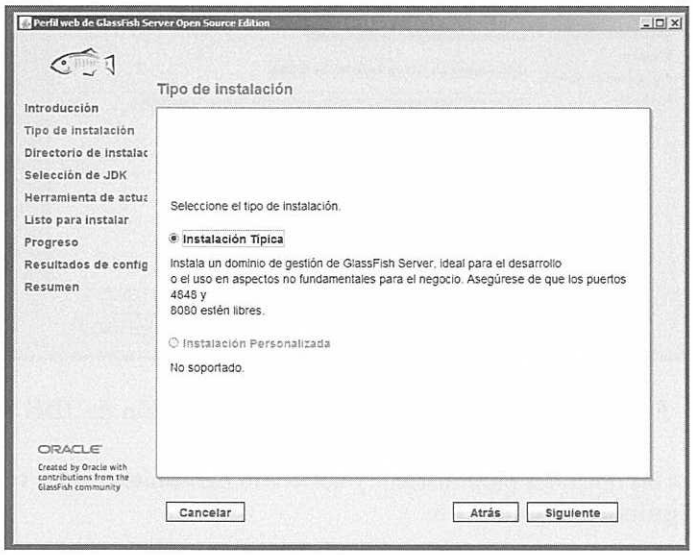


Figura 11.2: Asistente de instalación. Tipo de instalación

c. Deja el directorio de instalación por defecto y pincha en **Siguiente**, Figura 11.3.

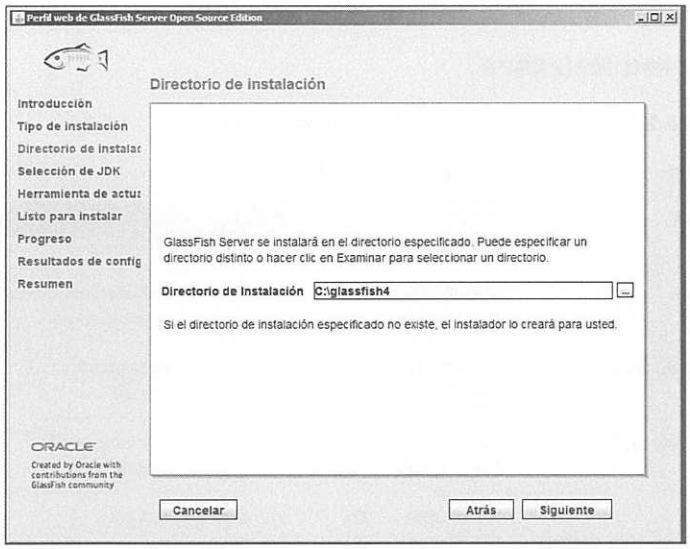


Figura 11.3: Asistente de instalación. Directorio de instalación

d. Deja la selección de JDK que se ofrece por defecto, Figura 11.4.

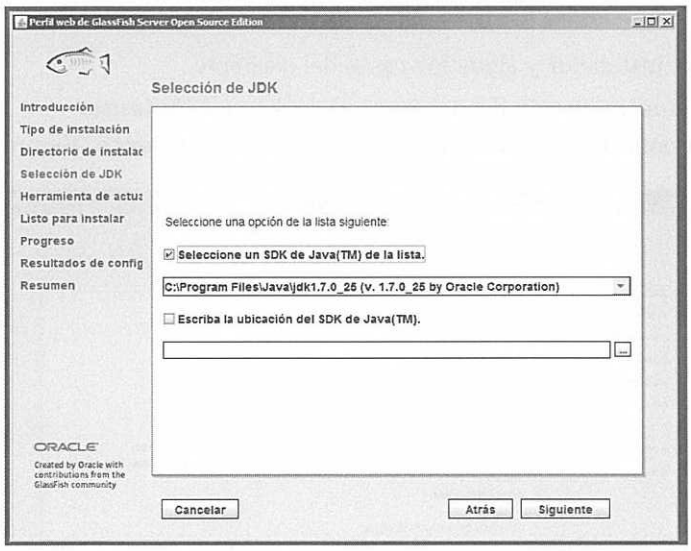


Figura 11.4: Asistente de instalación. Selección de JDK

e. Marca las opciones para instalar y activar la herramienta de actualizaciones y pincha en **Siguiente**, Figura 11.5.

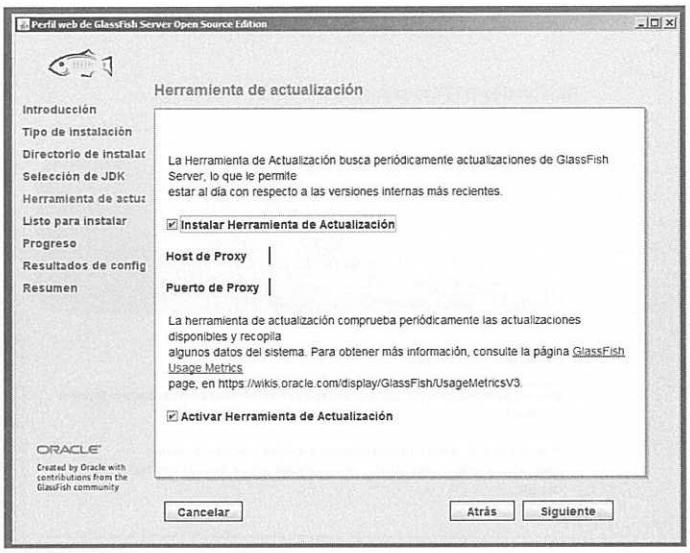


Figura 11.5: Asistente de instalación. Herramienta de actualizaciones

f. Pincha en **Instalar** para comenzar la instalación, Figura 11.6.

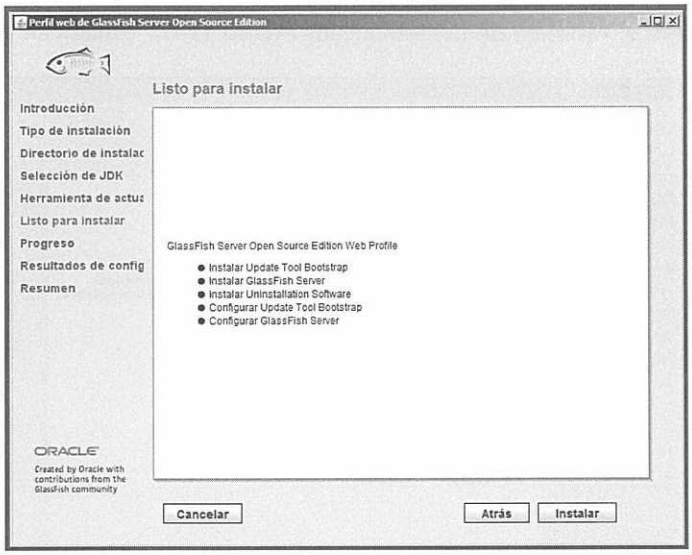


Figura 11.6: Asistente de instalación. Instalación

g. Observa que se ha configurado el dominio (**dominio1**) con el usuario **admin** (sin contraseña). El puerto de administración es 4848/TCP y el servidor los servidores HTTP y HTTPS usan los puertos 8080/TCP y 8181/TCP. El instalador también crea un servicio en Windows para el dominio, Figura 11.7. Pincha en **Siguiente** para continuar.

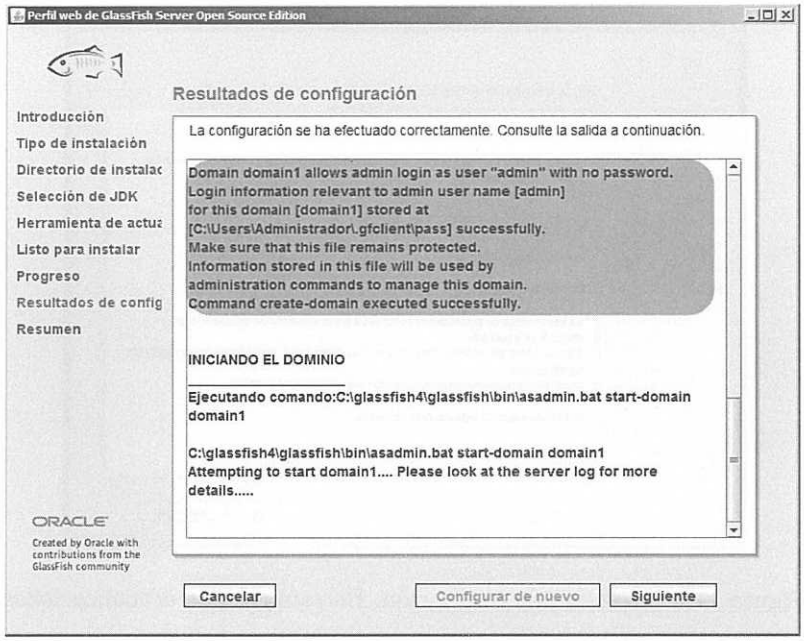


Figura 11.7: Asistente de instalación. Configuración de un dominio

h. Observa el resumen de la instalación, Figura 11.8 y pincha en **Salir**.

Figura 11.8: Asistente de instalación. Resumen de la instalación

2.6. Observa que se ha creado una entrada en el menú **Inicio, Todos los programas**, Figura 11.9.

Figura 11.9: Accesos a opciones sobre el servidor

2.7. Inicia el servidor pinchando en ***Iniciar Application Server***.

2.8. Comprueba que el servidor está escuchando en los puertos 8080/TCP (HTTP), 8181/TCP(HTTPS) y 4848/TCP (administración).

```
netstat -a -p TCP -n
```

2.9. Consulta el contenido del directorio **C:\glassfish4\domains\domain1\docroot**.

2.10. Consulta el contenido del fichero **C:\glassfish4\domains\domain1\docroot\index.html**.

2.11. Habilita una regla en el *Firewall* para que se permitan conexiones a *Glassfish*.

a. Accede a **Inicio, Panel de control, Seguridad, *Firewall* de *Windows***.

b. Pincha en **Cambiar la configuración**.

c. Pincha en **Excepciones**.

d. Pincha en **Agregar un puerto** y añade el puerto 8080/TCP, Figura 11.10.

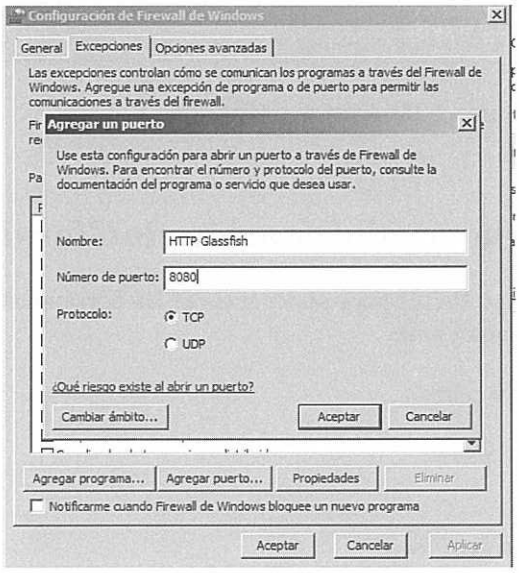


Figura 11.10: Abrir el puerto 8080/TCP en el *Firewall*

e. Agrega el puerto 8181/TCP.
f. Agrega el puerto 4848/TCP.

3. **Prueba de conexión al servidor**.

   3.1. Accede a `http://localhost:8080` para comprobar el correcto funcionamiento de *Glassfish*, Figura 11.11

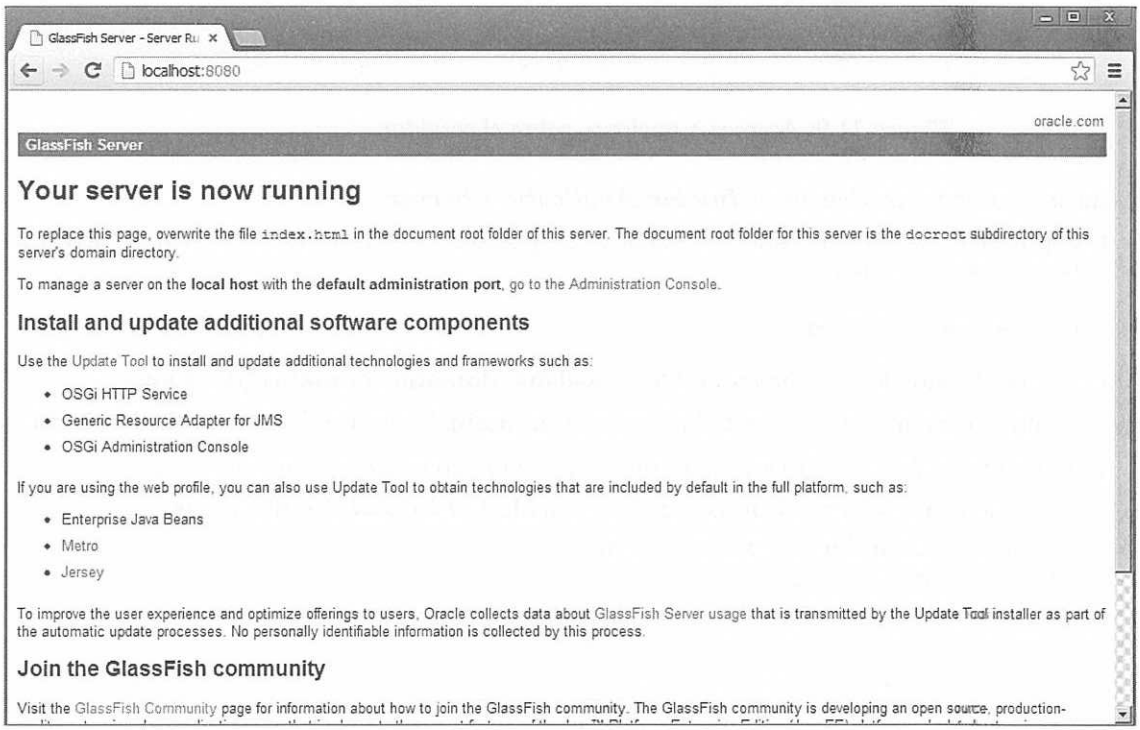


Figura 11.11: Acceso a *GlassFish*

11.2. Herramientas de administración de *GlassFish*

- Configura la variable de entorno PATH para poder invocar las herramientas de *GlassFish* en línea de comandos desde cualquier ruta.
- Utiliza la herramienta **asdmin** para:
  - Listar los dominios disponibles.
  - Iniciar y parar el dominio.
  - Consultar la versión de *GlassFish*.
  - Consultar los contenedores disponibles en *GlassFish*
- Accede a la consola de administración web de *GlassFish* desde **ServidorW2008XX**.

1. **Configurar la variable de entorno PATH .**

   Si queremos utilizar las aplicaciones de *GlassFish* desde la cualquier localización en la línea de comandos podemos añadir la ruta de los binarios de *GlassFish* en la variable de entorno **PATH**.

   1.1. Inicia sesión en **ServidorW2008XX** con un usuario con privilelgios de administrador.

   1.2. Accede a **Inicio, Panel de control, Sistema y mantenimiento, Sistema, Configuración avanzada del sistema, Variables de Entorno**.

   1.3. En **Variables de sistema** selecciona la variable **Path** y pincha en **Editar**.

   1.4. Añade la ruta **C:\glassfish4\glassfish4\bin**.

   1.5. Acepta los cambios.

2. **Herramienta asadmin.**

   Utiliza **asadmin** en línea de comandos para realizar las acciones indicadas.

   2.1. Inicia un terminal.

   2.2. Lista los dominios disponibles.

   ```
   asadmin list-domains
   ```

   2.3. Detén el dominio por defecto de *GlassFish*.

   ```
   asadmin stop-domain
   ```

   2.4. Inicia el dominio de *GlassFish*.

   ```
   asadmin start-domain
   ```

   2.5. Consulta la versión de *GlassFish*.

   ```
   asadmin
   asadmin>version
   ```

   2.6. Consulta los contenedores disponibles en *GlassFish*, Figura 11.12.

   ```
   asadmin
   asadmin>list-containers
   ```

3. **Acceso a la consola web de administración.**

   3.1. Accede a la consola de administración web de *GlassFish* en `http://localhost:4848`, Figuras 11.13.

   3.2. Accede ***Change Administrator Password*** y asígnale al usuario **admin** la contraseña despliegue, Figuras 11.14.

```
C:\Users\Administrador>asadmin
Use "exit" to exit and "help" for online help.
asadmin> version
Version = GlassFish Server Open Source Edition  4.0  (build 89)
Command version executed successfully.
asadmin> list-containers
List all known application containers
jpa
jpa
web
weld
weld
security
grizzly
ejb
osgi
resources_ear
resources
connector
Command list-containers executed successfully.
asadmin>
```

Figura 11.12: Contenedores disponibles

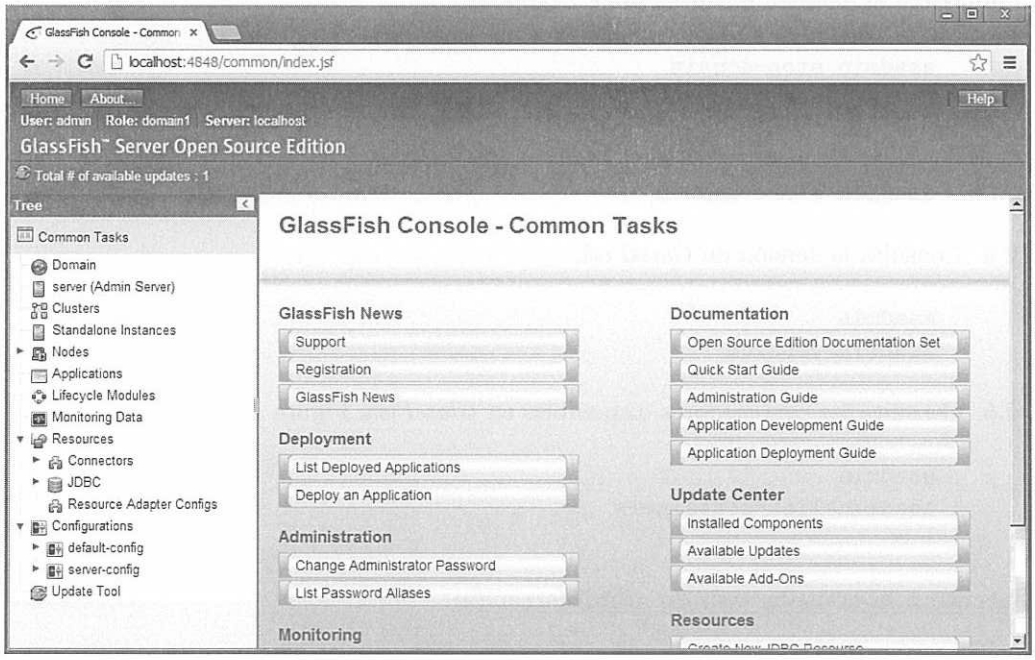


Figura 11.13: Acceso a la consola de administración web de *GlassFish*

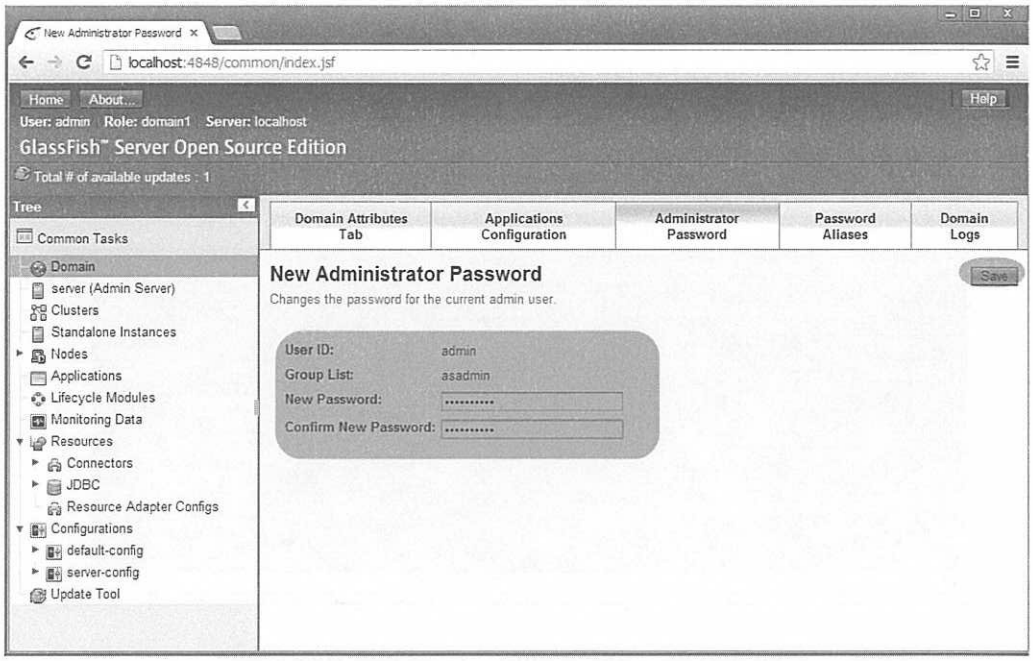


Figura 11.14: Acceso a la consola de administración web de *GlassFish*

11.3. Despliegue de aplicaciones web en *GlassFish*

- Utiliza la herramienta **asdmin** para:
  - Listar las aplicaciones web disponibles.
  - Desplegar la aplicación **Calendar**.
  - Eliminar la aplicación **Calendar**.
- Accede a la consola de administración web de *GlassFish* desde **ServidorW2008XX**.
  - Lista las aplicaciones web disponibles.
  - Despliega la aplicación **Calendar**.
- Habilita la administración segura para poder acceder al DAS de forma remota. Prueba el acceso a la consola de administración web desde **DesarrolloW7XX**.

1. **Despliegue de aplicaciones utilizando asadmin.**

   1.1. Inicia sesión en **ServidorW2008XX** con un usuario con privilelgios de administrador.

   1.2. Obtén el fichero **Calendar.war** y guárdalo en el directorio home del usuario **Administrador**.

   1.3. Inicia un terminal.

   1.4. Consulta las aplicaciones desplegadas en *GlassFish*.

```
asadmin list-applications
```

1.5. Despliega la aplicación **Calendar.war**

```
asadmin deploy Calendar.war
```

1.6. Consulta las aplicaciones desplegadas en *GlassFish*, Figura 11.15.

```
asadmin list-applicactions
```

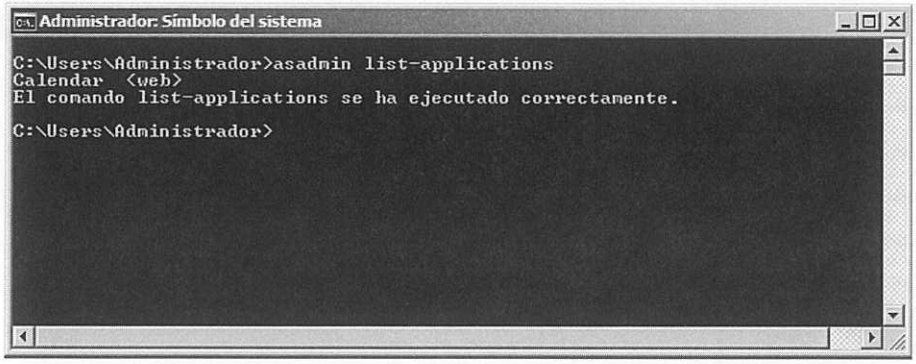


Figura 11.15: Aplicaciones desplegadas *GlassFish*

1.7. Abre un navegador y accede a `https://localhost/Calendar` para probar la aplicación desplegada, Figura 11.16.

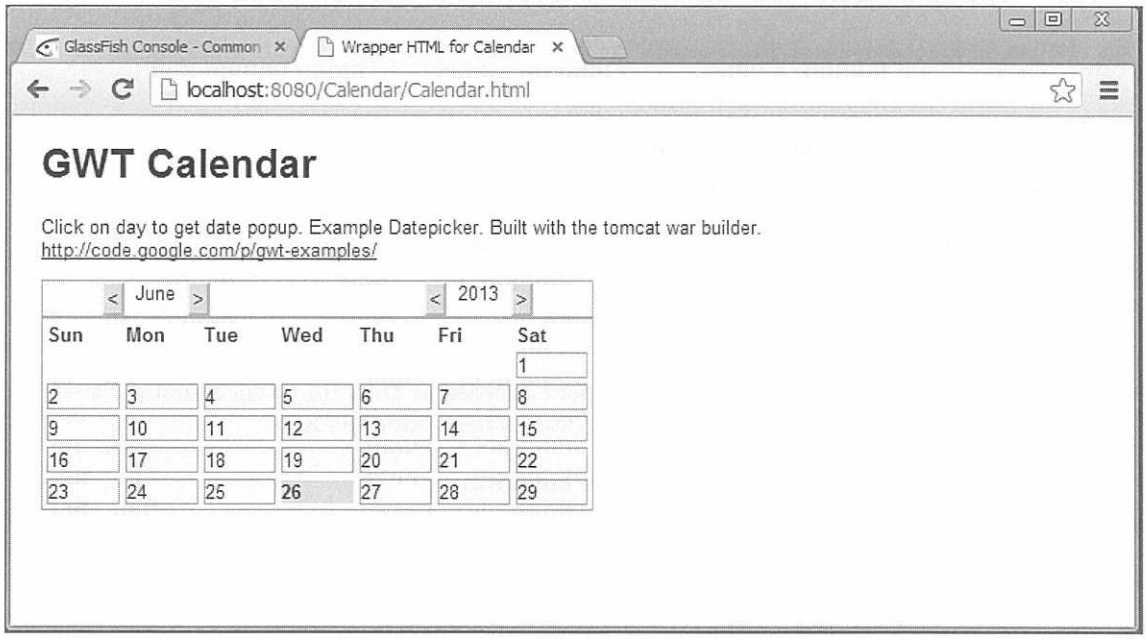


Figura 11.16: Aplicación Calendar

1.8. Consulta el contenido del directorio **C:\glassfish3\domains\domain1\applications**, Figura 11.17.

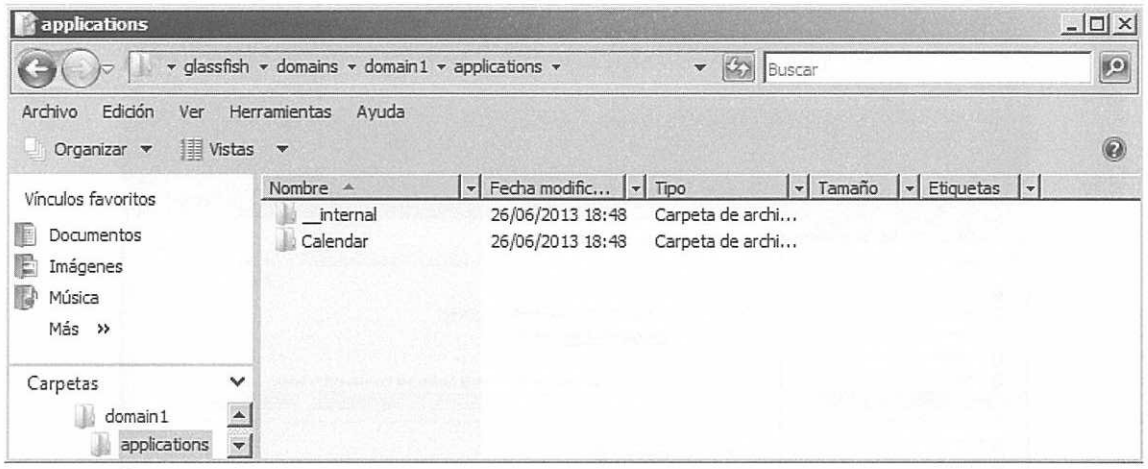


Figura 11.17: Directorio **C:\glassfish3\domains\domain1\applications**

1.9. Elimina la aplicación **Calendar.war** de *GlassFish*.

```
asadmin undeploy Calendar
```

2. **Despliegue de aplicaciones utilizando la consola de administración web.**

2.1. Accede a la consola de administración web de *GlassFish* en `http://localhost:4848`.

2.2. Accede Applications, Deploy, Seleccionar Archivo y elige el fichero Calendar.war, Figura 11.18.

2.3. Pincha en **OK** para desplegar la aplicación, Figura 11.19.

2.4. Pincha sobre **Calendar** para acceder a sus propiedades, Figura 11.20.

3. **Habilita la administración segura.**

3.1. Sobre **DesarrolloW7XX** establece una conexión a la consola de administración web de *GlassFish* y observa que no puedes iniciar sesión, Figura 11.21

3.2. En **ServidorW2008XX** inicia un terminal y habilita la administración remota segura.

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

3.3. Reinicia el dominio

```
asadmin stop-domain
asadmin start-domain
```

3.4. Sobre **DesarrolloW7XX** establece una conexión a la consola de administración web de *GlassFish* y observa que ahora sí puedes iniciar sesión, Figura 11.22

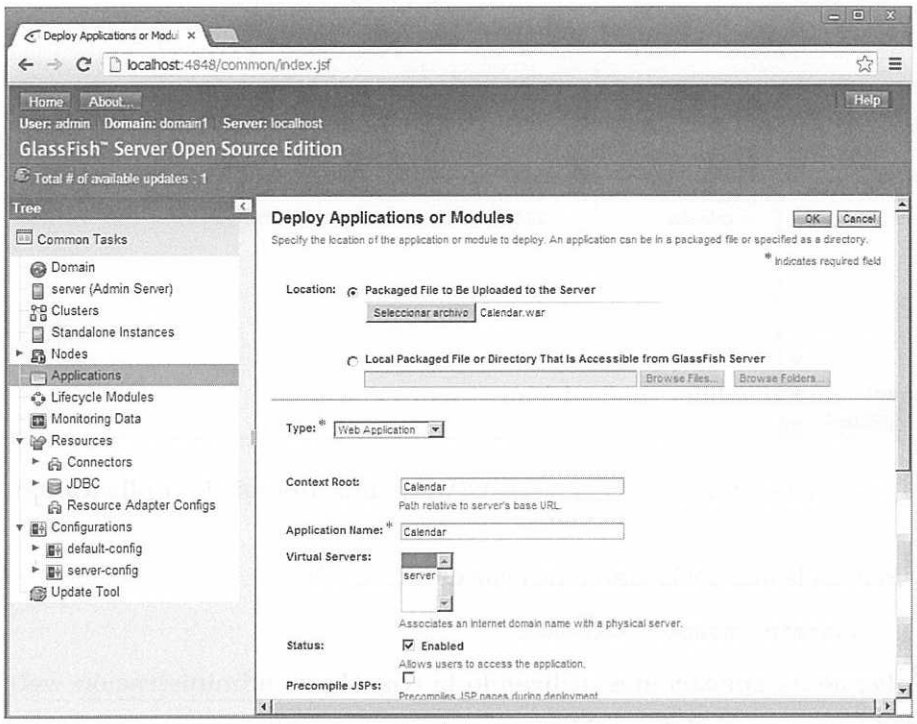


Figura 11.18: Desplegar la aplicación **Calendar**

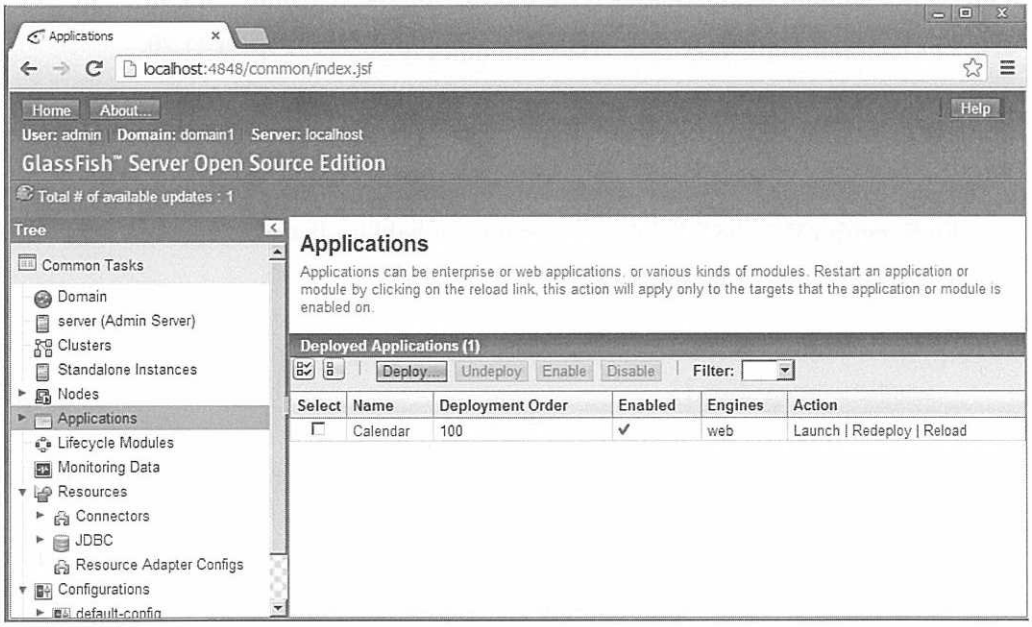


Figura 11.19: Aplicación **Calendar** desplegada

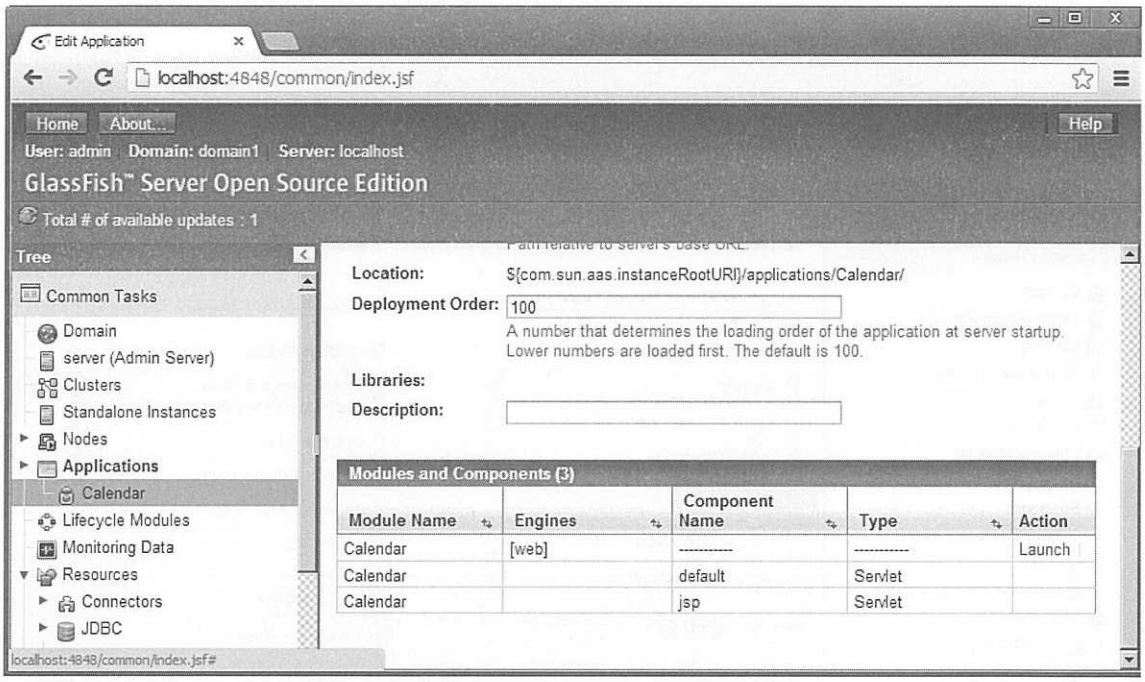


Figura 11.20: Propiedades de **Calendar**

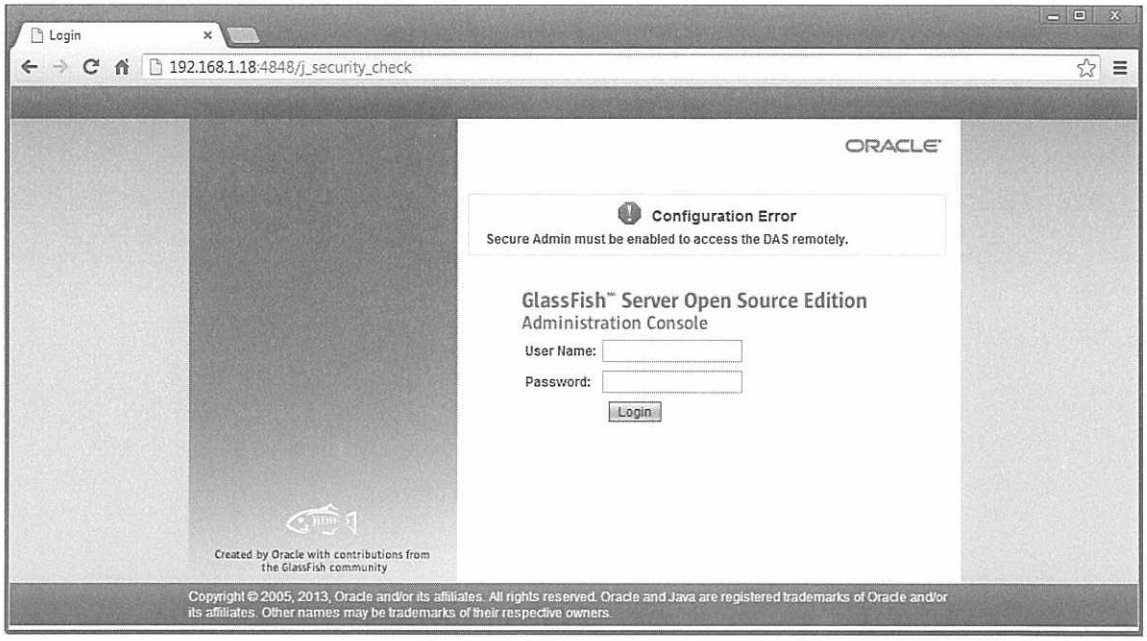


Figura 11.21: Acceso a la consola de administración web desde **DesarrolloW7XX**

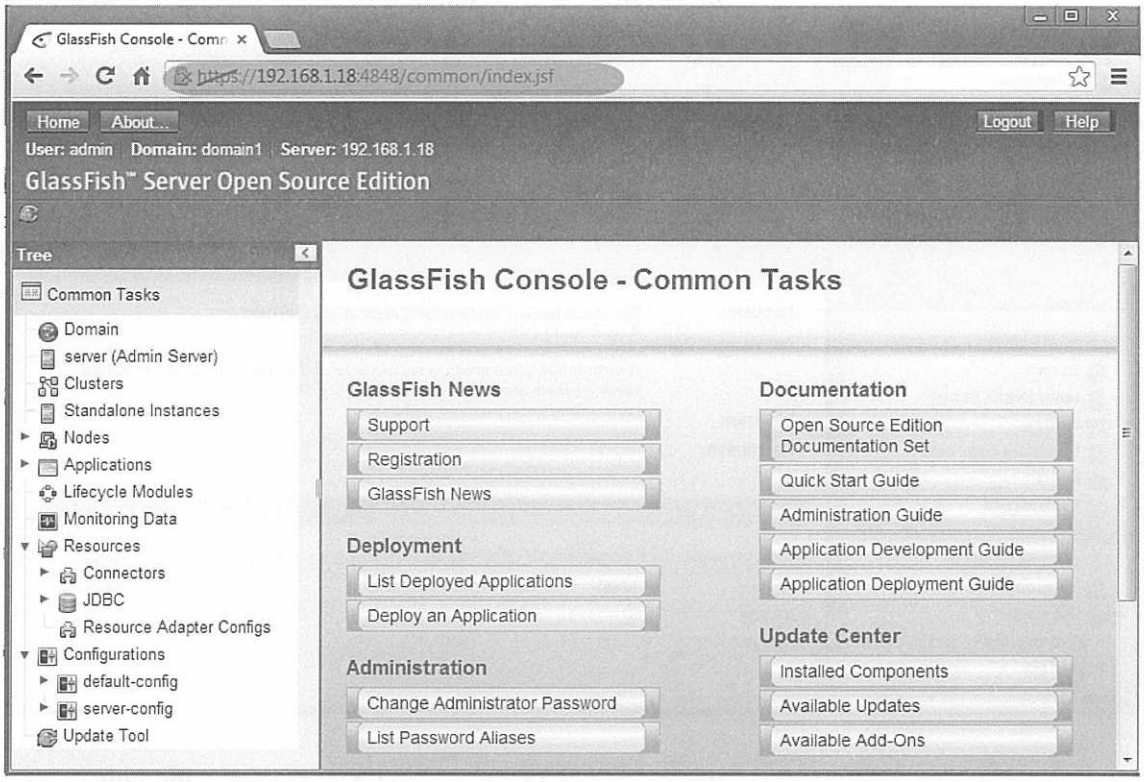


Figura 11.22: Acceso a la consola de administración web desde **DesarrolloW7XX**

3.5. Despliega otras aplicaciones desarrolladas durante el curso.