



# Despliegue de aplicaciones web

Javier Muñoz Carmona

```
mirror_mod.use_z = False
elif_operation = "MIRROR_Z"
mirror_mod.use_x = False
mirror_mod.use_y = False
mirror_mod.use_z = True

#selection at the end -add back the
mirror_op.select= 1
modifier_op.select=1
obj.context.scene.objects.active = modifier_op
print("Selected" + str(modifier_op)) # mirror op
```

Consulte nuestra página web: **[www.sintesis.com](http://www.sintesis.com)**  
En ella encontrará el catálogo completo y comentado

# Despliegue de aplicaciones web

Javier Muñoz Carmona



ASESOR EDITORIAL:

---

Juan Carlos Moreno Pérez

© Javier Muñoz Carmona

© EDITORIAL SÍNTESIS, S. A.  
Vallehermoso, 34. 28015 Madrid  
Teléfono 91 593 20 98  
<http://www.sintesis.com>

ISBN: 978-84-1357-070-9  
Depósito Legal: M-2.884-2021

Impreso en España - Printed in Spain

Reservados todos los derechos. Está prohibido, bajo las sanciones penales y el resarcimiento civil previstos en las leyes, reproducir, registrar o transmitir esta publicación, íntegra o parcialmente, por cualquier sistema de recuperación y por cualquier medio, sea mecánico, electrónico, magnético, electroóptico, por fotocopia o por cualquier otro, sin la autorización previa por escrito de Editorial Síntesis, S. A.

# Índice

<b>PRESENTACIÓN .....</b>	9
<b>1. SERVICIOS DE RED IMPLICADOS EN EL DESPLIEGUE DE UNA APLICACIÓN .....</b>	11
Objetivos .....	11
Mapa conceptual .....	12
Glosario .....	12
1.1. Introducción .....	13
1.2. Sistema de nombre de dominio .....	13
1.2.1. Resolución .....	14
1.2.2. Nombre de dominio .....	14
1.2.3. Objetivos .....	14
1.2.4. Niveles de dominio .....	15
1.3. Zonas de búsquedas, tipos de servidores DNS y registros .....	15
1.3.1. Tipos de servidores DNS .....	17
1.3.2. Registros DNS .....	17
1.4. Funcionamiento del servicio DNS y tipos de consultas .....	18
1.4.1. Consulta recursiva .....	20
1.4.2. Consulta iterativa .....	21
1.4.3. Consulta inversa .....	21
1.5. Instalación y configuración de un servidor DNS en SO Linux .....	21
1.6. Servicio de directorio: características y funcionalidad .....	29
1.7. Organización de LDAP .....	29
1.8. Archivos básicos de configuración y uso .....	31
1.9. Instalación de OpenLDAP en SO Linux .....	33
1.10. Adaptación de la configuración del servidor de directorios para el despliegue de la aplicación. Usuarios centralizados .....	37
1.10.1. Autenticación en el servicio de directorio .....	37
1.10.2. Usuarios centralizados .....	39

Resumen .....	43
Supuestos prácticos .....	43
Ejercicios propuestos .....	44
Actividades de autoevaluación .....	44
<b>2. INSTALACIÓN Y ADMINISTRACIÓN DE SERVIDORES DE TRANSFERENCIA DE ARCHIVOS .....</b>	<b>47</b>
Objetivos .....	47
Mapa conceptual .....	48
Glosario .....	48
2.1. Introducción .....	49
2.2. Servicio de transferencia de archivos. Permisos y cuotas .....	49
2.2.1. Permisos .....	50
2.2.2. Cuotas .....	53
2.3. Tipos de usuarios, accesos al servicio y transferencia de ficheros .....	57
2.3.1. Tipos de usuarios .....	57
2.3.2. Tipos de accesos al servicio .....	57
2.3.3. Tipos de transferencia de ficheros .....	58
2.4. Modos de conexión al cliente .....	59
2.4.1. Modo activo .....	59
2.4.2. Modo pasivo .....	59
2.5. Protocolo seguro de transferencia de archivos .....	59
2.6. Utilización de herramientas gráficas y en modo texto. Comandos .....	61
2.6.1. Herramientas .....	61
2.6.2. Comandos .....	63
2.7. Instalación y configuración del servidor proFTPD en SO Linux .....	64
2.7.1. Validación mediante un host virtual .....	67
2.7.2. Validación del servicio FTP mediante LDAP .....	68
2.8. Utilización del servicio de transferencia de archivos .....	71
2.8.1. Desde el navegador .....	71
2.8.2. En el proceso de despliegue de la aplicación web .....	72
Resumen .....	73
Supuestos prácticos .....	73
Ejercicios propuestos .....	74
Actividades de autoevaluación .....	74
<b>3. IMPLANTACIÓN DE ARQUITECTURAS WEB .....</b>	<b>77</b>
Objetivos .....	77
Mapa conceptual .....	78
Glosario .....	78
3.1. Introducción .....	79
3.2. Arquitecturas web .....	79
3.3. Evolución de la tecnología web .....	80
3.4. Tecnologías usadas en aplicaciones web .....	81
3.4.1. En el lado servidor .....	81
3.4.2. En el lado cliente .....	82
3.4.3. En ambos .....	82
3.5. Servidores y aplicaciones libres y propietarias .....	83
3.6. Protocolo HTTP vs HTTPS .....	85

3.7. Clasificación de servidores de aplicaciones del mercado actual .....	86
3.8. Instalación y configuración básica del servidor Apache .....	87
3.8.1. Definición y características .....	87
3.8.2. Instalación y configuración del servidor Apache en Debian .....	88
3.8.3. Instalación y configuración del servidor IIS en Windows Server .....	94
3.9. Estructura y recursos que componen una aplicación web. Descriptor de despliegue .....	98
3.9.1. Archivos war .....	99
3.9.2. Descriptor de despliegue .....	100
Resumen .....	101
Supuestos prácticos .....	101
Ejercicios propuestos .....	102
Actividades de autoevaluación .....	102
<b>4. ADMINISTRACIÓN DE SERVIDORES WEB .....</b>	<b>105</b>
Objetivos .....	105
Mapa conceptual .....	106
Glosario .....	106
4.1. Introducción .....	107
4.2. Configuración avanzada del servidor web .....	107
4.2.1. Directivas de control del servidor Apache .....	107
4.2.2. Parámetros del servidor .....	109
4.3. Hosts virtuales. Creación, configuración y utilización .....	110
4.3.1. Hosts virtuales basados en nombre .....	110
4.3.2. Hosts virtuales basados en IP .....	113
4.3.3. Host virtual mixto .....	116
4.4. Módulos: instalación, configuración y uso .....	116
4.5. Autenticación y control de acceso a directorios .....	119
4.5.1. Establecimiento del control de acceso .....	120
4.5.2. Autenticación básica .....	121
4.5.3. Autenticación de usuarios mediante LDAP .....	123
4.6. Certificados. Servidores de certificados .....	124
4.6.1. Módulo SSL para Apache .....	124
4.6.2. Servidor virtual seguro en Apache .....	125
4.7. Pruebas de funcionamiento, monitorización y rendimiento del servidor web .....	127
4.7.1. Registro y monitorización .....	127
4.7.2. Directivas para archivos de registro, log y error .....	129
4.7.3. Pruebas de rendimiento del servidor web .....	130
4.8. Configuración de hosts virtuales en SO Windows .....	131
Resumen .....	133
Supuestos prácticos .....	134
Ejercicios propuestos .....	134
Actividades de autoevaluación .....	135
<b>5. ADMINISTRACIÓN DE SERVIDORES DE APLICACIONES .....</b>	<b>137</b>
Objetivos .....	137
Mapa conceptual .....	138
Glosario .....	138
5.1. Introducción .....	139

<b>5.2. Arquitectura y configuración básica del servidor de aplicaciones .....</b>	139
5.2.1. Instalación del servidor de aplicaciones Apache-Tomcat .....	140
5.2.2. Arquitectura de Apache-Tomcat y variables de entorno .....	145
<b>5.3. Administrar aplicaciones web .....</b>	149
<b>5.4. Autenticación de usuarios. Dominios de seguridad para la autenticación .....</b>	153
<b>5.5. Administración de sesiones. Sesiones persistentes .....</b>	159
<b>5.6. Archivos de registro de acceso y filtro de solicitudes .....</b>	161
<b>5.7. Instalación y configuración del servidor de aplicaciones en SO Windows .....</b>	168
<b>5.8. Despliegue de aplicaciones en el servidor de aplicaciones .....</b>	169
<b>5.9. Seguridad en el servidor de aplicaciones. Configurar el servidor de aplicaciones con soporte SSL/T .....</b>	174
5.9.1. Seguridad y autenticación .....	174
5.9.2. Configuración SSL sobre Tomcat .....	175
<b>Resumen .....</b>	177
<b>Supuestos prácticos .....</b>	178
<b>Ejercicios propuestos .....</b>	179
<b>Actividades de autoevaluación .....</b>	180
<b>6. DOCUMENTACIÓN Y SISTEMAS DE CONTROL DE VERSIONES .....</b>	183
<b>    Objetivos .....</b>	183
<b>    Mapa conceptual .....</b>	184
<b>    Glosario .....</b>	184
<b>    6.1. Introducción .....</b>	185
<b>    6.2. Herramientas externas para la generación de documentación. Instalación, configuración y uso .....</b>	185
6.2.1. Instalación, configuración y uso de Javadoc .....	186
6.2.2. Instalación, configuración y uso de phpDocumentor .....	188
6.2.3. Instalación, configuración y uso de Doxygen .....	192
<b>    6.3. Formatos estándar para la documentación .....</b>	194
<b>    6.4. Creación y utilización de plantillas .....</b>	197
<b>    6.5. Herramientas colaborativas para la elaboración y mantenimiento de la documentación .....</b>	200
6.5.1. Herramienta Slack .....	200
6.5.2. Herramienta Microsoft Office 365 .....	203
<b>    6.6. Instalación, configuración y uso de sistemas de control de versiones en SO Windows .....</b>	204
6.6.1. Conceptos básicos .....	205
6.6.2. Funcionamiento del control de versiones .....	205
6.6.3. Instalación y configuración de Git en Netbeans .....	207
<b>    6.7 Operaciones avanzadas .....</b>	211
<b>    6.8. Seguridad de los sistemas de control de versiones .....</b>	214
6.8.1. Protocolo SSH .....	214
6.8.2. Protocolo HTTPS .....	215
<b>    6.9. Historia de un repositorio .....</b>	215
<b>Resumen .....</b>	216
<b>Supuestos prácticos .....</b>	217
<b>Ejercicios propuestos .....</b>	218
<b>Actividades de autoevaluación .....</b>	218

# Presentación

La informática es una ciencia en la que se producen cambios con vertiginosa rapidez. Esto se puede observar tanto en la evolución del hardware en los últimos años que, además de aumentar sus prestaciones a gran velocidad, se ha miniaturizado y convertido en ubicuo, como en el software que cada día nos permite realizar tareas que antes consumían una gran cantidad de recursos intelectuales, automatizando prácticamente todos y cada uno de los aspectos de la vida cotidiana.

En la actualidad, las aplicaciones web están en gran auge, tanto a nivel web como en aplicaciones móviles. Por ello, este libro es ideal para implementar una aplicación web, desde sus cimientos hasta el despliegue completo de la misma. El objetivo de cualquier programador web es la publicación de la aplicación desarrollada, y este libro es una guía muy útil para realizarla.

El libro se estructura en seis capítulos bien dimensionados y completos. Cada uno incluye un mapa conceptual, donde el usuario puede observar los distintos apartados que puede contener el tema; una parte teórica, donde se pueden aprender los distintos conceptos que son importantes para realizar la práctica que contiene el libro; numerosos ejemplos para entender los planteamientos; actividades, ejercicios y supuestos para poner en práctica lo aprendido, y un resumen que contempla los aspectos cruciales para comprender el tema.

En este libro se ven reflejadas las demandas formativas de los alumnos desde la experiencia docente y como jefatura de proyectos en el mundo laboral. El ámbito educativo y el profesional se aúnan para aportar a esta obra lo mejor de ambas perspectivas.

Para comenzar, cualquier tema en cualquier aspecto de la vida necesita de unos cimientos, y esa máxima se ha aplicado a los contenidos del libro. Por ello, se ha comenzado por explicar los servicios de red que intervienen en el despliegue de una aplicación web. Se sigue por las aplicaciones que son vitales para desplegar la aplicación, como son, por ejemplo, el FTP. El siguiente paso sería explicar las arquitecturas web que existen en el mercado. Llegando al ecuador del libro, es momento de explicar la administración de un servidor web y, posteriormente, cómo se

despliega una aplicación web en un servidor de aplicaciones. Y se termina con un capítulo no menos importante, como es la documentación y el sistema de control de versiones.

El libro que se presenta versa sobre el módulo Despliegue de Aplicaciones Web que pertenece al ciclo formativo de grado superior de Desarrollo de Aplicaciones Web en su segundo curso.

Este ciclo tiene como objetivo formar a profesionales que puedan desenvolverse con soltura en semejante escenario, siempre cambiante, en el que la formación continua y el autoaprendizaje deben de estar a la orden del día, por tanto, el aprendizaje a lo largo de la vida y la competencia de “aprender es aprender” deben ser ejes fundamentales en torno a los cuales se articule toda la actuación educativa.

El objetivo de este libro es que el alumnado adquiera una serie de competencias sobre el despliegue de una aplicación y su entorno, e inculcarle la importancia de realizar un buen despliegue de una aplicación significa un éxito en su utilización. Al finalizar el mismo, deben desarrollar capacidades que les permitan configurar y explotar los sistemas hardware y software relacionados con los servidores web actuales.

Las diferentes acciones formativas de formación profesional se integran en el Sistema Nacional de Cualificaciones y Formación Profesional, uno de cuyos fines esenciales es promover una oferta formativa de calidad, actualizada y adecuada a quienes se destina, de acuerdo con las necesidades de cualificación del mercado laboral y las expectativas personales de promoción profesional, según recoge el RDL 1087/2005 de 16 de septiembre. A continuación, se exponen las incluidas en este título:

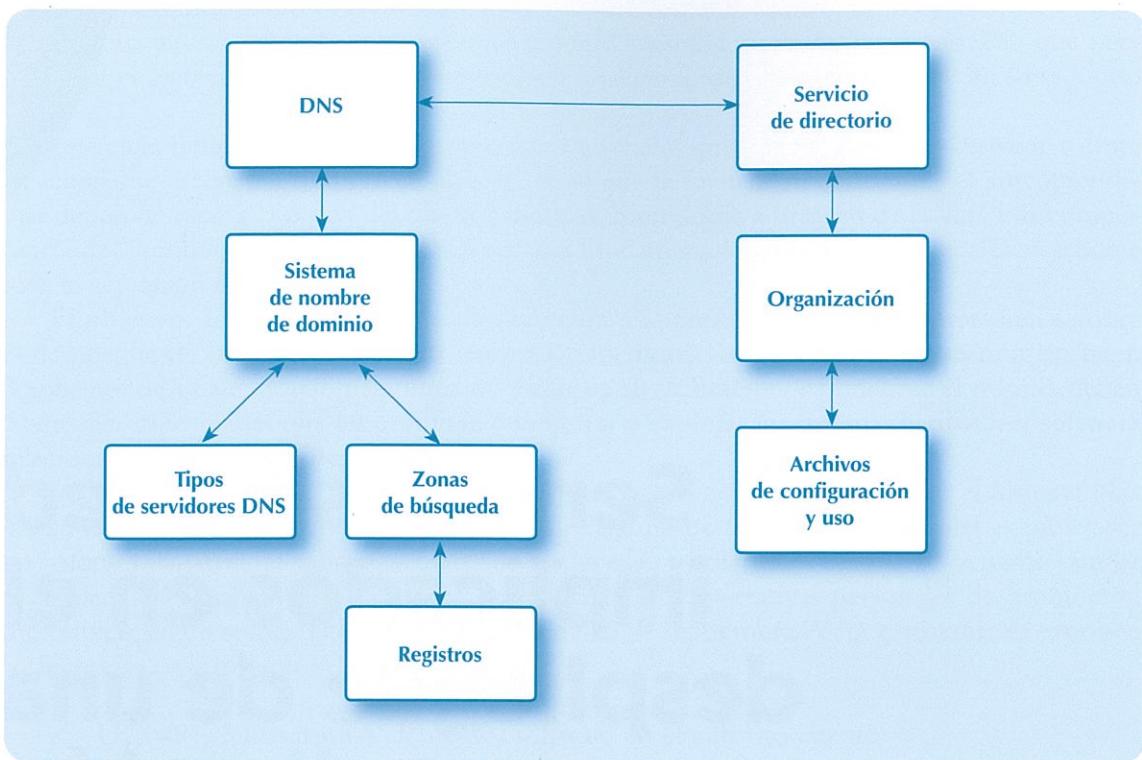
- ✓ UC0491\_3 Desarrollar elementos software en el entorno cliente.
- ✓ UC0492\_3 Desarrollar elementos software en el entorno servidor.
- ✓ UC0493\_3 Implementar, verificar y documentar aplicaciones web en entornos Internet, Intranet y Extranet.

# Servicios de red implicados en el despliegue de una aplicación

## Objetivos

- ✓ Conocer la estructura, nomenclatura y funcionalidad de los sistemas de nombres jerárquicos.
- ✓ Identificar los tipos de registros DNS para configurar las zonas directa e inversa.
- ✓ Instalar un servidor DNS en SO Linux.
- ✓ Reconocer la función, elementos y estructuras lógicas del servicio de directorio.
- ✓ Especificar los parámetros de configuración del servicio de directorio para validar usuarios.
- ✓ Instalar un servicio LDAP en SO Linux.

## Mapa conceptual



## Glosario

**Dirección IP.** Es un conjunto de caracteres que identifica plena y únicamente a un dispositivo que se conecta a una red de forma lógica y jerárquica. Está compuesto por cuatro números separados por tres puntos, cuyo intervalo va de 0 a 255. Un ejemplo de ella sería 192.168.1.5.

**Dominio.** Es un nombre único que identifica plenamente a un área o zona de Internet y que se traduce mediante el servicio DNS en una IP para poder acceder a él desde cualquier parte del mundo.

**Internet.** Es una red de redes que permite la interconexión de dispositivos mediante el conjunto de protocolos TCP/IP.

**Linux.** Sistema operativo que es multitarea, multiplataforma y multiusuario, que permite usar programas como editores de texto, navegadores, programas ofimáticos, etc. Se puede usar mediante comandos o de forma gráfica.

**Máquina virtual.** Es un software que simula una máquina física con todos sus componentes y que funciona encima de una máquina física. Puede ejecutar las mismas funciones que si fuera una máquina física.

**NFS (Network File System).** Es un protocolo a nivel de aplicación que permite el acceso a ficheros en remoto como si estuvieran en local.

**Paquetes Linux.** Son programas que se pueden instalar mediante el comando apt-get en las distribuciones Linux y que forman parte del repositorio que posea cada distribución.

**Red local.** Es un conjunto de dispositivos que se identifican mediante una IP y que normalmente se conectan a un switch de red en topología estrella para compartir recursos e información.

**Sistema operativo.** Es el software principal que permite controlar y gestionar el hardware de cualquier dispositivo, así como coordinar todas las aplicaciones y servicios que se instalan bajo su base. Los sistemas operativos actuales son Linux, Windows, Mac OS, Unix y Android. Existen bastantes más, pero parten como base de los anteriores.

**Switch.** Es un dispositivo de interconexión donde se conectan equipos o dispositivos formando normalmente una red de área local y cumpliendo los estándares del modelo TCP/IP.

## 1.1. Introducción

Una aplicación web necesita de servicios de red para poder funcionar de forma correcta y coherente. Estos servicios son el servicio DNS y el servicio de directorio LDAP.

En el caso del servicio DNS, es uno de los servicios más críticos de cualquier dispositivo que desee conectarse a Internet. Está dividido en dos zonas, directa e inversa, que definen distintos tipos de registros para que el servicio DNS permita resolverlos. Existen distintos tipos de servidores DNS en función de nuestras necesidades.

Con relación al servicio de directorio LDAP, es un software que permite la validación de usuarios de forma centralizada y permite controlar el acceso a cualquier aplicación que esté instalada en el sistema operativo. Tiene una estructura jerárquica y centralizada, a partir de una ruta de nombres se puede llegar al recurso solicitado por un cliente determinado.

## 1.2. Sistema de nombre de dominio

Es la forma en que los nombres de dominio se encuentran en Internet, que se traducen en direcciones del protocolo Internet (IP). Este nombre de dominio permite no tener que recordar la dirección IP de cada una de las páginas web que visitamos en Internet, de esta forma es más fácil recordar el nombre de una página que un número separado por puntos.

Por ejemplo, si un usuario escribe <http://www.tech.com>, el DNS asignado actúa de forma que recupera la IP para que se pueda visualizar la información que contiene ese sitio en particular.

### 1.2.1. Resolución

Casi la mayoría de la actividad de Internet se basa en los DNS, que permiten recuperar rápidamente la información correspondiente para poder conectarse al servidor o host remoto.

Existen básicamente dos formas para que el dispositivo resuelva el nombre de dominio requerido, y son las siguientes:

1. En el fichero */etc/hosts* en la ruta de cualquier SO Linux o en Windows en la ruta *C:\Windows\System32\drivers\etc* (dependerá de la versión de Windows, pero esta es la ruta estándar).
2. La segunda forma es la relacionada con los servidores DNS, que hay que escribir en la configuración de la tarjeta de red manualmente. Si existiera el servicio de DHCP, no sería necesario escribirlos manualmente.

De estas dos formas, la más habitual es la segunda, ya que no es operativo incluir en el fichero de host todas las direcciones IP y nombres a los que accedemos habitualmente.

El funcionamiento de cualquier dispositivo que se conecte a Internet es leer el fichero host y, si no se encuentra la dirección en este fichero, automáticamente se busca en los servidores DNS de forma jerárquica.

### 1.2.2. Nombre de dominio

Se define como la dirección de una empresa, organización, asociación, persona o grupos de personas en Internet. Permite que su información, sus productos y/o servicios sean accesibles en todo el mundo a través de la Red de redes.

#### RECUERDA

- ✓ Existen dos formas de resolución DNS, una mediante el archivo host y otra a través de la resolución del servidor DNS, y en ese orden.

### 1.2.3. Objetivos

En los inicios de la informática, se accedía a cualquier dispositivo tecleando la IP del host destino. Se observó que este procedimiento no era operativo y entonces se ideó el nombre de dominio con un doble objetivo:

- Es su dirección de red, ya que es la forma más fácil, rápida y práctica para encontrar un sitio en Internet.
- Es su identificación en Internet, y sirve para localizar sus productos o servicios en la Red.

### 1.2.4. Niveles de dominio

Existen tres niveles de dominios en Internet:

- a) De primer nivel: son los que terminan en .com, .gob, .org y algunos más. Estos son asignados por instituciones designadas por el ICANN. El registro de estos dominios no está sujeto a ninguna comprobación. Esto es, al primero que llega se lo asignan.
- b) De segundo nivel: son los relacionados con el país donde se dan de alta. En España corresponde a Red.es la asignación de estos dominios, que se otorgan al primero que los solicita. Son los más habituales. En España existen muchas páginas de hosting que permiten esta reserva de dominios, por ejemplo, www.arsys.es, www.freehostia.com, www.ionos.es, etc.
- c) De tercer nivel: son los correspondientes al tipo .com.es, .nom.es, .org.es, .gob.es, .edu.es. Estos dominios son diferentes, ya que seguirán un criterio de prioridad temporal en la solicitud. Además, se verificará con carácter previo a su asignación el cumplimiento de las leyes vigentes, así como el cumplimiento de las normas de sintaxis.

Para exemplificar lo anterior, entramos en detalle. Podemos teclear en cualquier navegador web la siguiente cadena de caracteres <http://www.kali.org>, que se compone de cuatro partes o cadenas separadas por puntos o dos puntos:

- ✓ *http://*: la primera parte es el protocolo de hipertexto que permite visualizar cualquier página web en un navegador de forma correcta.
- ✓ *www (World Wide Web)*: es un subdominio, en nuestro caso es de tercer nivel, que identifica el servidor web que almacena la página web.
- ✓ *kali*: es un subdominio, cuyo dominio padre en nuestro caso es org, que normalmente identifica de forma coherente al nombre de nuestra organización, empresa, organismo, etc.
- ✓ *org*: es el dominio de primer nivel que identifica a organizaciones y el padre de todos los subdominios.



#### Actividad propuesta 1.1

Busca tres ejemplos de páginas web, con distintos dominios de primer nivel.

### 1.3. Zonas de búsquedas, tipos de servidores DNS y registros

El servicio de DNS permite una ventaja fundamental de la que carece cualquier opción manual en un fichero, y es que cualquier cambio en la dirección IP o un nombre en cualquier dispositivo que esté en Internet o en una red se puede replicar a todos los servidores DNS que la configuración lo permita. Por lo que esta funcionalidad y simplicidad hacen que el servicio DNS sea fundamental en cualquier empresa u organización para su funcionamiento a nivel informático. Además, los cambios son dinámicos, como veremos en un apartado posterior.

Normalmente, los servidores DNS se componen de zonas que son las encargadas de contener los diferentes tipos de registros, en función de qué tipo de servicio ofrezca una dirección IP determinada, por ejemplo, no es lo mismo un servidor de correo que un servidor web. Existen dos tipos de zonas, que son las siguientes:

1. *Zona de búsqueda directa*: esta zona permite traducir el nombre de dominio a la dirección IP del recurso solicitado.
2. *Zona de búsqueda inversa*: los registros que se definen en esta zona permiten obtener un nombre de un dominio a partir de una dirección IP.



#### TOMA NOTA

Existen dos tipos de zonas en cualquier servicio DNS, la zona directa e inversa.

Las zonas, además de ser de búsqueda directa e inversa, pueden ser de tres tipos:

- *Master*: es un tipo de zona que crea sus propios registros, por lo tanto, no copia los ficheros de otro servidor DNS. Contiene una copia de lectura y escritura de la zona de registros. Todos los registros añadidos manualmente o automáticamente son escritos en la zona master del servidor DNS. Tales registros son almacenados en un fichero, por lo que es fácil recuperarlos en caso de fallo, previo backup. Lo ideal sería tener redundancia en otros servidores para poder acceder a la información en caso de desconexión del principal servidor.
- *Slave*: un servidor de este tipo tiene la misma información que cualquier otro servidor DNS. El objetivo principal de estos servidores DNS es ofrecer un respaldo en caso de fallo y reducir la carga de los servidores DNS principales, y, sobre todo, disminuir la carga administrativa en caso de requerir de varios servidores DNS.
- *Caché*: este tipo de servidor mantiene copias de las resoluciones de DNS que han sido buscadas en otros servidores, como pueden ser Master o Slave. En este tipo de servidores se evita tener archivos de zona. En conclusión, toda resolución de este servidor se realiza en un servidor Master o Slave.

El servicio DNS se utiliza tanto en Internet como en redes locales. Pero hay que ser restrictivo a la hora de configurar el servidor DNS en las redes locales, ya que depende del administrador del sistema que los nombres de DNS se conozcan en Internet o no.

Las ventajas de usar el DNS son las siguientes:

- ✓ No hay duplicidad de nombres. El problema se elimina debido a la existencia de dominios controlados por un único administrador. Pueden existir nombres iguales en dominios diferentes.
- ✓ Desaparece la carga excesiva en la red y en los hosts. Ahora la información está repartida por toda la red, al tratarse de una BBDD distribuida y jerárquica.
- ✓ Coherencia de la información. La información está actualizada automáticamente, sin intervención humana.

Aunque existen algunas desventajas, se pueden acotar. Son las siguientes:

- En algunos casos es fácil hackear el servicio DNS. Posteriormente comentaremos algunas medidas eficaces para evitar la intrusión.
- Es fácil de configurar, pero si se comete algún error en la configuración podría ser costoso e inoportuno; siempre queda la reinstalación.

### 1.3.1. Tipos de servidores DNS

El servicio DNS es un software que permite responder a las peticiones que realizan los clientes y que están estrechamente relacionadas con el espacio de nombres de dominio. Como norma general, los servidores DNS son dedicados para este servicio. Se suele realizar una clasificación como la siguiente:

- a) *Servidores primarios o maestros*: son servidores que guardan la información relacionada con las zonas de las que son autorizados. Sus archivos son de lectura y escritura. El administrador es el encargado de añadir, modificar o eliminar los nombres de dominio. Cualquier cambio debería ser notificado a este servidor para que de esta forma tenga la información actualizada.
- b) *Servidores secundarios o esclavos*: son un tipo de servidor que no tiene los propios archivos de zona, sino que están transferidos de un segundo o tercer nivel jerárquico. Estos servidores actúan cuando el servidor primario o maestro no puede resolver la petición por cualquier causa (servidor no disponible, caído, sin conexión, etc.). Los datos de DNS se guardan en un almacenamiento temporal llamado caché para peticiones futuras.
- c) *Servidores locales o caché*: estos servidores se configuran para mejorar los tiempos de respuesta de las consultas, reducir la carga de los equipos y disminuir el tráfico de la red. Por lo tanto, su función es contactar con otros servidores para resolver las peticiones de los clientes. Un servidor es solo caché cuando se dan las siguientes circunstancias:
  - Realiza las peticiones a otros servidores DNS para tener las respuestas preparadas para futuras solicitudes.
  - No tiene autoridad sobre ninguna zona.

RECUERDA

- ✓ Hay tres tipos de servidores DNS: primarios, secundarios y caché.

### 1.3.2. Registros DNS

Las zonas que se encuentran definidas en cada uno de los servidores DNS están compuestas de registros, que se definen como archivos de mapeo que le indican al servidor DNS a qué

dirección IP está asociado un nombre en particular o un dominio, depende de cómo se defina el registro. La cantidad total de registros DNS que se pueden definir asciende a un total de 25 (normalmente se usan menos en la zona DNS). A continuación, se detallan los registros más usados en las configuraciones:

**CUADRO 1.1**  
**Registros DNS**

Tipo de registro	Descripción	Sintaxis
A (Address)	Traduce nombres de dominio en direcciones IP.	new.com A xxx.xxx.xxx.xxx
PTR (Pointer)	Traduce direcciones IP en nombres de dominio.	3.0.0.20.in-addr.arpa PTR host.new.com
MX (Mail Exchanger)	Asocia un nombre de dominio a un servidor de correo. El número 10 indica preferencia; a menor número, mayor preferencia.	new.com MX 10 correo.new.com
CNAME	Es un alias que se le asigna a un host que tiene una dirección IP.	Alias.new.com CNAME nombre.new.com
NS	Define los servidores principales de un dominio; al menos debe haber uno.	new.com IN NS servidor1.new.com
SOA	Es el primer registro de la zona; solo puede haber uno configurado. Especifica el servidor DNS primario del dominio. Pieza clave del archivo de zona.	Es un tipo de registro que especifica información del DNS. Los campos se definen más adelante.
TXT	Ofrece información adicional a un dominio. También se usa como almacenamiento en claves de cifrado.	new.com TXT "Informacion adicional"
SPF	Es un registro de tipo texto que se crea en la zona directa del DNS. Se usa principalmente para evitar la suplantación de identidad.	new.com IN SPF "v=spf1 a:exchange.new.com -all"

**Actividad propuesta 1.2**



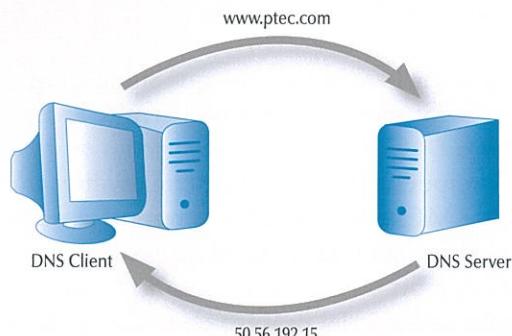
Crea los distintos tipos de registros y observa cómo se comportan.

## 1.4. Funcionamiento del servicio DNS y tipos de consultas

Cuando un dispositivo se conecta a una red local o Internet y hace alguna búsqueda de algún recurso (servidor), obligatoriamente debe usar para resolver la IP el servicio DNS y, por lo

tanto, realizar una consulta que puede proceder de una aplicación que se ejecute en el cliente o del propio cliente.

El cliente envía un mensaje de consulta al servidor DNS, que contiene un nombre totalmente calificado (FQDN), un tipo de consulta y la clase del nombre del dominio.



**Figura 1.1**  
Funcionamiento del servicio DNS

Las consultas DNS pueden seguir un flujo diferente, dependiendo de quién le responda al mensaje. Se comenta el proceso que sigue a la consulta a grandes rasgos, comenzando por el cliente que lanza una petición de DNS:

1. El cliente tiene una caché, que almacena los registros que se usan habitualmente y, por lo tanto, la respuesta a la petición puede estar localmente.
2. El siguiente paso, en caso de no encontrar la respuesta, es consultar al DNS, que puede responder desde su caché y terminar el proceso.
3. Si no responde el servidor DNS desde la caché, puede responder desde su zona, que tiene configurada y responde a la petición.
4. El servidor DNS puede consultar a otros servidores DNS para resolver el nombre que solicita el recurso. A este proceso se le llama *recursividad*, que trataremos con posterioridad.
5. La otra opción es que el mismo cliente realice la consulta a otros servidores DNS, a este proceso se llama *iteración*.



### Actividad propuesta 1.3

Explica cómo funciona el servicio DNS desde que un dispositivo consulta una página web en Internet.

Los clientes para resolver los nombres de dominio realizan peticiones a los DNS preferidos, ya que normalmente son dos, uno primario y otro secundario. Tales servicios son los encargados de resolver las solicitudes, y en caso de no existir la respuesta también se notifica al cliente. Por lo tanto, se van a detallar a continuación los tipos de consultas DNS que existen en la actualidad:

### 1.4.1. Consulta recursiva

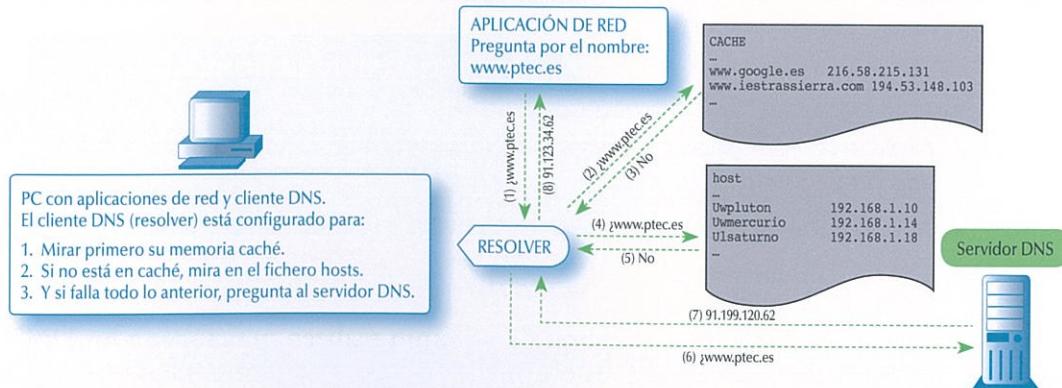
Es aquella a la que el servidor que alberga el servicio DNS dará la respuesta a la petición del PC o dispositivo. El servicio DNS no tiene la obligación de soportar este tipo de consultas, sino que es el dispositivo el que las negocia.

La respuesta a este tipo de consultas puede ser de tres tipos:

- ✓ Un error NXDOMAIN, que significa que el dominio o el equipo no existen.
- ✓ Un error temporal, que no permite acceder al servidor DNS por problemas de conectividad.
- ✓ La respuesta a la petición con la dirección del registro A y acompañada del registro CNAME, si existiera. En la respuesta siempre se indica si es autoritaria o no. Esto es, si la respuesta es dada por el mismo o por un servidor exterior.

Ejemplo de flujo del proceso de petición de IP:

1. El usuario inicia una aplicación que necesita de un servidor externo o teclea una URL en el navegador <http://www.ptec.com>.
2. El navegador o la aplicación consulta al dispositivo cuál es la IP de [www.ptec.com](http://www.ptec.com).
3. El PC pregunta al servicio de DNS configurado por defecto.
4. El servicio de DNS mira si la respuesta se encuentra en la caché local. Si la encuentra, termina el proceso, en caso contrario, continúa la búsqueda. La acción siguiente sería buscar en el fichero host de la máquina local.
5. El servicio de DNS le envía la pregunta a uno de los servidores raíz o primario.
6. El servidor que alberga el servicio DNS soporta consultas iterativas y responde con las IP disponibles de los servidores autoritarios del siguiente nivel. Busca en el nivel autoritario .com.
7. El servicio DNS selecciona una IP de la lista recibida, y le envía la pregunta a dicho servidor de nombres que posee el fichero de zona del dominio .com.
8. El siguiente paso es preguntar al siguiente nivel de dominio que es ptec.com
9. En este servidor del dominio ptec.com, dispondrá de una zona con nombres del dominio consultado.



**Figura 1.2**  
Flujo de petición de IP

10. El servidor devolverá la petición al dominio consultado www.ptec.com que estará formada por los registros A y CNAME, si los hubiera.
11. El servicio DNS envía la respuesta completa para el dispositivo que realiza la petición.
12. Este servicio envía al PC la primera IP que coincide con la petición.

### 1.4.2. Consulta iterativa

Es aquella a la que el servidor DNS responderá de forma parcial al cliente DNS del PC o dispositivo.

La respuesta a este tipo de consultas puede ser de tres tipos:

- Un error NXDOMAIN, que significa que el dominio o el equipo no existen.
- Un error temporal, que no permite acceder al servidor DNS por problemas de conectividad.
- La respuesta a la petición con la dirección del registro A y acompañada del registro CNAME, si existiera. En la respuesta siempre se indica si es autoritaria o no. Esto es, si la respuesta es cacheada o no.
- Una lista de servidores para preguntar por la petición del PC o dispositivo para que se pueda avanzar en la búsqueda de la IP solicitada. Esta respuesta es habitual en los servidores raíz o TLD (*Top Level Domain*), un tipo de servidores que solo soporta consultas iterativas.

### 1.4.3. Consulta inversa

Es una petición que se realiza cuando un PC o dispositivo quiere saber el nombre del dominio al que pertenece un registro en particular de cualquier tipo, por ejemplo, de tipo servidor de correo. Aunque este tipo de consultas se han convertido en obsoletas.

Hay que comentar que estas consultas se confunden a veces con el mapeo inverso o búsqueda inversa, que a partir de una IP se obtiene el nombre relacionado y que se localiza en las zonas con la opción in-addr.arpa.



#### Actividad propuesta 1.4

Expón las diferencias entre la consulta inversa, iterativa y recursiva. Razona tu respuesta.

## 1.5. Instalación y configuración de un servidor DNS en SO Linux

La instalación que se va a realizar es del servicio bind9 y sus utilidades. Más concretamente, el servicio DNS para SO Linux bajo la versión de Debian (Kali Linux). La instalación se puede realizar en una máquina virtual que posee una imagen de este sistema operativo o en una máquina física. El procedimiento de instalación es el siguiente:

1. Para empezar, es necesario tener conexión a Internet desde la máquina y tener configurados correctamente los repositorios que se encuentran localizados en la ruta `/etc/apt/sources.list`. A partir de aquí se ejecutan los comandos `apt-get update` y `apt-get upgrade`. Se comentan las opciones anteriores:
  - `update`: actualiza la lista de paquetes disponibles y versiones, pero no actualiza ningún paquete. Esta lista se selecciona de los servidores con repositorios que se definen en el `sources.list`.
  - `upgrade`: instalará las nuevas versiones respetando la configuración del software, cuando sea posible.
2. Antes de comenzar a instalar el paquete bind9 (servicio DNS), se debería configurar una IP en la máquina o PC en que se va a instalar este servicio. En el cuadro 1.2 se muestran algunos comandos útiles para tratar la IP.

#### CUADRO 1.2

#### Comandos para tratar IP

Comando	Descripción
<code>#ifconfig</code>	Muestra y modifica la configuración de la red.
<code>#ifconfig eth0</code>	Visualiza la configuración de la interfaz eth0.
<code>#ifconfig eth0 down/up</code>	Activa y desactiva la interfaz eth0.
<code>#ifconfig eth0 add 192.168.1.10</code>	Configura la IP para la interfaz eth0.
<code>#ifconfig eth0 mask 255.255.255.0</code>	Configura la máscara para la interfaz eth0.
<code>#ifconfig eth0 promisc</code>	Activa el modo promiscuo.
<code>#ifconfig eth0 -promisc</code>	Desactiva el modo promiscuo.
<code>#ifconfig eth0 hw ether XX:XX:XX:XX:XX:XX</code>	Cambia la MAC de la tarjeta de red.

Otra forma de configurar la IP sería en el fichero `/etc/network/interfaces`, de forma que se puede fijar la dirección IP, máscara y router por defecto en tal fichero.

Al ejecutar el comando `ifconfig`, se observa la pantalla de la figura 1.3.

```
root@osboxes: ~
Archivo Editar Ver Buscar Terminal Ayuda
root@osboxes: # ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
        inet 10.0.2.15 netmask 255.255.255.0 broadcast 10.0.2.255
              inet6 fe80::f402:b717:1bf:4da9 prefixlen 64 scopeid 0x20<link>
                ether 08:00:27:f6:51:97 txqueuelen 1000 (Ethernet)
                  RX packets 23 bytes 3164 (3.0 Kib)
                  RX errors 0 dropped 0 overruns 0 frame 0
                  TX packets 41 bytes 3327 (3.2 Kib)
                  TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
        inet 127.0.0.1 netmask 255.0.0.0
              inet6 ::1 prefixlen 128 scopeid 0x10<host>
                loop txqueuelen 1000 (Local Loopback)
                  RX packets 24 bytes 1360 (1.3 Kib)
                  RX errors 0 dropped 0 overruns 0 frame 0
                  TX packets 24 bytes 1360 (1.3 Kib)
                  TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Figura 1.3  
Comando `ifconfig`

- Desde aquí comienza la instalación de bind9 y de sus utilidades, y ello se realiza a partir del siguiente comando (figura 1.4).

```
# apt-get install bind9 bind9utils
```

```
Archivo Editar Ver Buscar Terminal Ayuda
root@osboxes:~# apt-get install bind9 bind9utils
Leyendo Lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
Los paquetes indicados a continuación se instalaron de forma automática y ya no
son necesarios.
  libbind9-160 libdns1102 libisc169 libisccc160 libisccfg160 liblwres160
Utilice «apt autoremove» para eliminarlos.
Paquetes sugeridos:
  bind9-doc resolvconf ufw
Se instalarán los siguientes paquetes NUEVOS:
  bind9 bind9utils
0 actualizados, 2 nuevos se instalarán, 0 para eliminar y 2131 no actualizados.
Se necesita descargar 0 B/1.099 kB de archivos.
Se utilizarán 3.919 kB de espacio de disco adicional después de esta operación.
Preconfigurando paquetes ...
Selecciónando el paquete bind9utils previamente no seleccionado.
(Leyendo la base de datos ... 319505 ficheros o directorios instalados actualmen
te.)
```

**Figura 1.4**

Instalación del servicio bind9 y sus utilidades

**RECUERDA**

- ✓ El comando para instalar un servicio en Linux es *apt-get install nombre\_servicio*.

- Posteriormente, se puede comprobar que el servicio DNS (bind9) se ha instalado correctamente. Para ello se usan los siguientes comandos (figura 1.5).

```
# service bind9 start
# service bind9 status
```

```
Archivo Editar Ver Buscar Terminal Ayuda
root@osboxes:~# service bind9 start
root@osboxes:~# service bind9 status
● bind9.service - BIND Domain Name Server
   Loaded: loaded (/lib/systemd/system/bind9.service; disabled; vendor prese
   Active: active (running) since Fri 2021-01-29 13:42:26 EST; 5s ago
     Docs: man:named(8)
   Process: 1212 ExecStart=/usr/sbin/named $OPTIONS (code=exited, status=0/SL
 Main PID: 1213 (named)
    Tasks: 4 (limit: 2368)
   Memory: 15.4M
      CGroup: /system.slice/bind9.service
              └─1213 /usr/sbin/named -u bind

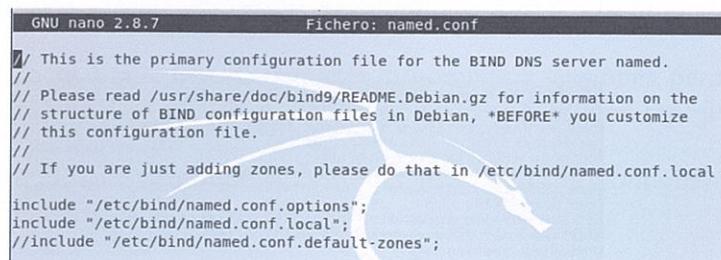
ene 29 13:42:26 osboxes named[1213]: network unreachable resolving './NS/IN':
```

**Figura 1.5**

Comandos de control del servicio bind9

- Una vez comprobado que el servicio está correcto, se está en disposición de configurar nuestro servidor DNS. Antes de comenzar a configurarlo, es necesario realizar una copia de seguridad de los ficheros de configuración que se van a modificar, por si es necesario deshacer los cambios realizados y comenzar desde el principio. El primer

fichero que lee el servicio bind9 es el named.conf. No es necesario modificar rdyr en principio; tiene la estructura que se muestra en la figura 1.6.



```

GNU nano 2.8.7          Fichero: named.conf
// This is the primary configuration file for the BIND DNS server named.
//
// Please read /usr/share/doc/bind9/README.Debian.gz for information on the
// structure of BIND configuration files in Debian, *BEFORE* you customize
// this configuration file.
//
// If you are just adding zones, please do that in /etc/bind/named.conf.local
include "/etc/bind/named.conf.options";
include "/etc/bind/named.conf.local";
//include "/etc/bind/named.conf.default-zones";

```

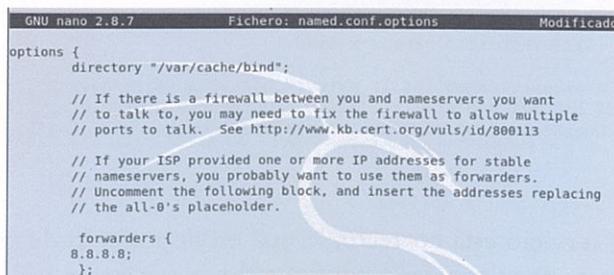
**Figura 1.6**  
Fichero de configuración named.conf

Este fichero incluye tres ficheros importantes que va a leer el servicio bind9, que son los siguientes:

- *named.conf.options*: en este primer fichero se define la caché de nuestro DNS, y la configuración genérica del servidor, como puede ser la transferencia de zonas, forwarders, etc. En la figura 1.7 se muestra el fichero.

La configuración más importante de este fichero son los forwarders (reenviadores), ya que es necesario configurarlos para que cuando un cliente realice una consulta DNS y no encuentre la respuesta en local, la respuesta se pueda encontrar en Internet, como puede ser el nombre de un servidor, un sitio web, un servidor de correo, etc. Por lo general, son los servidores DNS de nuestro ISP, Google, Telefónica, etc. Se puede poner por ejemplo el DNS de Google, 8.8.8.8.

- *named.conf.local*: este fichero es tan importante como el anterior, y es donde se definen las zonas de búsqueda directa e inversa. Como ejemplo se va a localizar un dominio denominado www.prueba.com, y los ficheros para la zona directa e inversa son respectivamente *prueba.local* y *prueba.127*, para seguir una nomenclatura. Las zonas en un principio son de tipo master (recordamos que las zonas puede ser de tres tipos master, slave y cache como comentamos anteriormente). En la figura 1.8 se muestra el fichero.
- *named.conf.default-zones*: es un fichero donde se definen las zonas por defecto. La inclusión de este fichero se puede comentar (ver figura 1.6). Realmente donde se definen las zonas nuevas es en el fichero *named.conf.local*.



```

GNU nano 2.8.7          Fichero: named.conf.options          Modificado
options {
    directory "/var/cache/bind";

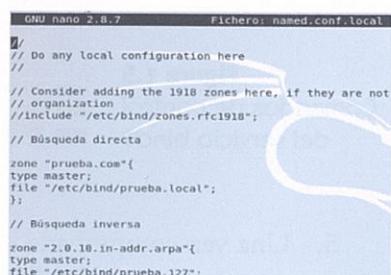
    // If there is a firewall between you and nameservers you want
    // to talk to, you may need to fix the firewall to allow multiple
    // ports to talk. See http://www.kb.cert.org/vuls/id/800113

    // If your ISP provided one or more IP addresses for stable
    // nameservers, you probably want to use them as forwarders.
    // Uncomment the following block, and insert the addresses replacing
    // the all-0's placeholder.

    forwarders {
        8.8.8.8;
    };
}

```

**Figura 1.7**  
Fichero de configuración named.conf.options



```

GNU nano 2.8.7          Fichero: named.conf.local
//
// Do any local configuration here
//

// Consider adding the 1918 zones here, if they are not
// organization
//include "/etc/bind/zones.rfc1918";

// Búsqueda directa
zone "prueba.com"{
    type master;
    file "/etc/bind/prueba.local";
};

// Búsqueda inversa
zone "2.0.10.in-addr.arpa"{
    type master;
    file "/etc/bind/prueba.127";
};

```

**Figura 1.8**  
Fichero de configuración named.conf.local

6. Los siguientes ficheros para configurar son prueba.local y prueba.127. Tales ficheros nos permitirán definir los registros DNS, que posteriormente se van a resolver para dar respuesta a las peticiones de nuestros clientes DNS. A continuación, se van a mostrar los ficheros que se han configurado para observar el funcionamiento de nuestro servidor DNS. Para no cambiar la configuración de IP, se ha dejado la IP actual que es 10.0.2.15 y que se puede observar mediante la ejecución del comando ifconfig.

El fichero prueba.local sería de la siguiente forma:

```

Archivo Editar Ver Buscar Terminal Ayuda
GNU nano 2.8.7 Fichero: prueba.local

$TTL 604800
@ IN SOA prueba.com. root.prueba.com. (
    2 ; Serial
    604800 ; Refresh
    86400 ; Retry
    2419200 ; Expire
    604800 ) ; Negative Cache TTL
;
@ IN NS prueba.com.
@ IN A 10.0.2.15
dns IN A 10.0.2.15
server2003 IN A 10.0.2.20
www IN CNAME prueba.com.

```

**Figura 1.9**  
Fichero de configuración prueba.local

Se van a detallar a continuación las diferentes opciones que posee el fichero de zona directa:

- **\$TTL (Time to Live):** indica la duración en segundos que se conservarán los datos en memoria caché.
- **Nombre\_zona:** FQDN de la zona administrada por este archivo, los nombres de zona deben acabar en punto, de lo contrario dará error en el chequeo de los archivos. Usualmente se pone @ para no cargar demasiado al archivo. Es necesario declarar el registro NS y A para que la zona conozca tanto el dominio como la IP del servidor DNS que suministra la zona.
- **IN:** esta opción es obsoleta pero es la única que se puede usar hasta la fecha. Es la clase de Internet.
- **SOA (Start of Authority):** registro obligatorio para indicar que el servidor actual es el propietario y legítimo de esta zona.
- **Serial:** número de serie del archivo, se usa cuando la zona se replica a otros servidores.
- **Refresh:** valor numérico que se utiliza cuando la zona se replica a un servidor esclavo, y el intervalo con el que se comprueba la validez.
- **Retry:** es un valor numérico que indica el tiempo que pasa hasta que contacta el servidor esclavo con el servidor maestro.
- **Expire:** es otro valor numérico que indica cuántos segundos como máximo el servidor retendrá los registros antes de expirarlos.
- **Negative:** indica cuánto tiempo el servidor debe conservar en su caché la respuesta negativa.
- **NS:** registro que indica cuál es el servidor de nombres para esta zona.

- Los demás registros que se indicaron anteriormente se declaran de la forma que se puede observar en la imagen anterior. Por ejemplo: *server2003 IN A 10.0.2.15*. El fichero prueba.127 sería de la siguiente forma:

```

GNU nano 2.8.7 Fichero: prueba.127

; BIND reverse data file for local loopback interface
;
$TTL 604800
@ IN SOA prueba.com. root.prueba.com. (
    1 ; Serial
    604800 ; Refresh
    86400 ; Retry
    2419200 ; Expire
    604800 ) ; Negative Cache TTL
;
@ IN NS prueba.com.
15 IN PTR prueba.com.
20 IN PTR server2003.prueba.com.

```

**Figura 1.10**  
Fichero de configuración  
prueba.127

Si se observa el fichero anterior, se comprobará que las opciones son prácticamente las mismas, salvo la diferencia a la hora de declarar los registros, ya que estamos en una zona inversa, entonces la declaración de un registro sería de la forma *15 IN PTR prueba.com.*

El número 15 sería el último byte de la dirección IP del dominio, IN para declarar un registro y la opción nueva es la siguiente:

- *PTR*: crea la correspondencia de un nombre con una dirección IP. Además de SOA y NS, son los únicos registros que se encuentran en las zonas inversas.
7. El siguiente fichero que es necesario configurar es resolv.conf, que nos permitirá decirle quiénes son los servidores DNS, el dominio de nuestro servidor y dónde es necesario realizar la búsqueda. Por lo tanto, la configuración de nuestro fichero se ve en la figura 1.11.

```

Archivo Editar Ver Buscar Terminal Ayuda
GNU nano 2.8.7 Fichero: resolv.conf

# Generated by NetworkManager
domain prueba.com
search prueba.com
nameserver 10.0.2.15

```

**Figura 1.11**  
Fichero de configuración  
resolv.conf

8. Por último, habría que configurar el fichero de host para la máquina en cuestión, pero para la resolución de DNS no es necesario.
9. A continuación, se va a comprobar que el DNS está funcionando correctamente con los registros que hemos declarado. Las pruebas se pueden hacer con varios comandos de cliente DNS, como pueden ser nslookup, dig o host. En nuestro caso, vamos a usar dos comandos nslookup y dig para que se observen las distintas opciones, una más breve y otra más extensa.

Con el comando nslookup y consultando zona directa e inversa el resultado sería el siguiente:

```
root@osboxes:/etc/bind# nslookup
> prueba.com
Server:      10.0.2.15
Address:     10.0.2.15#53

Name:   prueba.com
Address: 10.0.2.15
> server2003
Server:      10.0.2.15
Address:     10.0.2.15#53

Name:   server2003.prueba.com
Address: 10.0.2.20
> 10.0.2.15
15.2.0.10.in-addr.arpa name = prueba.com.
> 10.0.2.20
20.2.0.10.in-addr.arpa name = server2003.prueba.com.
> 
```

**Figura 1.12**  
Comando nslookup

Como se puede observar en la figura 1.12, se han realizado las siguientes consultas:

- ✓ Prueba.com: el resultado es la IP 10.0.2.15
- ✓ Server2003: el resultado es la IP 10.0.2.20
- ✓ 10.0.2.15: el resultado es el dominio creado por nosotros prueba.com
- ✓ 10.0.2.20: el registro de tipo A server2003.prueba.com

Con el comando dig y consultando zona directa *prueba.com* el resultado sería el siguiente:

```
Archivo Editar Ver Buscar Terminal Ayuda
root@osboxes:/etc# dig prueba.com

; <>> DiG 9.11.16-2-Debian <>> prueba.com
; global options: +cmd
; Got answer:
; ->>HEADER<- opcode: QUERY, status: NOERROR, id: 41371
; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 1
;
; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: eae164706481393cf64071b5e6fcadcc8a2f7c3d8de0ba50 (good)
; QUESTION SECTION:
;prueba.com.           IN      A
;
; ANSWER SECTION:
prueba.com.        604800  IN      A      10.0.2.15
;
; AUTHORITY SECTION:
prueba.com.        604800  IN      NS     prueba.com.

; Query time: 0 msec
; SERVER: 10.0.2.15#53(10.0.2.15)
; WHEN: lun mar 16 14:52:12 EDT 2020
; MSG SIZE rcvd: 97
```

**Figura 1.13**  
Comando dig

El resultado de la ejecución del comando dig acompañado del dominio prueba.com nos da como resultado:

- La pregunta es que IP tiene el dominio prueba.com y como respuesta primera es el registro tipo A 10.0.2.15.
- Como segunda opción de sección es el registro NS de dominio principal prueba.com.
- Además, esta consulta da otros datos más: cuándo se ha realizado la consulta, el tamaño del mensaje, el tiempo que ha tardado, el servidor que ha respondido a la consulta.

Con el comando dig y consultando zona inversa 10.0.2.15 el resultado sería el siguiente:

```
Archivo Editar Ver Buscar Terminal Ayuda
root@osboxes:/etc# dig -x 10.0.2.15

; <>> DiG 9.11.16-2-Debian <>> -x 10.0.2.15
;; global options: +cmd
;; Got answer:
;; ->HEADER<- opcode: QUERY, status: NOERROR, id: 37039
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 2
;;
;; OPT PSEUDOSECTION:
;; EDNS: version: 0, flags:; udp: 4096
;; COOKIE: 6de34f810a262153ec6b4f285e6fcbe801c115fd785abfa4 (good)
;; QUESTION SECTION:
;15.2.0.10.in-addr.arpa.           IN      PTR
;;
;; ANSWER SECTION:
15.2.0.10.in-addr.arpa. 604800  IN      PTR    prueba.com.
;;
;; AUTHORITY SECTION:
2.0.10.in-addr.arpa.   604800  IN      NS     prueba.com.
;;
;; ADDITIONAL SECTION:
prueba.com.            604800  IN      A      10.0.2.15
;;
;; Query time: 0 msec
;; SERVER: 10.0.2.15#53(10.0.2.15)
;; WHEN: lun mar 16 14:56:40 EDT 2020
```

**Figura 1.14**  
Comando dig -x

Para concluir, se van a detallar dos comandos que nos permiten chequear sintácticamente los ficheros que se han creado anteriormente, como son los siguientes: named.conf, named.conf.options, named.conf.local, prueba.local, prueba.127.

### CUADRO 1.3

#### Comandos de comprobación sintáctica

Comando	Descripción	Sintaxis
named-checkconf	Detecta posibles errores sintácticos en los ficheros de configuración named.	named-checkconf <b>fichero_configuración</b>
named-checkzone	Detecta posibles errores sintácticos en los ficheros de configuración de zona.	named-checkzone <b>nombreDominio fichero_de_zona</b>

```
root@osboxes:/etc/bind# named-checkconf named.conf
root@osboxes:/etc/bind# named-checkconf named.conf.local
root@osboxes:/etc/bind# named-checkconf named.conf.options
root@osboxes:/etc/bind# named-checkzone prueba.com /etc/bind/prueba.local
zone prueba.com/IN: loaded serial 2
OK
root@osboxes:/etc/bind# named-checkzone 10.0.2.15 /etc/bind/prueba.127
zone 10.0.2.15/IN: loaded serial 1
OK
root@osboxes:/etc/bind#
```

**Figura 1.15**  
Ejemplo de ejecución de comandos named-checkconf y named-checkzone



#### TOMA NOTA

Los comandos para comprobar si el servicio DNS funciona son: host, dig y nslookup.

## 1.6. Servicio de directorio: características y funcionalidad

Un directorio en cualquier SO es una base datos preparada para navegar, leer y buscar información almacenada en ella. Los directorios permiten encontrar información de forma electrónica, a diferencia de los directorios clásicos, como una guía telefónica donde se busca por apellidos ordenados alfabéticamente el teléfono de una persona en concreto.

Estos directorios electrónicos no contemplan transacciones complicadas, ni opción de vuelta atrás como los que se pueden encontrar en los sistemas de bases de datos para gestionar grandes cantidades de información. La actualización en estos directorios es mediante cambios simples, siempre que se permita en la definición del directorio. De todas formas, los directorios están preparados para dar respuesta a grandes solicitudes de búsqueda; también tienen la capacidad de incrementar la disponibilidad y la fiabilidad cuando se replica la información.

Las funciones básicas de un directorio son las siguientes:

- a) *Buscar información*: es la función principal por la cual se crea el directorio electrónico. Es accedida de múltiples formas, ya sea por nombre o por cualquier otro campo que esté implementado en el directorio.
- b) *Gestionar información*: es otra función primordial, ya que es la que permite agregar, editar o borrar usuarios por distintos campos. Normalmente se centraliza la información en un directorio, al cual acceden las aplicaciones. En caso de que exista más de un directorio, habría que sincronizar todos los directorios para que posean la misma información, de cara a las aplicaciones que accedan. Un ejemplo claro de este directorio es el acceso al directorio de Windows por parte de los usuarios de la red de una empresa.
- c) *Control de seguridad*: la última función sería controlar el acceso por parte de los usuarios, ya que la seguridad es un aspecto crucial en la gestión de los usuarios. Además de la seguridad, el servicio de directorio gestiona los certificados digitales de los usuarios del directorio, cuyas funciones serían:
  - Creación.
  - Distribución.
  - Destrucción.
  - Ubicación.

En conclusión, las aplicaciones que acceden a los servicios de directorio son muy diversas, desde aplicaciones web, pasando por el correo electrónico, acceso a edificios, sistemas operativos y una cantidad ingente de aplicaciones y de accesos que hoy en día se usan. Sin ir más lejos, el servicio de DNS es un servicio de directorio distribuido.



**Figura 1.16**  
Servicio de directorio

## 1.7. Organización de LDAP

La organización de este tipo de directorios puede estar centralizada o distribuida según interese el diseño, por lo que se comentan las dos vertientes:

1. *Centralizado*: en este caso todas las consultas se canalizan a un único servidor, y todas ellas son respondidas por él. Como ventaja tiene que no es necesario sincronizarlo a otros servidores, pero el principal inconveniente es que en caso de fallo el sistema se queda sin validación. Como mínimo sería necesario realizar un backup del servicio de directorio.
2. *Distribuido*: esta opción permite que la información esté dividida en varios servidores que permiten responder ante las consultas de los clientes. Los datos pueden estar fraccionados o replicados:
  - Cuando la información está fraccionada, los servidores que intervienen en el servicio de directorio no contienen toda la información, sino un subconjunto de ella.
  - Cuando la información está replicada, toda la información del directorio forma parte de todos los servidores que componen el cluster del servicio de directorio.

La opción más viable para configurar y dar servicio de directorio es una mezcla de las anteriores opciones, esto es, una parte de la información estará replicada y otra fraccionada. Sería lo óptimo para contrarrestar los inconvenientes de una y otra opción.



#### PARA SABER MÁS

El primer directorio de este tipo surgió en la década de los ochenta creado por la UIT (<https://www.itu.int/>) y se denominó X.500. Organiza las entradas en el directorio de forma jerárquica, y con gran cantidad de almacenamiento de datos, lo que provoca dos grandes ventajas: las facilidades de búsqueda de información y la ampliación de la arquitectura para que sea altamente escalable. X.500 necesitaba de un protocolo de conexión entre el cliente y el servidor, por lo que al principio decidieron usar DAP, pero deberían usar todos los protocolos subyacentes en el modelo OSI. Ahí es donde surge LDAP como éxito de conexión entre el cliente y el servidor.

LDAP (*Lightweight Directory Access Protocol*) surge como alternativa a DAP. Se trata de una serie estándares definidos por IETF que podemos observar en varios RFC (<https://ldap.com/ldap-related-rfcs/>). Las funciones más llamativas y que llevaron al éxito de LDAP son las siguientes:

- ✓ LDAP usa TCP/IP en lugar de protocolos teóricos del modelo OSI, que requiere menos recursos y es más accesible.
- ✓ LDAP representa la información mediante cadenas de caracteres en lugar de complicadas estructuras ASN.1.
- ✓ El modelo funcional de LDAP es más simple y ha eliminado las opciones raras usadas en X.500, por lo que es más intuitivo a la hora de comprenderlo e implementarlo.

LDAP define un protocolo para el contenido de los mensajes entre un cliente y el servidor propiamente dicho. Este protocolo permite acceder a las operaciones definidas por el cliente, las respuestas del servidor y los datos transportados en el mensaje.

La implementación de LDAP se realiza mediante Open LDAP de código abierto desarrollada por OpenLDAP Project y posee las siguientes características:

- Es multiplataforma.
- El código es de licencia libre.
- Basado en el estándar X.500.
- Tiene estructura de árbol denominado DIT.
- Soporta IPv3, LDAPv3 y esquema distribuido.
- Internacionalización mediante el uso de caracteres UTF-8.
- En el mundo Linux tiene magnífica integración con otras aplicaciones.
- Tiene mecanismos de búsqueda avanzados.

Entrando en profundidad, se va a analizar el estándar LDAP desde diferentes puntos de vista que componen el servicio de directorio:

1. *Modelo de información:* este modelo nos permite darle forma a la estructura de la información almacenada en LDAP. El dato básico en el directorio es una entrada que corresponde con un objeto del mundo real. Una entrada se compone de un conjunto de atributos, cada uno de ellos tiene un tipo con sus valores. El tipo como en cualquier lenguaje de programación define la información que se va a almacenar y los valores son la información en sí. Todos los atributos tienen un identificador llamado OID y una sintaxis que permite definir qué valores va a poseer. Se aproxima a la forma de almacenar la información en Windows Server 2012, que se basa también en LDAP. Un ejemplo de este tipo sería el siguiente:

```
dn: dc=ejemplo, dc=com
```

La clave de este tipo de modelo es que cada servidor puede definir su estructura o esquema, aunque se tienda a que el esquema de directorio sea estándar para todos los servidores que intervienen en la compartición del servicio de directorio.

2. *Modelo de referencia:* a la hora de nombrar los datos LDAP define cómo se organizan y referencian estos, primero se definen las estructuras de cómo se organizan las entradas y posteriormente se indica cómo referenciar o acceder a las mismas.

## 1.8. Archivos básicos de configuración y uso

El formato de intercambio de datos LDAP (LDIF) es una extensión de archivo de texto sin formato usada para almacenar datos del directorio LDAP (*Lightweight Directory Access Protocol*) como un conjunto de registros y solicitudes de actualizaciones de LDAP que incluyen agregar, eliminar, modificar y cambiar nombre. LDAP es un protocolo de software que determina la ubicación de los archivos, dispositivos, organización a través de la red TCP/IP.

LDIF fue creado en la década de los 90 por Tim Howes, Mark C. Smith y Gordon en una primera versión y posteriormente se actualizó y amplió a finales de 1990 para el uso con la versión 3 de LDAP. El RFC que posee toda la normativa es el 2849 y fue publicado en junio del año 2000.

La sintaxis que posee LDIF es de la siguiente forma:

```
dn: <nombre distinguido>
<nombre_atributo>: <valor>
<nombre_atributo>: <valor>
<nombre_atributo>: <valor>
```

Por lo tanto, una entrada del directorio en formato de intercambio de datos LDIF consta de dos partes:

1. DN que debe figurar en la primera línea de entrada y que se compone de la cadena dn: seguida del nombre distinguido (DN) de la entrada.
2. La segunda parte son los atributos de la entrada, como se puede observar en el ejemplo anterior.

No existe ningún orden prestablecido, pero sería ideal colocar primero el atributo object-class para mejorar la comprensión del fichero.

En un archivo LDIF puede existir más de una entrada definida. Cada entrada se separa de las demás por una línea en blanco y a su vez, en cada entrada puede haber una cantidad arbitraria de los pares nombre\_atributo → valor. Sería conveniente realizar copias de seguridad para que en caso de cualquier fallo o introducción de datos erróneos se pueda regresar al punto de partida.

A continuación, se ponen como ejemplo dos ficheros LDIF (extensión .ldif) para que se observe la forma de construir este tipo de ficheros:

- ✓ Crea una unidad organizativa llamada “empleados” dentro del dominio prueba.com:

```
dn:ou=empleados,dc=prueba,dc=com
objectClass:organizationalUnit
ou:empleados
```

- ✓ Crea un empleado de la organización con sus atributos dentro de la OU anterior:

```
dn:uid=fernando,ou=empleados,dc=prueba,dc=com
objectClass:inetOrgPerson
uid: fernando
ou:empleados
cn:Fernando
sn: vazquez
title:Gerente
userPassword:{MD5}1eb3691f6a502ca3065f9702b75b6a34
roomnumber:Room1
mail:fvazquez@prueba.com
```

www

### Recurso web

Consulta la página web Open LDAP para ampliar la información sobre LDAP y usar cualquier comando del servicio que se instalará a continuación.



## 1.9. Instalación de OpenLDAP en SO Linux

La instalación que se va a realizar es el servidor OpenLDAP, llamado slapd, y sus utilidades, que es el servicio DNS para SO Linux bajo la versión de Debian y, más concretamente, de Kali Linux. La instalación se puede realizar en una máquina virtual que posee una imagen de este sistema operativo o en una máquina física. El procedimiento de instalación es el siguiente:

1. Para comenzar, habría que tener conexión a Internet y actualizados los repositorios de la máquina virtual o física en la que vayamos a instalar el servicio de directorio LDAP, además de poner una dirección IP fija para que no haya problemas de conexión en un momento posterior. Una vez realizadas todas las acciones anteriores (comentadas en el servicio de DNS) se está en disposición de ejecutar la siguiente sentencia. Y se obtendrá una pantalla como la de la figura 1.17.

```
# apt-get install slapd ldap-utils
```

```
root@dosboxes: # apt-get install slapd ldap-utils
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
Los paquetes indicados a continuación se instalaron de forma automática y ya no
son necesarios.
  dh-python gir1.2-clutter-gst-3.0 gir1.2-gtkclutter-1.0 gir1.2-mutter-1
  gir1.2-networkmanager-1.0 gir1.2-nmgtk-1.0 libbind9-160 libdns1102
  libical-1.2-19 libedata-cal-1.2-28 libisc169 libisccc160 libiscfg160
  liblwres160 libmozjs-52-0 libmutter-1-0 libnm-glib4 libnm-gtk0 libnm-util2
  libpango-perl libwacom-bin python3-asn1crypto
Utilice «apt autoremove» para eliminarlos.
Se instalarán los siguientes paquetes adicionales:
  aspell evince evince-common evolution-data-server
  evolution-data-server-common fontconfig-config gcc-10-base gdm3
  gir1.2-atspi-2.0 gir1.2-freedesktop gir1.2-gdesktopevents-3.0
  gir1.2-gdkpixbuf-2.0 gir1.2-gdm-1.0 gir1.2-glib-2.0 gir1.2-gnomedesktop-3.0
  gir1.2-mutter-5 gir1.2-nm-1.0 gir1.2-nma-1.0 gjs gnome-desktop3-data
  gnome-session gnome-session-bin gnome-session-common gnome-settings-daemon
  gnome-settings-daemon-common gnome-shell gnome-shell-common
  gnome-shell-extension-dashdock gnome-shell-extension-easyscreencast
  gnome-shell-extension-proxyswitcher gnome-shell-extensions gnome-sushi
```

**Figura 1.17**  
Instalación LDAP

La siguiente pantalla que aparecerá, si todo es correcto, es la siguiente:

```
root@dosboxes: # apt-listchanges
Archivo Editar Ver Buscar Terminal Ayuda
apt-listchanges: News
-----
openldap (2.4.49+dfsg-1) unstable; urgency=medium
  This release fixes an issue with how the slapo-ppolicy(5) overlay
  stores the pwdChangedTime attribute in the database. Existing
  incorrect records could cause slapd to crash if a database
  administrator uses the Relax control to modify pwdChangedTime.

  Users of the ppolicy overlay are recommended to reload (slapcat and
  slapadd) their database in order to fix existing data.

  Please see <https://www.openldap.org/its/?findid=9126> for more
  information.

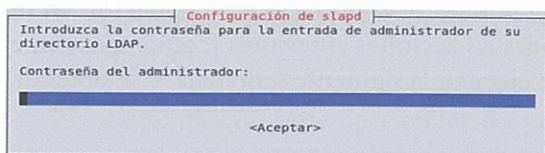
-- Ryan Tandy <ryan@nardis.ca> Mon, 03 Feb 2020 09:58:29 -0800
pulseaudio (11.1-2) unstable; urgency=medium
  * Since this version, pulseaudio disables autospawn by default on linux
    systems, and replaces that with systemd socket activation. If you are not
    using systemd, then please edit or remove
:
```

**Figura 1.18**  
Instalación LDAP continuación

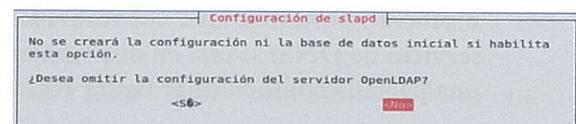
En esta pantalla se pulsará **q** para salir, ya que es meramente informativa. Posteriormente, el servicio pedirá una contraseña de administrador: se tecleará *admin* dos veces (la segunda vez de confirmación), aunque se puede teclear la que se crea oportuna. Se observa una imagen como la de la figura 1.19.

2. A continuación, se va configurar *slapd* mediante el siguiente comando. Se visualizará la ventana de la figura 1.20, donde se dejará seleccionada la opción por defecto (No):

```
# dpkg-configure
```

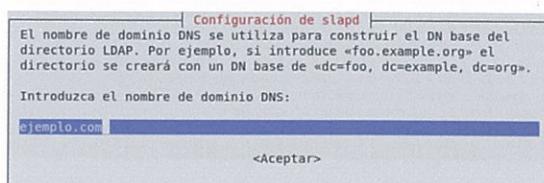


**Figura 1.19**  
Instalación LDAP configuración contraseña del administrador

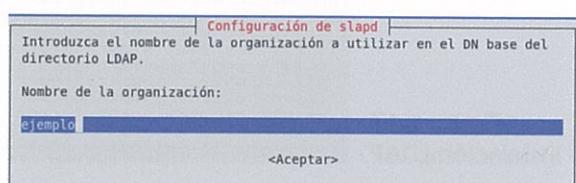


**Figura 1.20**  
Instalación LDAP configuración slapd

Cuando se pulse esta opción, saldrá otra pantalla donde habrá que teclear el dominio que sea necesario instalar en el servicio de directorio, en nuestro caso `ejemplo.com` (figura 1.21). A continuación, se pondrá el nombre de la organización, que será `ejemplo`, siguiendo la nomenclatura anterior (figura 1.22):



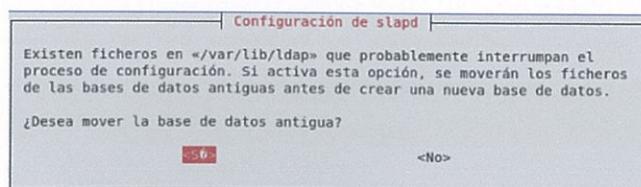
**Figura 1.21**  
Instalación LDAP configuración dominio



**Figura 1.22**  
Instalación LDAP configuración nombre de la organización

El siguiente paso sería teclear de nuevo la contraseña de administrador dos veces, que, en nuestro caso, sería *admin*. Por último, pregunta la configuración si se quiere purgar la base de datos; la respuesta más adecuada es que sí.

Posteriormente, pregunta si se quiere mover la base de datos de LDAP anterior; la respuesta es sí (figura 1.23).



**Figura 1.23**  
Instalación LDAP configuración base de datos

Ya se tiene instalado el servicio de directorio LDAP, por lo que lo único que nos queda es subir el servicio y bajar el servicio, y esto se realiza con los siguientes comandos:

```
# service slapd start
# service slapd stop
```

Una vez que levantemos el servicio, el servidor LDAP estará escuchando por el puerto 389 de TCP y UDP. Para conexiones seguras mediante SSL sería el puerto 636.

Otros comandos que nos permiten gestionar LDAP son los siguientes:

**# service slapd reload** → permite recargar la configuración del servicio sin reiniciarlo.

**# service slapd restart** → reinicia el servicio.

**# service slapd force-reload** → fuerza la recarga de la configuración del servicio.

**# service slapd status** → comprueba si el servicio está activo o inactivo.

3. Se van a detallar algunas herramientas del servidor LDAP. Para mostrar la información que tenemos se debe teclear el siguiente comando:

```
# slapcat
```

Y mostrará una salida, teniendo en cuenta nuestro dominio ejemplo.com, como la siguiente:

```
root@osboxes:~# service slapd start
root@osboxes:~# service slapd stop
root@osboxes:~# slapcat
dn: dc=ejemplo,dc=com
objectClass: top
objectClass: dcObject
objectClass: organization
o: ejemplo
dc: ejemplo
structuralObjectClass: organization
entryUUID: 007f5d6c-fcd5-1039-8861-b7464ed071fd
creatorsName: cn=admin,dc=ejemplo,dc=com
createTimestamp: 20200317195531Z
entryCSN: 20200317195531.408752Z#000000#000#000000
modifiersName: cn=admin,dc=ejemplo,dc=com
modifyTimestamp: 20200317195531Z

dn: cn=admin,dc=ejemplo,dc=com
objectClass: simpleSecurityObject
objectClass: organizationalRole
cn: admin
description: LDAP administrator
userPassword:: eINTSEF9Ui9LMmp2TWFUZ3dZ0HpYeWtj0WdyblZ4SDZF0dla0s=
structuralObjectClass: organizationalRole
entryUUID: 007f9c8c-fcd5-1039-8862-b7464ed071fd
creatorsName: cn=admin,dc=ejemplo,dc=com
createTimestamp: 20200317195531Z
entryCSN: 20200317195531.410409Z#000000#000#000000
modifiersName: cn=admin,dc=ejemplo,dc=com
modifyTimestamp: 20200317195531Z
```

**Figura 1.24**  
Instalación LDAP  
iniciar, parar y mostrar  
información

**#slapdd:** permite insertar entradas desde un fichero LDIF.

**#slapindex:** regenera los índices.

**#slappasswd:** sirve para obtener una contraseña encriptada.

Por último, veremos algunos comandos para agregar información al servicio de directorio LDAP desde el punto de vista del cliente, como son los siguientes:

- *ldapsearch*: para seleccionar los objetos de un directorio a partir de cierta raíz del árbol de directorio. Un ejemplo de sentencia sería la siguiente (buscaría todos los objetos dentro de nuestro dominio ejemplo.com):

```
ldapsearch -b "dc=ejemplo,dc=com" "objectclass=*"
```

- *ldapmodify*: para modificar los objetos de un directorio a partir de cierta raíz del servicio de directorio. Un ejemplo sería:

```
ldapmodify -D cn=admin,dc=ejemplo,dc=com -w admin -f Modify.ldif -c
```

La línea anterior permite conectarnos como administrador (admin) a nuestro dominio ejemplo.com y modificar en el servicio de directorio la información contenida en el fichero Modify1.ldif. La opción *-c* permite reportar los errores.

- *ldapadd*: para añadir objetos a nuestro servicio de directorio.

```
ldapadd -D cn=admin,dc=ejemplo,dc=com -w admin -f Add.ldif -c
```

La línea anterior permite conectarnos como administrador (admin) a nuestro dominio ejemplo.com e introducir en el servicio de directorio el fichero Add.ldif. La opción *-c* permite reportar los errores.

- *ldapdelete*: para borrar entradas de nuestro servicio de directorio.

```
ldapdelete -D "cn=root,dc=ejemplo,dc=com"
>cn=Juan Muñoz, ou=usuarios, dc=ejemplo,dc=com
```



#### PARA SABER MÁS

En la página web de Oracle se puede profundizar más en estos comandos.



#### Actividad propuesta 1.5



¿Cuántos comandos existen para manipular el directorio LDAP? Haz pruebas con cada uno de ellos.

## 1.10. Adaptación de la configuración del servidor de directorios para el despliegue de la aplicación. Usuarios centralizados

Una vez que se ha realizado la instalación de LDAP en un sistema operativo Linux se va a proceder a la autenticación por parte de una aplicación o del sistema operativo en el servicio de directorio. La autenticación de LDAP es una de las funciones más importantes, ya sea en el FTP o en páginas web o en cualquier otro servicio.

Hasta ahora el sistema operativo Linux autentica mediante los clásicos archivos /etc/passwd, /etc/group y /etc/shadow. Pero se puede autenticar a partir de otros métodos como pueden ser NIS, LDAP, WINS, etc.

### 1.10.1. Autenticación en el servicio de directorio

Se va a proceder a instalar en la máquina DAW las librerías oportunas para autenticar a los usuarios mediante LDAP. Los datos de LDAP instalados previamente son los siguientes:

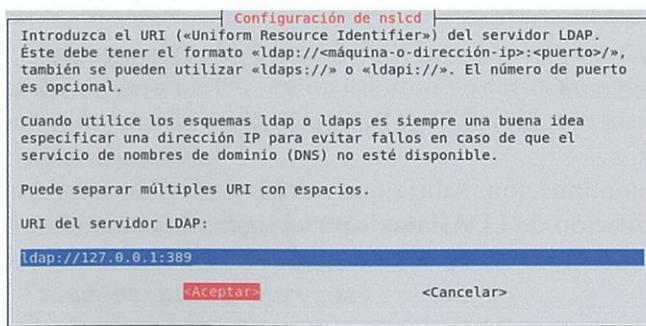
- Dirección: ldap://127.0.0.1:389
- Base: dc=cursolinux,dc=com
- Usuario: rrhh
- Password: cat123

Se pueden crear todos los usuarios que queramos e introducirlos en el directorio LDAP, y con ello permitirá la validación. Para esto se realiza el siguiente procedimiento:

1. Instalación y configuración de los paquetes libnss-ldapd y libpam-ldapd:

```
#apt-get install libpam-ldapd libnss-ldapd
```

2. Después de realizar una serie de operaciones, se visualizará una pantalla como la de la figura 1.25.

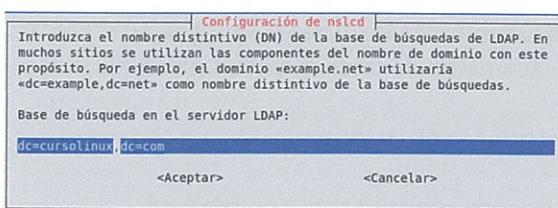


**Figura 1.25**  
Configuración nsLCD

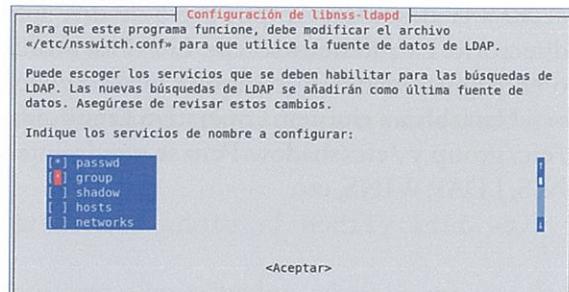
3. Una vez completado el campo de LDAP con la IP del servidor de LDAP (nuestro caso 127.0.0.1 y el puerto 389), es necesario completar el dominio por el cual accedemos al

servicio de directorio (en nuestro caso dc=cursolinux, dc=com), que se podrá observar en la figura 1.26.

4. El siguiente paso es elegir los servicios que se van a usar para la conexión (en nuestro caso sería password y group). Como se puede observar en la figura 1.27.



**Figura 1.26**  
Configuración nsLCD



**Figura 1.27**  
Configuración versión libnss-ldapd

5. Si por error se ha introducido algún parámetro erróneo, siempre se tiene la posibilidad de modificar los ficheros que intervienen en la configuración, que son nsLCD.conf y nsswitch.conf. Lo importante en el fichero nsLCD.conf es la localización del servidor LDAP. En el fichero nsswitch.conf se permite habilitar las opciones de group y passwd. Se puede observar en las siguientes figuras:

```
GNU nano 2.8.7          Fichero: nsLCD.conf
# /etc/nsLCD.conf
# nsLCD configuration file. See nsLCD.conf(5)
# for details.

# The user and group nsLCD should run as.
uid nsLCD
gid nsLCD

# The location at which the LDAP server(s) should be reachable.
uri ldap://127.0.0.1:389

# The search base that will be used for all queries.
base dc=cursolinux,dc=com

# The LDAP protocol version to use.
#ldap_version 3
```

**Figura 1.28**  
Fichero nsLCD.conf

```
GNU nano 2.8.7          Fichero: nsswitch.conf
# /etc/nsswitch.conf
#
# Example configuration of GNU Name Service Switch functionality.
# If you have the 'glibc-doc-reference' and 'info' packages installed, try
# 'info libc "Name Service Switch"' for information about this file.

passwd: compat ldap
group: compat ldap
shadow: compat ldap
gshadow: files

hosts: files mdns4_minimal [NOTFOUND=return] dns myhostname
networks: files

protocols: db files
services: db files
ethers: db files
rpc: db files
```

**Figura 1.29**  
Fichero nsswitch.conf

6. A continuación, habría que iniciar el servicio nsLCD para poner en funcionamiento la validación de LDAP mediante el siguiente comando:

```
#service nsLCD restart
```

7. Por último, se probaría si se ha configurado correctamente la validación mediante el comando login y debería validarse el usuario que al principio se comentó (usuario:rrhh y password:cat123), como se observa en la figura 1.30.

```

root@osboxes:~# service nslcd restart
root@osboxes:~# login
osboxes nombre: rrhh
Contraseña:
Último inicio de sesión: vie jun  5 14:47:54 EDT 2020en pts/0

The programs included with the Kali GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/*copyright.

Kali GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
rrhh@osboxes:~$ mkdir prueba

```

**Figura 1.30**  
Validación LDAP

### 1.10.2. Usuarios centralizados

Los usuarios centralizados es una forma de agrupar los usuarios en un servicio de directorio o base de datos común para realizar una sola autenticación.

Para comprobar esta autenticación se ha creado otra máquina que se va conectar, denominada DAW2, con los mismos paquetes anteriormente instalados en la máquina DAW. En esta prueba se va a usar NFS, que permitirá a un usuario acceder a otra computadora como si fuera local. Antes de comenzar el procedimiento, se comprueba que tenemos acceso a la máquina mediante ping y, posteriormente, en el fichero nslcd.conf es necesario poner la IP de la máquina DAW (en nuestro caso, 10.0.2.15).

Se comprueba que funciona el login desde la máquina DAW2. Como se muestra en la figura 1.31:

```

root@osboxes:~# login
osboxes nombre: rrhh
Contraseña:

The programs included with the Kali GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/*copyright.

Kali GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Sin directorio, accediendo con HOME=/

```

**Figura 1.31**  
Validación desde máquina remota

El siguiente paso es montar NFS mediante validación LDAP. Esto se podría hacer con cualquier servicio como por ejemplo FTP (abordaremos en un capítulo posterior) u otra aplicación.

Comenzamos el procedimiento instalando en la máquina DAW (servidor principal) NFS y sus utilidades mediante los siguientes comandos:

```
#apt-get install nfs-kernel-server
#apt-get install nfs-common
```

1. Una vez instalado el paquete correspondiente al servicio NFS, se crea el directorio nfs dentro de /home y, a su vez, el directorio rrhh. Se le dan permisos al usuario rrhh (figura 1.32).

El comando chown permite dar los permisos al directorio rrhh. Los dos atributos separados por los dos puntos son respectivamente el atributo uid number y guid number, que deberán ser los mismos que están declarados en el directorio LDAP. Para que se pueda entrar en el directorio /home/nfs/rrhh.

```
root@osboxes:/home# mkdir nfs
root@osboxes:/home# cd nfs
root@osboxes:/home/nfs# ls
root@osboxes:/home/nfs# mkdir rrhh
root@osboxes:/home/nfs# ls
rrhh
root@osboxes:/home/nfs# chown 2000:2000 rrhh
root@osboxes:/home/nfs# ls -l
total 4
drwxr-xr-x 2 2000 2000 4096 jun  6 05:01 rrhh
root@osboxes:/home/nfs#
```

**Figura 1.32**  
Creación de carpetas

2. Posteriormente, es necesario modificar el fichero /etc/exports, que permite a los clientes poder escribir en el sistema de ficheros. Para ello es imprescindible añadir una línea como la de la figura 1.33.

```
GNU nano 2.8.7          Fichero: /etc/exports
#
# /etc/exports: the access control list for filesystems which may be exported
#               to NFS clients. See exports(5).
#
# Example for NFSv2 and NFSv3:
# /srv/homes    hostnamel(rw,sync,no_subtree_check) hostname2(ro,sync,no_sub$#
#
# Example for NFSv4:
# /srv/nfs4      gss/krb5i(rw,sync,fsid=0,crossmnt,no_subtree_check)
# /srv/nfs4/homes gss/krb5i(rw,sync,no_subtree_check)
#
/home/nfs  *(rw,sync,fsid=0,subtree_check)
```

**Figura 1.33**  
Fichero exports

#### CUADRO 1.4 Opciones de exports

Comando	Descripción
/home/nfs	Es el directorio que compartir por NFS.
*	Las IP que pueden acceder al directorio, en nuestro caso, todo el mundo.
rw	Se permite leer y escribir en el directorio.
sync	El servicio responde a las solicitudes una vez que se han producido los cambios en el sistema de ficheros.
fsid=0	NFS necesita identificar el punto de montaje; se le asigna el valor 0.
subtree_check	Chequea por seguridad los subdirectorios.

**Recurso web****www**

Se recomienda consultar la web [linux.die.net](http://linux.die.net) con interesante documentación para ampliar la información sobre la exportación de directorios mediante nfs.



3. El siguiente paso es reiniciar el servicio NFS para que se configuren los cambios realizados en el fichero /etc/exports, mediante el siguiente comando:

```
#service nfs-kernel-server restart
```

4. Una vez reiniciado el servicio NFS es el momento de probar si funciona correctamente la validación del usuario rrhh y la ubicación del home. Esta información se lee del directorio LDAP configurado previamente. Se realiza la comprobación en la figura 1.34.

```
root@osboxes:~# login
osboxes nombre: rrhh
Contraseña:
Último inicio de sesión:sáb jun  6 10:55:09 EDT 2020en pts/0

The programs included with the Kali GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*copyright.

Kali GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
rrhh@osboxes:~$ pwd
/home/nfs/rrhh
rrhh@osboxes:~$
```

**Figura 1.34**  
Comprobar validación

**PARA SABER MÁS**

Es necesario configurar el atributo loginShell con /bin/bash de la cuenta de rrhh del directorio LDAP, si no, no se podrá comprobar que el usuario inicia sesión y mapea el directorio home configurado.

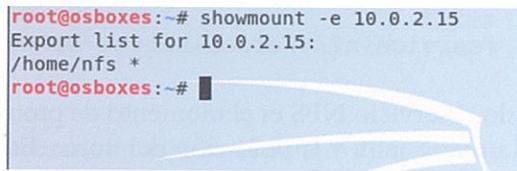
5. Por último, se comprobará desde la máquina DAW2 que hace de cliente la conexión a DAW con usuario rrhh (directorio LDAP) y mediante NFS. Para ello se instalarán los mismos paquetes que se instalaron en DAW y se configurará el fichero /etc/ldap/ldap.conf de la máquina cliente (DAW2):

```
BASE dc=cursolinux,dc=com
URI ldap://10.0.2.15 (ip del servidor LDAP, DAW)
```

**RECUERDA**

- ✓ Las dos máquinas (DAW y DAW2) se tienen que ver. Esto se realiza con el comando ping y la IP que tenga configurada cada máquina.

6. La siguiente fase, desde el cliente, es comprobar que el servidor tiene montado el directorio que se va a mapear, en nuestro caso /home/nfs. Para ello usamos el comando de la figura 1.35.



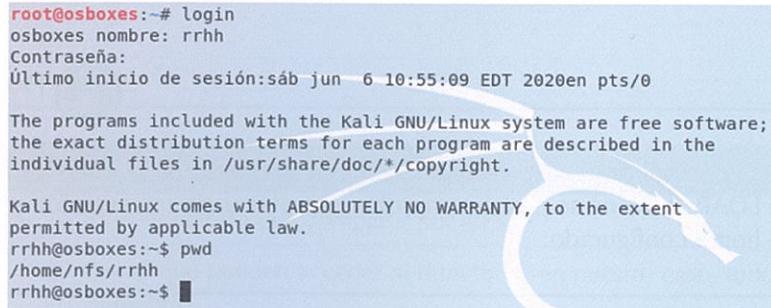
```
root@osboxes:~# showmount -e 10.0.2.15
Export list for 10.0.2.15:
/home/nfs *
```

**Figura 1.35**  
Comprobar montaje

7. Si ahora se valida directamente el usuario tendría como home la raíz del sistema operativo. Para solucionar este inconveniente es necesario montar el sistema de ficheros desde el cliente (DAW2), con el siguiente comando:

```
#mount -t nfs 10.0.2.15:/home/nfs/ /home/nfs/
```

8. Es el momento de comprobar la validación desde el cliente DAW2. El home se ubicará en la ruta /home/nfs/rrhh (figura 1.36).



```
root@osboxes:~# login
osboxes nombre: rrhh
Contraseña:
Último inicio de sesión:sáb jun  6 10:55:09 EDT 2020en pts/0

The programs included with the Kali GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*copyright.

Kali GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
rrhh@osboxes:~$ pwd
/home/nfs/rrhh
rrhh@osboxes:~$
```

**Figura 1.36**  
Comprobación de la validación desde el cliente

9. Si se necesita que el montaje se quede de forma definitiva, es decir, una vez que las máquinas reinician, es necesario introducir en el fichero /etc/fstab de la máquina DAW2 (cliente) la siguiente línea:

```
10.0.2.15:/home/nfs/ /home/nfs/ nfsrw 0 0
```

## Resumen

- En este capítulo inicial se han explicado detalladamente los servicios de red principales que intervienen en el despliegue de una aplicación, como son el servicio de DNS y el servicio de directorio LDAP.
- El servicio de DNS es el encargado de traducir los nombres de dominio en direcciones IP, y a la inversa, acción que es clave en el mundo tecnológico en que vivimos, de lo contrario, habría que recordar todas las IP a las que accedemos.
- Los dos componentes básicos del servicio DNS son las zonas directa e inversa, que permiten resolver los registros que definimos en las mismas. Así como el fichero de resolución de DNS resolv.conf.
- Existen tres tipos de servidores DNS, primarios, secundarios y locales o caché. Dependiendo de las necesidades de la organización instalaremos un tipo u otro. Y, por otro lado, están los tipos de registros que podemos definir en estos tipos de servidores.
- Otro punto clave es conocer el funcionamiento de DNS mediante las consultas iterativas y recursivas.
- Por último, con respecto al servicio de DNS, se detalla la instalación y configuración en el SO Linux, y sus respectivos comandos de manejo.
- El servicio de directorio es una base de datos preparada para almacenar información que normalmente necesita cualquier aplicación que despleguemos en una organización. Así como las funciones de gestión, acceso y seguridad de la información.
- El servicio de directorio tratado en este capítulo es LDAP, que puede instalarse en dos modalidades: centralizado y distribuido. Lo ideal sería una mezcla de ambas.
- Lo fundamental en este servicio es la integración con otros servicios para dar valor a este protocolo.
- LDAP tiene como formato de intercambio de datos a LDIF, que se basa en una estructura en dos partes: por un lado DN y por otro el conjunto de atributos que lo componen.
- Para terminar, se explica detalladamente la instalación y configuración de LDAP (slapd) en SO Linux, con sus respectivos comandos para añadir información al servidor de directorio.

## Supuestos prácticos

1. Obtén la dirección IP de los siguientes dominios:
  - a) www.debian.org
  - b) www.aemet.es
  - c) www.hostalia.com
  - d) es.wikipedia.org
  - e) www.iestrass Sierra.com
2. ¿Qué tipo de registro es el que resuelve las direcciones anteriores?
3. Comprueba la dirección IP y el servidor DNS de la dirección web de tu trabajo o donde estés estudiando, hazlo desde Internet y desde la Intranet de la dirección web. ¿Cuáles son las diferencias? ¿A qué crees que se debe?

4. Un BIND simple no está cifrado, para cifrarlo hay que usar el protocolo.....
5. La empresa ABX desea crear un árbol LDAP. Para ello, el personal encargado ha creado el dominio DNS “nex. ejemplo.com.”. En cualquier operación sobre los objetos del árbol LDAP habrá que referirse al dominio DNS indicado siempre en este orden y de la siguiente manera: dc=....., dc=....., dc=.....



## Ejercicios propuestos

1. Instala y configura de un servidor DNS en el SO Linux Debian. El servidor DNS será bind9 y será necesario configurarlo y comprobar que funciona correctamente. Para ello será necesario configurar como IP fija en el fichero interfaces y también el fichero named.conf.options donde se configurarán las búsquedas directas e inversas. Por último, habrá que configurar el fichero resolv.conf para indicar que el servidor es el mismo y ya estamos preparados para comprobar que resuelve nombres e IP mediante el comando nslookup, host o dig. El dominio sería ejemplo.com y es necesario definir un par de registros en zona directa e inversa.
2. Instala el servidor OpenLdap, llamado slapd (apt-get install slapd ldap-utils). El directorio raíz será ptec.com, la password del administrador será admin. Hay que crear un fichero .ldif que contendrá una unidad organizativa llamada “empleados”. Posteriormente, hay que añadir a esta unidad organizativa el nombre de un empleado con sus características, para ello se pondrá el nombre “alumno”. Es necesario crear los ficheros .ldif, que contendrán la información del directorio activo. Así como el fichero script.sh, que permite añadir los ficheros .ldif al directorio activo.
3. Instala y configura un servidor DNS en el sistema operativo Windows 2019 Server, y crea una zona inversa y otra zona directa. El dominio puede ser empresa.com, hay que crear un registro en zona directa y otro en zona inversa. Comprueba mediante un cliente de Windows que el servidor DNS funciona correctamente.
4. Instala y configura de un servidor LDAP en el sistema operativo Windows 10 y crea una unidad organizativa llamada “empleados”. Posteriormente, añade un empleado (nombre del alumno) con sus características principales. Accede a ellas y muestra por pantalla tal información.

## ACTIVIDADES DE AUTOEVALUACIÓN

1. ¿En qué consiste el servidor DNS?:
  - a) Dar nombres a las páginas web.
  - b) Interpretar la información.
  - c) Traducir nombres de dominio a direcciones IP.
  - d) Ninguna de las opciones anteriores es correcta.

2. ¿Qué zonas de búsqueda tiene el DNS?:

- a) Zona de búsqueda directa e inversa.
- b) Zona de búsqueda proporcional e inversa.
- c) Zona de búsqueda indirecta e inversa.
- d) Zona de búsqueda directa y proporcional.

3. ¿Cuáles son tipos de servidores DNS?:

- a) Servidores primarios.
- b) Servidores secundarios.
- c) Servidores solo caché.
- d) Todas las opciones son válidas.

4. ¿Cuáles son tipos de registros DNS?:

- a) AA.
- b) NSX.
- c) MXT.
- d) Ninguna de las opciones es válida.

5. ¿Qué es una consulta inversa?:

- a) Es el proceso de resolver una IP a partir de un nombre.
- b) Es el proceso de resolver un nombre a partir de otro nombre de host.
- c) Es el proceso de resolver un nombre a partir de una IP.
- d) Ninguna de las opciones anteriores es correcta.

6. ¿Cuál es el formato de intercambio de datos en LDAP?:

- a) CSV.
- b) Texto.
- c) LDIF.
- d) LDIV.

7. ¿Cuál es el puerto por defecto que escucha el servidor LDAP?:

- a) 399.
- b) 389.
- c) 489.
- d) 488.

8. ¿Qué estructura tiene el directorio LDAP?:

- a) Forma de árbol DIT.
- b) Forma de árbol DN.
- c) Forma de árbol SN.
- d) Forma de árbol DTI.

9. ¿En qué estándar está basado LDAP?:

- a) DIT.
- b) DN.
- c) X.500.
- d) DTI.

10. ¿Cuál es el puerto por defecto que escucha el servidor LDAP de forma segura?:

- a) 736.
- b) 389.
- c) 636.
- d) 336.

**SOLUCIONES:**

1. **a** **b** **c** **d**

2. **a** **b** **c** **d**

3. **a** **b** **c** **d**

4. **a** **b** **c** **d**

5. **a** **b** **c** **d**

6. **a** **b** **c** **d**

7. **a** **b** **c** **d**

8. **a** **b** **c** **d**

9. **a** **b** **c** **d**

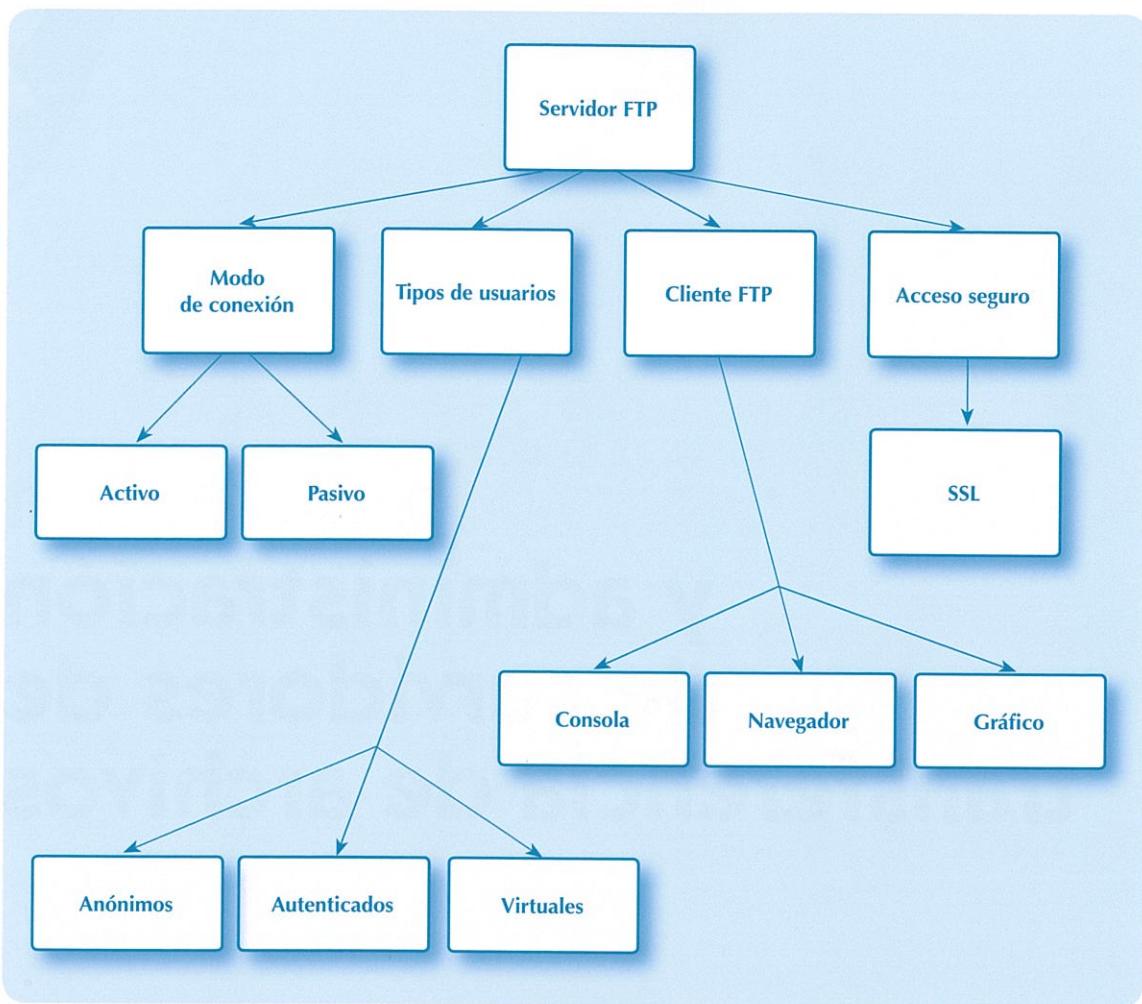
10. **a** **b** **c** **d**

# Instalación y administración de servidores de transferencia de archivos

## Objetivos

- ✓ Instalar y configurar servidores de transferencia de archivos.
- ✓ Crear usuarios y grupos para el acceso remoto al servidor.
- ✓ Configurar el acceso anónimo.
- ✓ Comprobar el acceso al servidor, tanto en modo activo como en modo pasivo.
- ✓ Usar el protocolo seguro de transferencia de archivos.
- ✓ Configurar y utilizar servicios de transferencia de archivos integrados en servidores web.
- ✓ Emplear el navegador como cliente del servicio de transferencia de archivos.

## Mapa conceptual



### Glosario

**Cortafuegos.** Es un componente software o hardware que permite bloquear el acceso no autorizado y permitir las comunicaciones deseadas en nuestra organización.

**FTP.** Es el protocolo de aplicación de transferencia de ficheros que intercambia información entre dos dispositivos conectados a una red TCP.

**Internet.** Es una red de redes que permite la interconexión de dispositivos mediante el conjunto de protocolos TCP/IP.

**Paquetes Linux.** Son programas que se pueden instalar mediante el comando apt-get en las distribuciones Linux y que forman parte del repositorio que posea cada distribución.

**Puerto de conexión UDP o TCP.** Es un número que va desde el 0 hasta el 65535 y sirve para enviar y recibir datos mediante el socket.

**Socket.** Es un ente abstracto a bajo nivel que permite la comunicación y el intercambio de datos entre un cliente y un servidor. Los sockets se pueden programar. Un socket se identifica mediante una IP y un puerto.

**TCP.** Es el protocolo orientado a conexión dentro del modelo TCP/IP que se encarga de crear las conexiones entre dos dispositivos mediante puertos. Es fundamental dentro del funcionamiento de Internet.

**UDP.** Es el protocolo no orientado a conexión en redes IP y a nivel de transporte. Se encarga de la trasmisión de información sin conexión de datagramas y mediante puertos.

## 2.1. Introducción

El servicio de transferencia de ficheros tiene un papel fundamental a la hora de desplegar una aplicación. Su función es transferir la información de desarrollo a producción en un entorno empresarial.

En el mundo tecnológico existen distintos servidores FTP. Para poder trabajar con ellos, se ha elegido la instalación y configuración de proFTPD, por ser uno de los más completos.

Existen varios modos de conexión, como son el activo y el pasivo, que van a depender de si existe un cortafuegos en mitad de la conexión o no.

Por otro lado, existen tres tipos de usuarios que se pueden habilitar en este tipo de servicio, como son usuarios autenticados, virtuales y anónimos.

## 2.2. Servicio de transferencia de archivos. Permisos y cuotas

Es un protocolo de red TCP que permite la transferencia de archivos entre sistemas conectados entre sí. Se basa en la arquitectura cliente-servidor del RFC 959. De manera que desde un cliente existe la posibilidad de conectar a un servidor para descargar archivos desde él o para enviarle nuestros propios archivos, independientemente del sistema operativo. El sistema operativo instalado y configurado en cada equipo es multiplataforma y heterogéneo.

La transferencia de archivos puede ser de todo tipo, como imágenes, vídeos, texto, etc., entre el cliente y el servidor. El interfaz de transferencia puede ser mediante comandos o modo gráfico. Aunque es uno de los métodos más usados en Internet, tiene un gran inconveniente, que es la seguridad, si no se configura correctamente y se toman las medidas oportunas para evitar el acceso fraudulento de la información que se transfiere.

Como se ha comentado anteriormente, este servicio se basa en una arquitectura cliente-servidor, siendo el cliente quien solicita la conexión para transferir los archivos y el servidor es el que ofrece o almacena archivos dependiendo de la solicitud del cliente. Por lo tanto, es

un servicio orientado a conexión que necesita establecer una conexión para poder transferir archivos.

Es necesario usar este servicio acompañado de algún protocolo de seguridad, como puede ser SSL, ya que de no hacerlo se corre el riesgo de que la información transferida sea hackeada por entidades externas.

El servidor de FTP funciona a través de los siguientes puertos configurables:

- Puerto 21: control de la conexión.
- Puerto 20 o mayor de 2014: puerto de transferencia de datos.

Hay que tener en cuenta que estos puertos son modificables mediante los archivos de configuración correspondientes. Pero si no se configura nada, por defecto, los puertos anteriores son utilizados por la arquitectura.

### 2.2.1. Permisos

Los permisos y las cuotas son una parte importante de la configuración del servicio FTP. Es necesario controlar el espacio y los permisos de lectura y escritura a usuarios que entran en el sistema desde el exterior. De no hacerlo así, podría aumentar el riesgo de amenaza desde el exterior y convertirse en riesgo real. Las amenazas podrían ser desde entrar en otro directorio que no sea el dedicado para el FTP, hasta que se caiga el sistema por falta de espacio.

Antes de entrar en detalle con las cuotas, se va a proceder a detallar cómo funcionan los permisos en Linux, para posteriormente explicar cómo a un usuario se le asignan los permisos.

Cuando se crea un fichero o carpeta en Linux, existen tres niveles de acceso que permiten controlar sus accesos, que son los siguientes:

- a) *Nivel propietario*: son los permisos que se asignan al propietario del archivo o directorio.
- b) *Nivel grupo*: son aquellos que se asignan a los grupos de usuarios. Esto es, un grupo puede tener de 1 a n usuarios.
- c) *Nivel usuarios*: este nivel corresponde a todos los usuarios definidos en el sistema operativo que no son los anteriores, o llamados “los otros”.



#### TOMA NOTA

El servicio de FTP trabaja con puertos: 21 de control y 20 de transferencia de datos.

Los permisos en Linux son tres y se distinguen de la siguiente forma:

1. *Lectura (r)*: el usuario podrá ver el contenido y visualizar un fichero o directorio. Si tiene asignado (-) no podrá visualizarlo.
2. *Escritura (w)*: el usuario podrá modificar el contenido del archivo o directorio.
3. *Ejecución (x)*: el usuario podrá ejecutar el archivo.

Normalmente son aplicados estos permisos para archivos ejecutables.

Y se le pueden aplicar a cada uno de los niveles anteriores. Por ejemplo, cuando nosotros creamos un archivo o directorio con el usuario root, y listamos el directorio con el comando ls -l, saldría algo parecido a la pantalla de la figura 2.1.

```
root@osboxes:~# ls -l
total 40
drwxr-xr-x 2 root root 4096 mar 16 12:53 Desktop
drwxr-xr-x 2 root root 4096 nov 23 2017 Documents
drwxr-xr-x 2 root root 4096 nov 23 2017 Downloads
drwxr-xr-x 2 root root 4096 nov 23 2017 Music
drwxr-xr-x 2 root root 4096 mar 29 14:17 Pictures
drwxr-xr-x 2 root root 4096 abr  3 05:11 prueba
-rw-r--r-- 1 root root  20 mar 29 15:36 prueba.txt
drwxr-xr-x 2 root root 4096 nov 23 2017 Public
drwxr-xr-x 2 root root 4096 nov 23 2017 Templates
drwxr-xr-x 2 root root 4096 nov 23 2017 Videos
```

**Figura 2.1**  
Listado de ficheros



PARA SABER MÁS

El comando ls -l permite conocer los permisos que tiene cada fichero o directorio. Investiga y prueba todas las opciones que permite el comando ls.

Se va a explicar cada una de las partes que acompañan a cada fichero o directorio:

- El primer carácter identifica a los siguientes tipos de ficheros:
  - (d): es un directorio.
  - (-): es un fichero.
  - (l): representa un enlace (link).
  - (b): indica que es un archivo binario.
  - (p): es un archivo especial de cauce (tubería).
  - (c): es un archivo de caracteres especiales, como puede ser una impresora.
- Despues del primer carácter le siguen rwxr-xr-x, que son los permisos correspondientes al propietario del directorio o archivo en sus primeros tres caracteres, los tres siguientes son los correspondientes al grupo y los últimos tres caracteres están relacionados con los demás usuarios del sistema operativo.
- Despues de los caracteres anteriores, aparece un número que indica el número de enlaces al archivo.
  - El primer root corresponde al usuario propietario del archivo o directorio.
  - El segundo root corresponde al grupo al que pertenece el archivo.
  - Las siguientes columnas representan el tamaño, fecha y hora de la última modificación del archivo o directorio.
  - La última columna es el nombre del directorio o archivo.

## Actividad propuesta 2.1



Explica cada uno de los campos que acompañan al directorio o fichero del sistema operativo.

Para asignar permisos en Linux se usan los siguientes comandos:

1. *chmod*: este comando puede modificar el permiso del propietario (u), los grupos (g) y los otros (o). Existen multitud de opciones para usar este comando, incluso se puede usar el sistema octal para aplicar permisos. La sintaxis general del comando es la siguiente:

```
chmod [opciones] modo-octal fichero.
```

El modo octal relacionado con los permisos aplicados a las tres columnas sería el siguiente:

Número decimal	Binario	Permisos efectivos
0	000	---
1	001	--x
2	010	-w-
3	011	-wx
4	100	r--
5	101	r-x
6	110	rw-
7	111	rwx

Por ejemplo, si se quiere asignar permisos de lectura (r) y escritura (w) al fichero prueba.txt al usuario propietario solo sería de la forma:

```
chmod 600 prueba.txt o  
chmod u+r w prueba.txt
```

2. *chown*: permite cambiar el propietario del archivo o directorio. La estructura general del comando sería la siguiente:

```
chown [opciones] [usuario] [:grupo] ficheros
```

Por ejemplo, si se quiere hacer propietario a Javier del fichero prueba.txt sería de la siguiente forma:

```
chown javier prueba.txt
```



## Actividad propuesta 2.2

Crea un fichero de texto y le das permiso de lectura y escritura al propietario, grupo y otros.

### 2.2.2. Cuotas

A continuación, se va a demostrar cómo funcionan las cuotas, y para ello se instalará el servicio *quota* en el sistema operativo, además de realizar algunas configuraciones para que funcione el programa:

- Instalación del programa *quota*, como se observa en la figura 2.2.

```
root@osboxes:~# apt-get install quota
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
Se instalarán los siguientes paquetes adicionales:
  libtirpc-common libtirpc3
Paquetes sugeridos:
  libnet-ldap-perl rpcbind
Se instalarán los siguientes paquetes NUEVOS:
  libtirpc-common libtirpc3 quota
0 actualizados, 3 nuevos se instalarán, 0 para eliminar y 1981 no actualizados.
```

**Figura 2.2**

Instalación de *quota*

- El siguiente paso es configurar el fichero */etc/fstab* con las opciones *usrquota* y *grpquota*. Se pueden aplicar a todo el sistema de ficheros, pero lo más recomendable es donde se ubican los usuarios, que es el directorio */home*. Se puede observar en los recuadros de la figura 2.3.

```
GNU nano 2.8.7          Fichero: /etc/fstab
#
# /etc/fstab: static file system information.
#
# Use 'blkid' to print the universally unique identifier for a
# device; this may be used with UUID= as a more robust way to name devices
# that works even if disks are added and removed. See fstab(5).
#
# <file system> <mount point> <type> <options>      <dump>  <pass>
# / was on /dev/sdal during installation
UUID=51373f23-ee37-4d51-97ad-8ec04b57b53f /          ext4    errors=remount-ro  1
# /boot was on /dev/sda2 during installation
UUID=3adff58e1-1f88-4d7f-ae59-aebb2002d3bc /boot      ext4    defaults        0      2
# /home was on /dev/sda4 during installation
UUID=4a94062c-d003-4b6a-9038-9283dd21f88a /home        ext4    defaults,usrquota,grpquota  0
# swap was on /dev/sda3 during installation
UUID=3c19fc72-61ef-4836-96ae-8160de1b5acd none        swap    sw            0      0
/dev/sr0           /media/cdrom0 udf,iso9660 user,noauto  0        0
```

**Figura 2.3**

Configuración de *fstab*

- Para que se apliquen los cambios, se han de ejecutar los siguientes comandos. Y se observarán las figuras 2.4 y 2.5.

```
# mount -o remount /home
# mount
```

```
root@osboxes:~# mount -o remount /home
root@osboxes:~# mount
sysfs on /sys type sysfs (rw,nosuid,nodev,noexec,relatime)
proc on /proc type proc (rw,nosuid,nodev,noexec,relatime)
udev on /dev type devtmpfs (rw,nosuid,relatime,size=1010156k,nr_inodes=252539,mode=755)
devpts on /dev/pts type devpts (rw,nosuid,noexec,relatime,gid=5,mode=620,ptmxmode=000)
tmpfs on /run type tmpfs (rw,nosuid,noexec,relatime,size=205208k,mode=755)
/dev/sdal on / type ext4 (rw,relatime,errors=remount-ro,data=ordered)
securityfs on /sys/kernel/security type securityfs (rw,nosuid,nodev,noexec,relatime)
tmpfs on /dev/shm type tmpfs (rw,nosuid)
tmpfs on /run/lock type tmpfs (rw,nosuid,nodev,noexec,relatime,size=5120k)
tmpfs on /sys/fs/cgroup type tmpfs (ro,nosuid,nodev,noexec,mode=755)
cgroup2 on /sys/fs/cgroup/unified type cgroup2 (rw,nosuid,nodev,noexec,relatime,nsdelegate)
cgroup on /sys/fs/cgroup/systemd type cgroup (rw,nosuid,nodev,noexec,relatime,xattr,name=systemd)
```

**Figura 2.4**  
Comando mount

```
cgroup on /sys/fs/cgroup/freezer type cgroup (rw,nosuid,nodev,noexec,relatime,freezer)
cgroup on /sys/fs/cgroup/perf_event type cgroup (rw,nosuid,nodev,noexec,relatime,perf_event)
cgroup on /sys/fs/cgroup/pids type cgroup (rw,nosuid,nodev,noexec,relatime,pids)
cgroup on /sys/fs/cgroup/net_cls,net_prio type cgroup (rw,nosuid,nodev,noexec,relatime,net_cls)
cgroup on /sys/fs/cgroup/memory type cgroup (rw,nosuid,nodev,noexec,relatime,memory)
cgroup on /sys/fs/cgroup/cpu,cpuacct type cgroup (rw,nosuid,nodev,noexec,relatime,cpu,cpuacct)
systemd-1 on /proc/sys/fs/binfmt_misc type autofs (rw,relatime,fd=28,pgrp=1,timeout=0,minproto=5,direct,pipe_ino=9755)
mqueue on /dev/mqueue type mqueue (rw,nosuid,nodev,noexec,relatime)
hugetlbfs on /dev/hugepages type hugetlbfs (rw,relatime,pagesize=2M)
debugfs on /sys/kernel/debug type debugfs (rw,nosuid,nodev,noexec,relatime)
/dev/sda2 on /boot type ext4 (rw,relatime,data=ordered)
/dev/sdad on /home type ext4 (rw,relatime,quota,usrquota,grpquota,data=ordered)
binfmt_misc on /proc/sys/fs/binfmt_misc type binfmt_misc (rw,nosuid,nodev,noexec,relatime)
```

**Figura 2.5**  
Comando mount

### Actividad propuesta 2.3



Prueba todas las opciones del comando *mount*, incluso ejecuta el comando *mount* en tu equipo y comprobarás qué directorios tienes montados.

- A continuación, el sistema está preparado y, más concretamente, el directorio/home está preparado para soportar cuotas. Es necesario verificar que todo es correcto mediante el comando siguiente y se observará figura 2.6.

```
# quotacheck -augmv
```

```
root@osboxes:/home# quotacheck -ugmv /home
quotacheck: Your kernel probably supports journaled quota
or journaled quota to avoid running quotacheck after an unclean
quotacheck: Quota for users is enabled on mountpoint /home
Please turn quotas off or use -f to force checking.
```

**Figura 2.6**  
Comando quotacheck

- En la figura anterior se observa que parece que da un error, pero realmente lo que está diciendo es que las cuotas están habilitadas y que si se necesita chequear es necesario desactivarlas. A continuación, se verá cómo se activan y desactivan las cuotas, con los siguientes comandos:

```
# quotaon -ugv /home
# quotaoff -ugv /home
```

6. Una vez activado el servicio de cuota con el comando quotaon, se está en disposición de crear cuota, por ejemplo, al directorio *examen* y usuario del mismo nombre que se encuentra dentro de home, para ello se hará de la siguiente forma:

```
# setquota -u examen 2048 4096 0 0 /home
```

**CUADRO 2.1**  
**Opciones de setquota**

Comando	Descripción
examen	Usuario que se le aplica la cuota.
2048	Indica que el usuario examen tiene 2048 bloques de 1k para almacenar información.
4096	Si el usuario usa más de 4095 bloques de 1k obtendrá un mensaje de aviso que ha sobrepasado el límite. Y no podrá escribir más en el disco.
0	Indica que no hay límite para soft en el inodos.
0	Indica que no hay límite para hard en el inodos.
/home	Se aplica a la partición de home.

7. Después de haber asignado la cuota al usuario examen, se almacena en el directorio examen información para que sobrepase el límite colocado anteriormente. Y se ejecuta el comando siguiente:

```
# edquota -u examen
```

```
Archivo Editar Ver Buscar Terminal Ayuda
Disk quotas for user examen (uid 1000):
Filesystem      blocks   soft    hard   inodes   soft    hard
/dev/sda4        11552   2048   4096    117      0      0
```

**Figura 2.7**  
**Comando edquota**

Si lo hacemos por grupo, el comando sería el siguiente:

```
# edquota -g NombreGrupo
```

En la figura 2.7 aparecen conceptos nuevos que se explicarán en el siguiente cuadro:

**CUADRO 2.2**  
Opciones de edquota

Comando	Descripción
Filesystem	Ruta de montaje de /home.
Blocks	Indica que el número de bloques usados por el usuario en Kb.
Soft	Indicar el número de bloques en Kb antes de recibir un warning.
Hard	Indica el límite absoluto en Kb, que no podrá sobrepasar en ningún caso.
Inodes	Especifica el número de ficheros que puede usar el usuario.
Soft	Indica que no hay límite en el número de inodos.
Hard	Indica que no hay límite en el número de inodos.

8. El siguiente paso es listar la cuota del usuario en cuestión, o la de todos. Para ello se usarán los siguientes comandos y posteriormente se mostrará una imagen con su uso (figura 2.8).

```
# quota -u examen (muestra la cuota del usuario examen)
# repquota -a (muestra la cuota de todos los usuarios)
```

```
root@osboxes:/home# quota -u examen
Disk quotas for user examen (uid 1000):
  Filesystem blocks   quota   limit   grace   files   quota   limit   grace
    /dev/sda4   11552*   2048   4096   6days     117      0      0      0
root@osboxes:/home# repquota -a
*** Report for user quotas on device /dev/sda4
Block grace time: 7days; Inode grace time: 7days
          Block limits                               File limits
User        used   soft   hard grace       used   soft   hard grace
-----
root      --  32852      0      0           13      0      0
examen    +-  11552   2048   4096  6days     117      0      0
admin      --     24      0      0           6      0      0
#2000     --     12      0      0           3      0      0
```

**Figura 2.8**  
Comando vista cuota

**Actividad propuesta 2.4**



Crea un directorio dentro del sistema ficheros y le aplicas quota.

9. Por último, se puede establecer un tiempo de gracia (grace) que permite al usuario un tiempo para poder liberar espacio; por defecto en la instalación es de 6 días (el tiempo se ajusta en segundos). Se establecen 120 segundos para el usuario examen, como se observa en la figura 2.9.

```

root@osboxes:/home# setquota -u examen -T 120 unset /home
setquota: Not setting inode grace time on /dev/sda4 because softlimit is not exceeded.
root@osboxes:/home# repquota -a
*** Report for user quotas on device /dev/sda4
Block grace time: 7days; Inode grace time: 7days
      Block limits                           File limits
User        used    soft    hard grace     used    soft    hard grace
-----
root       --  32852      0      0          13      0      0
examen    +- 11552  2048  4096 00:02      117      0      0
admin      --    24      0      0          6      0      0
#2000    --    12      0      0          3      0      0

```

**Figura 2.9**  
Opción Grace

## 2.3. Tipos de usuarios, accesos al servicio y transferencia de ficheros

### 2.3.1. Tipos de usuarios

Existen tres grandes grupos de usuarios que se pueden conectar al servidor para almacenar o recuperar información, que se comentan a continuación:

- Usuarios anónimos:* tienen acceso pero los permisos están limitados por el sistema de archivos. Para conectarse al sistema usan una cuenta simbólica como anonymous y como password una cuenta de correo electrónico. Este tipo de usuario es un agujero de seguridad, por lo que es necesario tomar las medidas oportunas para evitar posibles accesos no deseados a la información.
- Usuarios autenticados:* son aquellos que son propios del sistema operativo. Se requiere de un usuario y contraseña para entrar en el servidor FTP.
- Usuarios virtuales:* se crean independientemente del sistema operativo con sus directorios home apropiados y creados para tal fin. Servidores como proFTPD poseen este tipo de usuarios que permiten no comprometer la seguridad del sistema, ya que no están creados en el mismo. La validación de estos usuarios no tiene por qué realizarla el sistema, sino que puede ser en un fichero de texto, una base de datos como Mysql o un servicio de directorio como LDAP.

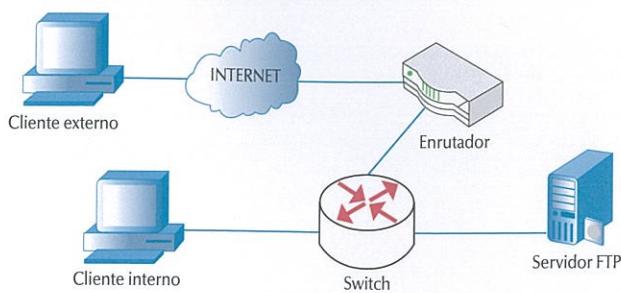


### Actividad propuesta 2.5

Comenta las diferencias entre los distintos tipos de usuarios que se pueden crear en el servicio FTP.

### 2.3.2. Tipos de accesos al servicio

Con relación al acceso al servicio de FTP, se puede acceder de diferentes formas, ya sea desde una red local o desde Internet. A continuación se pondrá un esquema de cómo acceder al servicio de FTP desde un cliente:



**Figura 2.10**  
Acceso al FTP

### 2.3.3. Tipos de transferencia de ficheros

A la hora de transferir archivos es necesario distinguir dos tipos de archivos para que la información que se traspase no sea inconsistente. Se comentan los tipos:

- ✓ *Archivos binarios*: son aquellos archivos que no son de texto y están codificados, por ejemplo, serían los archivos tipo ejecutable, imágenes, archivos de audio y vídeo, etc. El comando para poder cambiar al tipo de fichero es *binary*.
- ✓ *Archivos de texto*: son ficheros de tipo ASCII, legibles totalmente, esto es, se puede interpretar la información fácilmente. Se representa el fichero ASCII mediante 7 dígitos binarios en base decimal para representar la información. Un ejemplo de este tipo de ficheros son los que terminan en .txt, .xml, .html, .ps, etc. El comando para poder cambiar al tipo de fichero es el comando *ascii*.

#### Actividad propuesta 2.6



Define qué tipos de ficheros son: fichero Word, .html, archivo de sonido y archivo de vídeo.

Es crucial saber de antemano qué ficheros se van a transferir, ya que, si no se usan las opciones adecuadas, se podría destruir la información. El servicio FTP permite configurar las opciones adecuadas en la transferencia de ficheros.

En la figura 2.11 se demuestra cómo se usan las dos opciones anteriores en la línea de comandos de la consola del servicio FTP:

```

root@osboxes:/etc/proftpd# ftp localhost
Connected to localhost.
220 ProFTPD Server (Servidor FTP) [::1]
Name (localhost:root): admin
331 Contraseña necesaria para admin
Password:
230 Usuario admin conectado
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> binary
200 Tipo establecido en I
ftp> ASCII
?Invalid command
ftp> ascii
200 Tipo establecido en A
ftp> 

```

**Figura 2.11**  
Tipos de ficheros

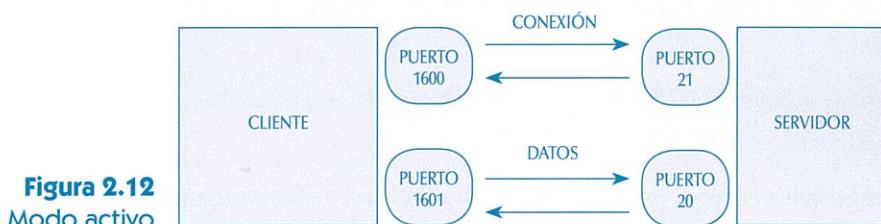
## 2.4. Modos de conexión al cliente

Cuando se realiza la comunicación entre el cliente y servidor existen dos modos de conexión por parte del cliente: modo activo y pasivo.

### 2.4.1. Modo activo

En el modo activo el servidor siempre crea un canal para datos por el puerto 20, mientras que el cliente asocia un puerto aleatorio mayor que 1024. El cliente envía un paquete al servidor, indicando el número de puerto para transferir archivos.

Se puede observar en la siguiente figura el modo activo en funcionamiento:

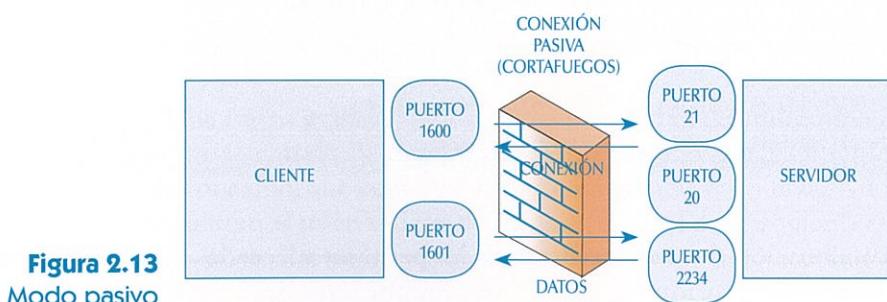


**Figura 2.12**  
Modo activo

### 2.4.2. Modo pasivo

Sin embargo, en modo pasivo, es el cliente quien comienza la conexión con el servidor para evitar bloqueos de conexión mediante configuraciones NAT o cortafuegos. En este modo, el cliente inicia ambas conexiones, control y datos. En este caso, si no existiera cortafuegos no habría ningún problema. Pero, al existir cortafuegos, el servidor, que intenta conectarse, devuelve la respuesta por un puerto diferente, que hace que el cortafuegos bloquee la conexión.

Con cortafuegos el diagrama de la conexión sería el siguiente:



**Figura 2.13**  
Modo pasivo

## 2.5. Protocolo seguro de transferencia de archivos

La instalación que se va a realizar es el servidor FTP en modo seguro con certificado y configurado en el fichero tls.conf para SO Linux bajo la versión de Debian y, más concretamente, de

Kali Linux. La instalación se puede realizar en una máquina virtual, que posee una imagen de este sistema operativo o en una máquina física. El procedimiento de instalación es el siguiente:

1. El primer paso es descomentar la línea que permite incluir el fichero `tls.conf` para configurar la conexión segura:

```
#include /etc/proftpd/tls.conf
```

2. Posteriormente, habría que eliminar el comentario en las siguientes líneas del fichero `tls.conf` para comprobar que funciona la seguridad en la conexión con el servidor FTP (figura 2.14).



**Figura 2.14**  
Fichero `tlf.conf`

3. El siguiente paso es generar las claves públicas que se colocarán en la ruta `/etc/ssl` mediante el comando `proftpd-gencert`, y obtenemos la pantalla de la figura 2.15.

```
root@osboxes:/etc/proftpd# proftpd-gencert
Generating a 2048 bit RSA private key
.....+++++
.....+++++
writing new private key to '/etc/ssl/private/proftpd.key'
-----
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]:Cordoba
Locality Name (eg, city) []:Cordoba
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Educacion
Organizational Unit Name (eg, section) []:Practica
Common Name (e.g. server FQDN or YOUR name) []:FTP
Email Address []:

Use the following information in your ProFTPD configuration:
TLSRSACertificateFile    /etc/ssl/certs/proftpd.crt
TLSRSACertificateKeyFile /etc/ssl/private/proftpd.key
```

**Figura 2.15**  
Fichero certificado

4. A continuación, se va a proceder a dar los permisos adecuados a los ficheros generados:

```
#chmod 600 /etc/ssl/private/proftpd.key
#chmod 644 /etc/ssl/certs/proftpd.crt
```

5. Por último, reiniciamos el servidor FTP con el comando siguiente. Y obtendremos la conexión segura con certificado, como se puede observar en la pantalla de la figura 2.16.

```
#service proftpd restart
```

**Figura 2.16**  
Fichero certificado FTP

El certificado del servidor es desconocido. Por favor, examine cuidadosamente el certificado para asegurarse de que se puede confiar en el servidor.	
Detalles	
Desde válido:	30/03/20 14:26:37
Válido hasta:	30/03/21 14:26:37
Número de serie:	00:db:a0:9c:0a:30:9a:75:14
Algoritmo de clave pública:	RSA con 2048 bits
Algoritmo de firma:	RSA-SHA256
Huella digital (SHA-256):	66:57:35:cc:b3:79:55:0e:79:07:11:ed:ab:02:c2:90:f6:56:3f:c7:0e:a8:e1:42:0b:4a:63:8a:32:21:b4:3a
Huella digital (SHA-1):	0d:f0:f2:fb:ad:11:db:a1:a7:87:13:ca:9a:fd:78:82:57:d0:51:fc
Asunto del certificado	Agente de certificado
Nombre común:	FTP
Organización:	Educacion
Unidad:	Practica
País:	ES
Estado o provincia:	Cordoba
Localidad:	Cordoba
Agentes de certificado	
Nombre común:	FTP
Organización:	Educacion
Unidad:	Practica
País:	ES
Estado o provincia:	Cordoba
Localidad:	Cordoba
Detalles de la sesión	
Sitio:	10.0.2.15:21
Protocolo:	TLS1.3
Intercambio de clave:	ECDHE-RSA
Cifrado:	AES-256-GCM
MAC:	AEAD



### Actividad propuesta 2.7

Practica con la herramienta de generar certificados para crear un certificado digital para el servicio FTP.

## 2.6. Utilización de herramientas gráficas y en modo texto. Comandos

En este apartado se va a comentar uno de los interfaces gráficos más usados en el mundo de Internet y, más concretamente, a la hora de realizar transferencias de ficheros vía FTP. Posteriormente, se verán los comandos usados en la línea de comandos en la consola de FTP. Pero este método no es intuitivo ni práctico, por ello se usa una interfaz gráfica o el navegador.

### 2.6.1. Herramientas

Todos los clientes de FTP de forma gráfica funcionan y se comportan del mismo modo. Normalmente poseen unos campos de entrada de datos típicos. La IP del servidor FTP remoto, usuario, contraseña y el puerto de conexión, que como se sabe, es configurable en el fichero de configuración del servicio FTP. Por último, el interfaz tiene dos paneles, el panel de la izquierda es la máquina local y el panel de la derecha es el servidor remoto cuando se realiza la conexión. Por lo que el traspaso de información es tan fácil como arrastrar el fichero o directorios que se desee traspasar de un panel a otro, siempre teniendo en cuenta que se posea permiso para realizar la acción.

Se van a listar los cinco clientes FTP gratuitos más usados o mejores que existen en el mercado:

- **Filezilla:** es un cliente FTP Open Source que es rápido y capaz de manejar conexiones simultáneas. Soporta SFTP y FTPS. Además, está disponible para todos los sistemas operativos, Mac OS, Linux y Windows.

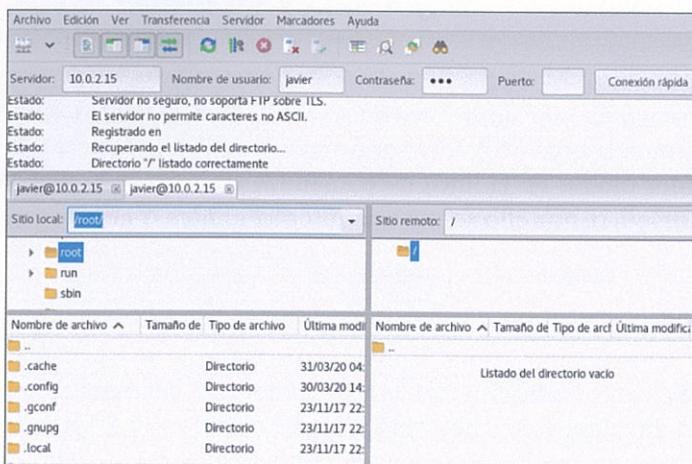
- *Cyberduck*: es un software para gran cantidad de información y soporta WebDAW Amazon S3, etc. Disponible para Windows y Mac OS.
- *Classic FTP*: es un cliente para transferir archivos y cuenta con una interfaz sencilla e intuitiva. Disponible para Windows y Mac OS.
- *WinSCP*: este cliente es el diseñado exclusivamente para Windows y tiene muchas características. Maneja múltiples transferencias de archivos, tiene un pequeño editor de texto para cambios rápidos y una consola para usuarios avanzados.
- *GFTP*: es uno de los clientes más usados junto con Filezilla. Es muy sencillo pero a la vez muy potente. Soporta http, https, ssh, fsp, ftp y ftps. En un Ubuntu y en la mayoría de las distribuciones de Linux se instala como un paquete con el comando apt-get install gftp.



#### PARA SABER MÁS

El primer FTP se usó en abril de 1971.

Vamos a entrar en detalle con el cliente FTP Filezilla, y para ello vamos a mostrar una imagen de su interfaz conectado a un servidor FTP que se ha configurado anteriormente:



**Figura 2.17**  
Filezilla

Filezilla se compone de un amplio menú que se puede comprobar sin mayor complicación:

- ✓ *Servidor*: se puede teclear en este campo la IP o el nombre DNS del servidor FTP.
- ✓ *Nombre usuario*: nombre de usuario que está dado de alta en el servicio FTP.
- ✓ *Contraseña*: la cadena de caracteres secreta que permite la conexión al servidor FTP.
- ✓ *Puerto*: el número decimal configurado para atender peticiones del servidor FTP.
- ✓ *Conexiones simultáneas*: puede existir más de una conexión al servidor u otros servidores FTP.

- ✓ **Sitio local:** el directorio del equipo local que se conecta al servidor FTP.
- ✓ **Sitio remoto:** la ubicación del servidor FTP habilitado para el traspaso o descarga de información.

Por otro lado, se puede conectar al servidor FTP mediante línea de comandos y ejecutar distintos comandos para transferir información entre el cliente y el servidor. En la figura siguiente se puede observar cómo se puede conectar y posteriormente ejecutar distintas operaciones:



### Actividad propuesta 2.8

Ejecuta un cliente FTP para familiarizarte con él y observa todas las opciones que contiene.

```
root@osboxes:/etc/proftpd# ftp localhost
Connected to localhost.
220 ProFTPD Server (Servidor FTP) [::1]
Name (localhost:root): admin
331 Contraseña necesaria para admin
Password:
230 Usuario admin conectado
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> !pwd
/etc/proftpd
ftp> pwd
257 "/" es el directorio corriente
ftp> !ls
blacklist.dat  ldap.conf      passwd.virtuales prueba.txt      tls.conf
conf.d         modules.conf   proftpd.conf       sql.conf      usuario.ldif
dhparams.pem  ou.ldif        prueba           testdisk.log  virtuals.conf
ftp> ls
200 Comando EPRT exitoso
150 Abriendo ASCII modo conexión de datos para file list
-rw-r--r--  1 admin      pulse          20 Mar 30 17:20 prueba.txt
226 Transferencia completada
ftp> 
```

**Figura 2.18**  
Comandos

## 2.6.2. Comandos

En esta sección se van a comentar los comandos más importantes que se usan en la consola de FTP y que permite realizar operaciones en el servidor. Se han agrupado en tres cuadros, el primero son comandos para la conexión, el segundo está relacionado con los directorios y el último con los ficheros.

**CUADRO 2.3**  
Comandos de conexión

Comando	Descripción
open [IP]	Abre una conexión con la IP que tecleemos.
user [usuario]	Solicita un usuario para autenticar.
bye/exit/quit	Se sale del interfaz de comandos de FTP, nos devuelve al sistema.
close	Cierra la conexión del usuario activo, sin salirnos de la consola FTP.

**CUADRO 2.4****Comandos para directorios**

Comando	Descripción
<code>pwd</code>	Muestra la ruta de donde nos encontramos del equipo destino.
<code>!pwd</code>	Muestra la ruta de donde nos encontramos del equipo local.
<code>ls</code>	Lista la información del directorio del equipo destino.
<code>!ls</code>	Lista la información del directorio del equipo local.
<code>mkdir</code>	Crea un directorio en el equipo destino si tiene permisos.
<code>!mkdir</code>	Crea un directorio en el equipo local.
<code>rmdir</code>	Elimina un directorio en el equipo destino si tiene permisos.
<code>!rmdir</code>	Elimina un directorio en el equipo local.

**CUADRO 2.5****Comandos para ficheros**

Comando	Descripción
<code>get [archivo]</code>	Recupera un archivo del servidor destino o remoto.
<code>mget [archivo]</code>	Recupera una lista de archivos que cumplan un patrón del servidor destino.
<code>put [archivo]</code>	Transfiere un archivo del servidor local al servidor remoto.
<code>mput [archivo]</code>	Transfiere una lista de archivos que cumplan un patrón del servidor local al servidor remoto.
<code>binary</code>	Cambia el tipo de transferencia a binario.
<code>ascii</code>	Cambia el tipo de transferencia a texto.
<code>Delete [archivo]</code>	Borra un archivo en el servidor destino si tiene permisos.
<code>mdelete</code>	Borra archivos en base a un patrón en el servidor destino si tiene permisos.

**Actividad propuesta 2.9**

Entra con el cliente FTP en el servidor FTP y practica con los comandos anteriores, sobre todo la diferencia de comandos entre el servidor y el cliente.

**2.7. Instalación y configuración del servidor proFTPd en SO Linux**

La instalación que se va a realizar es del servicio proFTPd y el cliente FTP para SO Linux bajo la versión de Debian y, más concretamente, de Kali Linux. La instalación se va a llevar a cabo en una máquina virtual, que consiste en crear un usuario virtual llamado admin y transferir un archivo de muestra llamado prueba.txt. Tal configuración consta de los siguientes pasos:

- En primer lugar se instalan tanto el servidor proFTPD como el servicio FTP del cliente para poder conectarnos al servidor, como se puede observar en las pantallas de las figuras 2.19 y 2.20:

```
root@osboxes:~# apt-get install proftpd
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
Nota, seleccionando «proftpd-basic» en lugar de «proftpd»
Se instalarán los siguientes paquetes adicionales:
  libhiredis0.14 libmemcached11 libmemcachedutil2 proftpd-doc
Paquetes sugeridos:
  proftpd-mod-ldap proftpd-mod-mysql proftpd-mod-odbc proftpd-mod-pgsql
  proftpd-mod-sqlite proftpd-mod-geoip proftpd-mod-snmp
```

**Figura 2.19**

Instalación servidor proFTPD

```
root@osboxes:~# apt-get install ftp
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
Se instalarán los siguientes paquetes NUEVOS:
  ftp
0 actualizados, 1 nuevos se instalarán, 0 para eliminar y 1951 no actualizados.
Se necesita descargar 59,6 kB de archivos.
Se utilizarán 141 kB de espacio de disco adicional después de esta operación.
Des:1 http://kali.download/kali kali-rolling/main amd64 ftp amd64 8.17-34.1+b1 [59,6 kB]
Descargados 59,6 kB en 2s (20,3 kB/s)
/usr/share/apt-listchanges/apt_listchanges.py:540: FutureWarning: Possible neste
d set at position 25
    email_re = re.compile(r'([a-zA-Z0-9_+\.\-\.]+)@(([0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.)|(([a-zA-Z0-9\-\_]+\.)+)([a-zA-Z]{2,4}|[0-9]{1,3})\(\?\)'')
Selecciónando el paquete ftp previamente no seleccionado.
```

**Figura 2.20**

Instalación cliente FTP

- A continuación, se va a crear un directorio de destino donde se va a ubicar el usuario del FTP cuando se conecte. Se crea con la siguiente sentencia:

```
# mkdir /var/ftp
# mkdir /var/ftp/bbdd
```

- El siguiente paso sería obtener el uid (User ID), esto es, la identificación del usuario FTP del sistema, que se realiza con el siguiente comando y se observa en la figura 2.21:

```
# id ftp
```

```
root@osboxes:/etc/proftpd# id ftp
uid=137(ftp) gid=65534(nogroup) grupos=65534(nogroup)
root@osboxes:/etc/proftpd#
```

**Figura 2.21**

Obtención del id

- Con el uid obtenido con el anterior comando estamos en disposición de ejecutar el comando que nos permite crear el usuario virtual admin. Este usuario permitirá conectarnos al servidor FTP instalado con las opciones home (carpeta donde se ubicará el usuario), uid (identificación del usuario), Shell (para conectarse al sistema), file (fichero que almacenará la información del usuario) y la opción passwd para teclear la contraseña que poseerá el usuario admin. Se puede observar en la pantalla de la figura 2.22.

```

root@osboxes:/etc/proftpd# ftppasswd --passwd --name admin --home /var/ftp/bbdd
--uid 137 --shell /bin/false --file /etc/proftpd/passwd.virtuales
ftppasswd: using alternate file: /etc/proftpd/passwd.virtuales
ftppasswd: --passwd: missing --gid argument: default gid set to uid
ftppasswd: updating passwd entry for user admin

ftppasswd: /bin/false is not among the valid system shells. Use of
ftppasswd: "RequireValidShell off" may be required, and the PAM
ftppasswd: module configuration may need to be adjusted.

Password:
Re-type password:
ftppasswd: entry updated
root@osboxes:/etc/proftpd#

```

**Figura 2.22**

**Creación de la password del usuario admin**

5. Se le da permisos al directorio donde se va a ubicar el usuario de FTP llamado admin, de la siguiente forma:

```
# chown -R ftp /var/ftp/bbdd
```

6. En este paso, se configurará el fichero de configuración /etc/proftpd/proftpd.conf con las siguientes líneas:

```

AuthUserFile /etc/proftpd/passwd.virtuales
RequireValidShell off

```

En el fichero passwd.virtuales se ha añadido el usuario admin con la password curso, tal usuario será el que se utilizará para conectarnos al servidor FTP. Y la directiva RequireValidShell que permite validar al usuario sin shell del sistema.

7. Por último, se va a realizar una prueba para comprobar que funciona correctamente el servicio de FTP con los siguientes comandos:

```

# ftp localhost
#!ls
# ascii
# put prueba.txt

```

```

root@osboxes:~# ftp localhost
Connected to localhost.
220 ProFTPD Server (Servidor FTP) [::ffff:127.0.0.1]
Name (localhost:root): admin
331 Contraseña necesaria para admin
Password:
230 Usuario admin conectado
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ascii
200 Tipo establecido en A
ftp> !ls
Desktop Downloads Pictures Public Videos
Documents Music prueba.txt Templates
ftp> put prueba.txt
local: prueba.txt remote: prueba.txt
200 Comando PORT exitoso
150 Abriendo ASCII modo conexión de datos para prueba.txt
226 Transferencia completada
21 bytes sent in 0.00 secs (1.4305 MB/s)
ftp> 

```

**Figura 2.23**

**Demostración del funcionamiento del servicio FTP**



## Actividad propuesta 2.10

Conéctate y comprueba que eres capaz de instalar el servidor FTP y configurar las diferentes opciones.

### 2.7.1. Validación mediante un host virtual

En la siguiente instalación se va a configurar un host virtual para que la conexión sea desde una dirección IP configurada (192.168.1.15). Siguiendo la configuración anterior, habría que establecer ciertas líneas en los ficheros de configuración:

- Partiendo de la instalación anterior, se realizarán las siguientes acciones. La primera es activar o descomentar en el fichero proftpd.conf la siguiente línea:

```
#include /etc/proftpd/virtuals.conf
```

- La línea anterior permite incluir el fichero de configuración virtuals, donde se tendrían que configurar las opciones para poder conectarnos al servidor con una determinada IP. Para ello, se usarán las siguientes líneas:

```
<VirtualHost 192.168.1.15>
    ServerAdmin      ftpmaster@server.com
    ServerName       "FTP Server"
    AuthUserFile    /etc/proftpd/passwd.virtuales
    MaxLoginAttempts 3
    RequireValidShell   no
    DefaultRoot     /var/ftp/bbdd
    AllowOverwrite   yes
</VirtualHost>
```

La explicación a las opciones anteriores son las siguientes:

- ServerAdmin*: sirve para añadir un correo electrónico.
  - ServerName*: nombre del servidor FTP.
  - AuthUserFile*: fichero de autorización de usuarios.
  - MaxLoginAttempts*: número máximo de intentos fallidos.
  - RequireValidShell*: se usa para confirmar si se quiere que el usuario pertenezca al sistema operativo.
  - DefaultRoot*: ruta en la que se ubicará el usuario de FTP cuando se conecte.
  - AllowOverwrite*: para poder escribir en la ruta anterior.
- Posteriormente, es necesario configurar la IP 192.168.1.5 en el fichero */etc/network/interfaces*, después se debería reiniciar la interfaz *eth0* y la red mediante el comando *service networking restart*. Como se puede observar en la pantalla de la figura 2.24, nos conectamos al servidor FTP y transferimos el fichero prueba.txt.

```
root@osboxes:~# ftp 192.168.1.15
Connected to 192.168.1.15.
220 ProFTPD Server (FTP Server) [::ffff:192.168.1.15]
Name (192.168.1.15:root): admin
331 Contraseña necesaria para admin
Password:
230 Usuario admin conectado
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> !ls
Desktop    Downloads    Pictures    Public    Videos
Documents  Music       prueba.txt  Templates
ftp> put prueba.txt
local: prueba.txt remote: prueba.txt
200 Comando PORT exitoso
150 Abriendo BINARY modo conexión de datos para prueba.txt
226 Transferencia completada
20 bytes sent in 0.00 secs (27.1645 kB/s)
ftp> █
```

**Figura 2.24**  
FTP con host virtual

### 2.7.2. Validación del servicio FTP mediante LDAP

Por último, se va a proceder a la validación del usuario *javier* y password *cat* mediante el acceso al servicio de directorio LDAP. Para ello son necesarios los siguientes pasos:

1. En primer lugar, habría que instalar el servicio de directorio *LDAP*, anteriormente explicado en este libro, y el servicio *proFTPD*, que ya está instalado de las configuraciones anteriores, con los siguientes parámetros:
  - *Dominio*: dc=cursolinux,dc=com.
  - *Organización*: cursolinux.
  - *Contraseña*: teclear dos veces admin admin.
  - Las demás opciones por defecto.
2. En segundo lugar, es preciso instalar el módulo que conecta proFTPD con LDAP con el comando:

```
#apt-get install proftpd-mod-ldap (figura 2.25)
```

```
root@osboxes:/etc/proftpd# apt-get install proftpd-mod-ldap
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
Se instalarán los siguientes paquetes adicionales:
  proftpd-basic
Paquetes sugeridos:
  proftpd-mod-mysql proftpd-mod-odbc proftpd-mod-pgsql proftpd-mod-sqlite
  proftpd-mod-geoip proftpd-mod-snmp
Se instalarán los siguientes paquetes NUEVOS:
  proftpd-mod-ldap
Se actualizarán los siguientes paquetes:
  proftpd-basic
1 actualizados, 1 nuevos se instalarán, 0 para eliminar y 1994 no actualizados.
Se necesita descargar 3.165 kB de archivos.
Se utilizarán 577 kB de espacio de disco adicional después de esta operación.
¿Desea continuar? [S/n] S
Des:1 http://kali.download/kali kali-rolling/main amd64 proftpd-basic amd64 1.3.6c-2+b1 [2.657 kB]
```

**Figura 2.25**  
Instalación módulo LDAP

3. A partir de ahora, ya se está en disposición de configurar los ficheros *proftpd.conf*, *ldap.conf* y *modules.conf*. Se comienza con la configuración en el fichero *proftpd.conf* con las siguientes líneas:
  - **Include /etc/proftpd/ldap.conf** (permitirá que el fichero de configuración *proftpd.conf* incluya la conexión a LDAP).

- Hay que descomentar la directiva `RequireValidShell off` (permite entrar al usuario sin shell válida).
4. En el fichero `modules.conf` se descomenta la línea siguiente:
- ```
LoadModule mod_ldap.c
```
5. El último fichero por configurar es `ldap.conf`. Hay que añadir las líneas de la figura 2.26.

```
LDAPLog /var/log/proftpd/ldap.log

LDAPServer ldap://10.0.2.15:389/??sub
LDAPBindDN "cn=admin,dc=cursolinux,dc=com" "admin"
LDAPUsers "ou=usuarios,dc=cursolinux,dc=com" "(&(uid=%v) (objectclass=*))"
```

**Figura 2.26**  
Configuración `ldap.conf`

La primera línea es para configurar un log relacionado con LDAP para comprobar que el usuario es validado por el servicio de directorio.

Las tres siguientes son para la conexión al servicio LDAP y son las siguientes:

- `LDAPServer`: especifica el nombre o IP del servidor.
- `LDAPBindDN`: configura la entrada absoluta al servicio de directorio.
- `LDAPUsers`: detalla la unidad organizativa en la que se va a buscar el usuario dentro del directorio activo.

#### RECUERDA

- ✓ Las tres opciones de LDAP (`LDAPServer`, `LDAPBindDN` y `LDAPUsers`) permiten configurar el módulo LDAP para validar un usuario del servicio FTP.

6. El siguiente paso sería añadir al servicio de directorio el usuario que permite validar tal usuario cuando se conecta al servicio de FTP. La figura 2.27 crea una unidad organizativa llamada `usuarios` y la figura 2.28 crea un usuario llamado `javier` y de password `cat`:

```
dn:ou=usuarios,dc=cursolinux,dc=com
objectClass: organizationalUnit
ou:usuarios
```

```
dn:uid=javier,ou=usuarios,dc=cursolinux,dc=com
objectclass: account
objectclass: posixAccount
objectclass: top
uid: javier
cn: Javier
uidNumber: 2002
gidNumber: 2002
homeDirectory: /srv/ftp/javier
userPassword: cat
loginShell: /bin/false
```

**Figura 2.27**  
Configuración `ou.ldif`

**Figura 2.28**  
Configuración `usuario.ldif`

Para añadir estos ficheros al servicio de directorio habría que teclear en la línea de comandos de un terminal las siguientes líneas, que permitirán añadir a LDAP los dos ficheros anteriores:

```
#ldapadd -D cn=admin,dc=cursolinux,dc=com -w admin -f ou.ldif -c
#ldapadd -D cn=admin,dc=cursolinux,dc=com -w admin -f usuario.ldif -c
```

Para comprobar que esta información es correcta, se teclea slapcat (figura 2.29).

```
dn: ou=usuarios,dc=cursolinux,dc=com
objectClass: organizationalUnit
ou: usuarios
structuralObjectClass: organizationalUnit
entryUUID: 42ad75aa-07cc-103a-8bad-ef0af431b3a0
creatorsName: cn=admin,dc=cursolinux,dc=com
createTimestamp: 20200331185039Z
entryCSN: 20200331185039.760490Z#000000#000#000000
modifiersName: cn=admin,dc=cursolinux,dc=com
modifyTimestamp: 20200331185039Z

dn: uid=javier,ou=usuarios,dc=cursolinux,dc=com
objectClass: account
objectClass: posixAccount
objectClass: top
uid: javier
cn: Javier
uidNumber: 2020
gidNumber: 2020
homeDirectory: /srv/ftp/javier
userPassword:: Y2F0
loginShell: /bin/false
structuralObjectClass: account
entryUUID: 45eeff62-07cc-103a-8bae-ef0af431b3a0
creatorsName: cn=admin,dc=cursolinux,dc=com
createTimestamp: 20200331185045Z
entryCSN: 20200331185045.222944Z#000000#000#000000
modifiersName: cn=admin,dc=cursolinux,dc=com
modifyTimestamp: 20200331185045Z
```

**Figura 2.29**  
Información LDAP

7. Ahora es el momento de crear el directorio en el que se va a ubicar el usuario javier cuando se conecte:

```
#mkdir -p /srv/ftp/javier
#chgrp 2020 /srv/ftp
#chown 2020 /srv/ftp/javier
#chmod 700 /srv/ftp/javier
```

Estas líneas permiten crear el directorio, darle permisos de grupos y de propietario asociados al uidNúmero y gidNúmero que es el 2020, y darle permisos al propietario del directorio.

8. Por último, se reinicia el servicio de proFTPD y se prueba la configuración del usuario javier y password cat, para lo que realizamos los siguientes pasos:

```
#service proftpd restart
# ftp 10.0.2.15
```

Y a partir de aquí podemos observar la pantalla de la figura 2.30 y el log de LDAP para observar que todo funciona correctamente.

```
root@osboxes:/etc/proftpd# service proftpd restart
root@osboxes:/etc/proftpd# ftp 10.0.2.15
Connected to 10.0.2.15 [controls log] [proftpd log]
220 ProFTPD Server (Servidor FTP) [:ffff:10.0.2.15]
Name (10.0.2.15:root): javier
331 Contraseña necesaria para javier
Password: [password]
230 Usuario javier conectado
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> [downloads]
```

```
2020-03-31 14:52:22,278 mod_ldap/2.9.4[3148]: attempting connection to URL ldap://10.0.2.15:389/?sub
2020-03-31 14:52:22,278 mod_ldap/2.9.4[3148]: set LDAP protocol version to 3
2020-03-31 14:52:22,278 mod_ldap/2.9.4[3148]: connected to URL ldap://10.0.2.15:389/?sub
2020-03-31 14:52:22,278 mod_ldap/2.9.4[3148]: set dereferencing to 0
2020-03-31 14:52:22,278 mod_ldap/2.9.4[3148]: set query timeout to 5 secs
2020-03-31 14:52:22,279 mod_ldap/2.9.4[3148]: generated filter ou=usuarios,dc=cursolinux,dc=com from template ou=usuarios,dc=cursolinux,dc=com and value javier
2020-03-31 14:52:22,279 mod_ldap/2.9.4[3148]: generated filter (&(uid=javier) (objectclass=*)) from template (&(uid=%v) (objectclass=*)) and value javier
2020-03-31 14:52:22,279 mod_ldap/2.9.4[3148]: searched under base DN ou=usuarios,dc=cursolinux,dc=com using filter (&(uid=javier) (objectclass=*))
2020-03-31 14:52:22,279 mod_ldap/2.9.4[3148]: fetching values for attribute uid
2020-03-31 14:52:22,279 mod_ldap/2.9.4[3148]: fetching values for attribute uidNumber
2020-03-31 14:52:22,279 mod_ldap/2.9.4[3148]: fetching values for attribute gidNumber
2020-03-31 14:52:22,279 mod_ldap/2.9.4[3148]: fetching values for attribute homeDirectory
2020-03-31 14:52:22,279 mod_ldap/2.9.4[3148]: fetching values for attribute loginShell
2020-03-31 14:52:22,279 mod_ldap/2.9.4[3148]: found user javier, UID 2020, GID 2020, homedir /srv/ftp/javier, shell /bin/false
```

**Figura 2.30**  
Demostración de validación LDAP

## 2.8. Utilización del servicio de transferencia de archivos

### 2.8.1. Desde el navegador

Hoy en día los navegadores que existen en el mercado usan normalmente el protocolo HTTP pero la mayoría de los usuarios desconocen que tales navegadores tienen otro uso. Ese uso se refiere al empleo de otros protocolos, como pueden ser el FTP. Como se comentó anteriormente, se pueden usar herramientas diseñadas para ello, pero si se emplea el navegador se puede descargar información perfectamente.

Si se quiere usar el navegador como cliente FTP, solamente es necesario teclear en la barra de direcciones del navegador la siguiente línea:

ftp://nombre\_servidor\_ftp o dirección IP: puerto

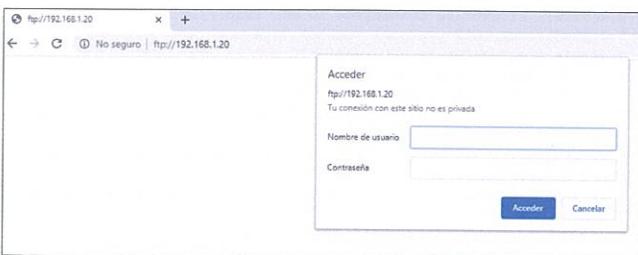
Se pueden observar en el cuadro 2.6 los componentes de la línea anterior:

**CUADRO 2.6**  
Componentes de la URL

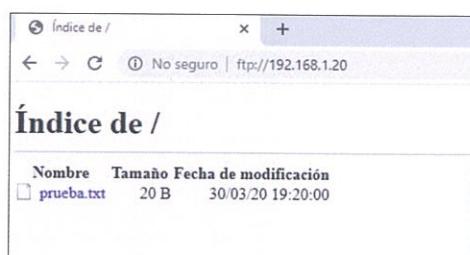
| Comando         | Descripción                                                                                                                                  |
|-----------------|----------------------------------------------------------------------------------------------------------------------------------------------|
| FTP             | Protocolo usado para la conexión.                                                                                                            |
| Nombre_servidor | Nombre del servidor FTP que acepta conexiones desde los clientes.                                                                            |
| IP              | IP del servidor FTP que acepta conexiones desde los clientes.                                                                                |
| puerto          | Número decimal que indica el puerto configurado en el servidor FTP para atender las peticiones de los clientes. Por defecto es el puerto 21. |

En nuestro caso, la IP del servidor es 192.168.1.20, lo único que hay que hacer es teclear en la barra de direcciones del navegador `ftp://192.168.1.20`, no es necesario el puerto porque está configurado por defecto. En caso de que sea otro, sí es necesario teclearlo. La conexión se observa en la figura 2.31.

Si se teclean el usuario admin y la password curso se entrará en la carpeta correspondiente y configurada para ello en el servicio FTP, como se muestra en la figura 2.32.



**Figura 2.31**  
URL Navegador



**Figura 2.32**  
URL Navegador

Si se observa el navegador de cliente FTP, no es muy funcional que digamos, simplemente sirve para descargar ficheros del servidor FTP. Incluso tampoco permite reanudar la conexión como cualquier programa que existe en el mercado, ejemplo de Filezilla. Por lo que para un uso puntual es efectivo, pero poco más. Si se desea descargar el fichero prueba.txt de la figura simplemente se pulsa sobre él y automáticamente se descargará o guardará como en otros navegadores como Firefox. Google se está planteando no dar más soporte al protocolo FTP en sus navegadores.

## 2.8.2. En el proceso de despliegue de la aplicación web

Para desplegar una aplicación, es necesario subir el código de la misma a un servidor web para que pueda ser usada por los usuarios para una tarea determinada. Por lo que este capítulo permite usar el servicio de FTP para desplegar una aplicación.

Suele ser habitual que cualquier aplicación web del mercado pueda permitir la transferencia de archivos que componen el código de la aplicación, ya sea propia de la empresa o de terceros entre el cliente y un servidor.

Es importante conocer cuánto tiempo se tiene para mantener la conexión abierta entre el cliente y el servidor, puesto que será necesario subir gran cantidad de información.

El servidor web permite desplegar las aplicaciones que se programan. Para subir el código de las aplicaciones es necesario configurar el tiempo de conexión y el tamaño de los archivos para no colapsar ni el cliente ni el servidor. Hay que tener en cuenta que si el servidor web está en producción sería necesario controlar este aspecto o realizar una parada técnica para que los usuarios no sufran cortes innecesarios.

Hoy en día estos problemas son fácilmente solucionables por parte de los hosting de Internet, que permiten acceder mediante aplicaciones FTP, como Filezilla, Cyberduck, o cualquier otra del mercado.

Además, también suelen permitir conexiones seguras implementando protocolos seguros como SSL, de hecho, proFTPD permite una implementación segura mediante certificados (implementada en este capítulo).

En conclusión, es necesario que uses una aplicación segura para que no comprometa tu seguridad ni fuga de información con relación al código de tu aplicación.

## Resumen

- En este segundo capítulo se ha explicado la instalación y administración del servidor de transferencia de ficheros.
- El servicio de FTP es el encargado de transferir información desde una IP a otra (más concretamente ficheros) e interviene como factor principal en el despliegue de una aplicación web.
- Existen tres tipos de usuarios para configurar en cualquier servicio de FTP que resultan clave para controlar la seguridad.
- Es importante conocer los modos de conexión del cliente al servidor, sobre todo si existe un cortafuegos que controla la conexión de entrada y salida.
- Debido a la cantidad de información que existe en la red y la existencia de amenazas de interceptación de información, se hace indispensable configurar un protocolo de comunicación seguro, como es SSL en el servicio FTP.
- Se desarrolla la instalación y configuración de un servicio de FTP, como es proFTPD y la validación del FTP mediante una validación externa.
- Se comentan las herramientas gráficas y comandos del servicio FTP que intervienen directamente en la interacción entre el cliente y el servidor.
- Por último, se detallan las nociones básicas de la intervención de este servicio en el despliegue de una aplicación en cualquier entorno de producción.

## Supuestos prácticos

1. Una vez instalado el servicio FTP, prueba los comandos propios del interfaz de FTP analizados a lo largo del capítulo.
2. Crea un fichero llamado *prueba.txt* y asignale los diferentes permisos que tiene el fichero mediante el comando *chmod*.
3. Una vez configurado el servicio de FTP, transmite un fichero de forma binaria y otro en modo texto desde el cliente al servidor.
4. Instala al menos dos programas de interfaz gráfica de servicio FTP y haz pruebas con ellos.



## Ejercicios propuestos

1. Instala y configura el servicio de FTP pero con el paquete vsftpd para que pueda entrar en el sistema un usuario llamado prueba y password prueba. También debe tener asignado un directorio para poder descargarse información. Este directorio debe estar acotado para que no pueda subir o bajar por el árbol de directorios.
2. Instala y configura vsftpd con LDAP. Que un usuario se valide mediante la declaración de este en el servicio de directorio LDAP.
3. Aprovechando que conocemos la instalación de un DNS en nuestro sistema, configura el servicio proFTPD para que funcione con el DNS, por ejemplo, declara el registro DNS local ftpexterno y que se conecte a nuestro servicio FTP a través de este nombre.
4. Instala y configura del servicio de *quota* en el servidor Linux. Comprueba que funciona correctamente.

## ACTIVIDADES DE AUTOEVALUACIÓN

1. ¿Qué cifrado lleva el servicio FTPS?:  
 a) SLP.  
 b) SLA.  
 c) WPA.  
 d) SSL/TLS.
2. ¿Qué dos puertos utiliza el servicio FTP?:  
 a) El puerto 22 para la conexión y el 20 para los datos.  
 b) El puerto 23 para la conexión y el 20 para los datos.  
 c) El puerto 21 para la conexión y el 20 para los datos.  
 d) El puerto 20 para la conexión y el 22 para los datos.
3. ¿En qué modo se puede mandar un archivo zip por el FTP?:  
 a) Hexadecimal.  
 b) Decimal.  
 c) Binario.  
 d) ASCII.
4. ¿Qué comando nos permite mostrar el directorio activo en el equipo local?:  
 a) lcd.  
 b) pwd.  
 c) !pwd.  
 d) dir.

5. ¿Cuántos tipos de usuarios existen en el FTP?:

- a) 3.
- b) 4.
- c) 2.
- d) 5.

6. ¿Qué modos de conexión tiene el cliente FTP?:

- a) Activo, pasivo y recíproco.
- b) Activo y pasivo.
- c) Forma de árbol SN.
- d) Ninguna de las respuestas anteriores es correcta.

7. ¿Cuál es el fichero de configuración del servicio de proFTPD?:

- a) proftpd.ini.
- b) proftpd.setup.
- c) ftp.conf.
- d) proftpd.conf.

8. ¿Qué función nos permite realizar el servicio *quota* en el sistema de ficheros?:

- a) Controlar el acceso malintencionado.
- b) Rastrear un posible virus que exista.
- c) Controlar el espacio de almacenamiento a los usuarios que posean carpetas en nuestro servidor.
- d) Ninguna de las respuestas anteriores es correcta.

9. ¿Qué significa en binario 110 para darle permisos efectivos a un fichero en Linux?:

- a) --x.
- b) rw-.
- c) rwx.
- d) r-x.

10. ¿Se puede validar un usuario de FTP en LDAP?:

- a) Depende del equipo.
- b) Depende del sistema operativo.
- c) Sí.
- d) Sí, pero realizando la configuración adecuada.

### SOLUCIONES:

1. **a b c d**  
2. **a b c d**  
3. **a b c d**  
4. **a b c d**

5. **a b c d**  
6. **a b c d**  
7. **a b c d**  
8. **a b c d**

9. **a b c d**  
10. **a b c d**

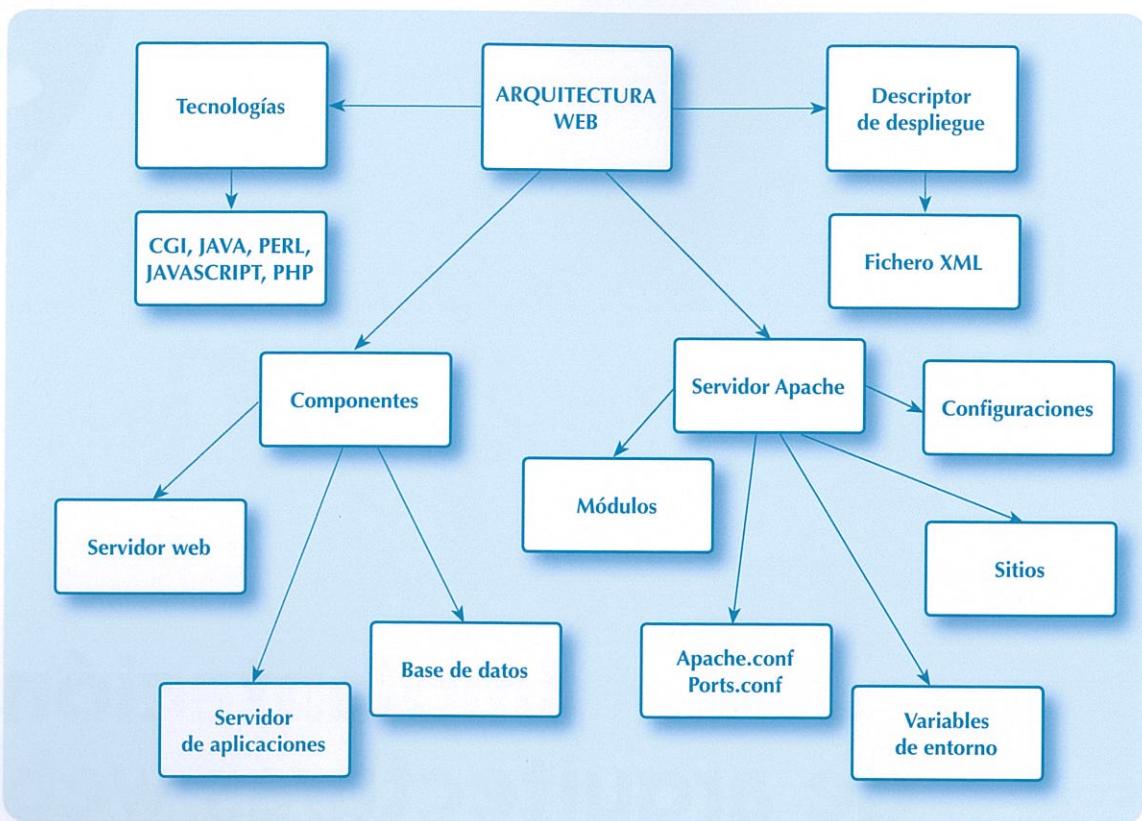


# Implantación de arquitecturas web

## Objetivos

- ✓ Analizar aspectos generales de arquitecturas web, sus características, ventajas e inconvenientes.
- ✓ Describir los fundamentos y protocolos en los que se basa el funcionamiento de un servidor web.
- ✓ Instalar y configurar los servidores web.
- ✓ Clasificar y describir los principales servidores de aplicaciones.
- ✓ Instalar y configurar los servidores de aplicaciones.
- ✓ Realizar pruebas de funcionamiento de los servidores web y de aplicaciones.
- ✓ Analizar la estructura y recursos que componen una aplicación web.
- ✓ Describir los requerimientos del proceso de implantación de una aplicación web.

## Mapa conceptual



## Glosario

**Bases de datos.** Es un conjunto de información estructurada de tal forma que se organiza para ser accedida, consultada y modificada tal información. Están basadas en un modelo, por ejemplo, modelo relacional, modelo en red, modelo de objetos, etc.

**Directiva.** Es una palabra reservada dentro de un fichero de configuración que va acompañada de parámetros para realizar una función específica.

**Encapsulamiento.** En programación permite ocultar el estado de cada uno de los módulos y permite localizar los errores más fácilmente.

**Fichero XML.** Es un fichero que se caracteriza por un conjunto de marcas rodeado de información, de tal forma que esta sea consultada, modificada y eliminada. Se basa en el conjunto de SGML.

**HTTP.** Protocolo a nivel de aplicación que permite la visualización de información y la transferencia de información en la World Wide Web.

**Plataforma.** Es un conjunto de software que persigue un objetivo común, que es darle una solución completa al programador, por ejemplo LAMP (Linux, Apache, MySQL y PHP).

**Linux.** Sistema operativo que es multitarea, multiplataforma y multiusuario, que permite usar programas como editores de texto, navegadores, programas ofimáticos, etc. Se puede usar mediante comandos o de forma gráfica.

**Variables de entorno.** Son aquellas variables que es necesario configurar en cualquier servidor, sistema operativo o dispositivo que sea configurable.

### 3.1. Introducción

La arquitectura web es la que define cómo se va a jerarquizar la información dentro de un sitio web de forma racional y lógica.

En el mercado actual existen una serie de tecnologías que desarrollar tanto a nivel de cliente como de servidor, la elección va a depender de la dimensión y el coste del proyecto.

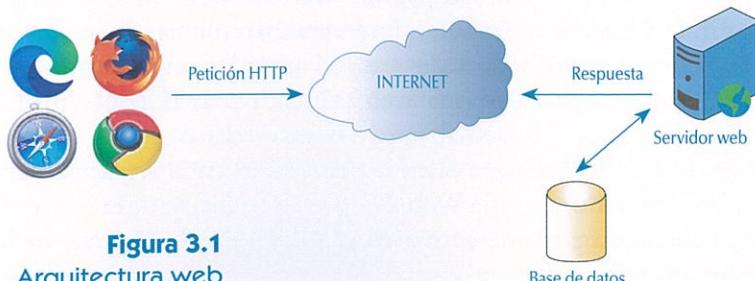
Su objetivo es la agrupación visual de información en un sitio web, que permitirá el acceso intuitivo y la navegabilidad fácil de cara a la empresa y al usuario.

### 3.2. Arquitecturas web

Actualmente, para que una aplicación funcione correctamente tiene tres elementos principales que permiten la conexión y el acceso a datos por parte de cualquier petición de un cliente.

Por ello se van a comentar estos tres componentes:

- Servidor web:* es el servidor o cerebro de la arquitectura, donde está escuchando las peticiones HTTP desde el navegador. También realiza consultas a la base de datos para responder a las peticiones.
- Base de datos:* es un conjunto de datos organizados jerárquicamente al servicio del servidor web.
- Cliente web:* es el que realiza las peticiones al servidor web mediante un navegador y desde un sistema operativo concreto que es independiente de la arquitectura.



**Figura 3.1**  
Arquitectura web

### 3.3. Evolución de la tecnología web

Desde que comenzó la era de la informática, la tecnología ha ido evolucionando de forma exponencial hasta nuestros días. Y en el contexto de la web no iba a ser menos, ya que está estrechamente ligado a la tecnología de Internet. Tal contexto presta sus servicios para que las arquitecturas que existen en cada empresa o particular continúen funcionando y avanzando gracias a las investigaciones de todo el mundo.

Se van a nombrar algunos de los componentes de la web que han ido evolucionando:

- ✓ *Ancho de banda*: al principio el hardware de comunicaciones era muy costoso y poco veloz, en nuestros días es más económico y tiene más velocidad.
- ✓ *Almacenamiento*: actualmente se manejan cantidades ingentes de información que se pueden almacenar sin problemas gracias a las investigaciones y los dispositivos de almacenamiento actuales.
- ✓ *Información*: en el pasado la información era mucho más estática, ahora se convierte en bastante más dinámica y existe más interacción con el usuario.
- ✓ *Computación*: los procesadores, memoria y disco duro actuales permiten que las operaciones se realicen bastante más rápido que en el pasado.
- ✓ *Tecnología*: hoy en día existe un amplio abanico de posibilidades que se pueden implementar tanto en el cliente como en el servidor; antes existían bastantes menos. Por lo tanto, la capacidad para desarrollar una aplicación era infinitamente menor.
- ✓ *Infraestructuras*: este factor es de los más importantes, ya que engloba todo lo anterior. Hoy en día, existe alta disponibilidad, backup remoto, duplicidad de nodos en caso de fallo hardware y software, y una variedad de posibilidades que antes no existían por el avance de la tecnología.

#### RECUERDA

- ✓ Los componentes de una web son fundamentales a la hora de diseñar una arquitectura web.

A principios de los tiempos de la informática, el contenido de la web era estático y no existían protocolos de seguridad que permitieran la fiabilidad de las conexiones. A día de hoy, esto sería impensable, ya que es un factor importante para tener en cuenta en cualquier organización. Esta primera fase, en la que no existía interacción con el usuario y las páginas web eran estáticas y básicamente el lenguaje de marcas HTML, se denominó Web 1.0. En 1991 Tim Berners-Lee fue el creador de la primera página web con contenido estático, donde existían marcas y botones gif. Los usuarios solamente interactuaban mediante email, foros y chats.

A partir de aquí, los desarrolladores de todo el mundo empezaron a investigar sobre la posibilidad de ampliar esta primera página web e incluir más funcionalidades y las primeras aplicaciones web dinámicas que interactuaban con bases de datos. Algunos autores asignan a este nuevo descubrimiento Web 1.5, aunque otros la mantienen en la versión anterior.

La verdadera evolución vino con la Web 2.0 o web social, ya que se pasó a la interacción total con el usuario y un componente activo en el modelo web. Algunos de los ítems importantes fueron los siguientes:

- Hojas de estilo CSS que dan vistosidad a las webs.
- Uso de JSON.
- Desarrollo en Ajax.
- Soporte para los blogs.
- Comienzo de las redes sociales.
- Control total de los usuarios en el manejo de información.

El siguiente avance significativo fue la aparición de la Web 3.0 o data web, también llamada *web semántica*. Este gran salto supuso un avance tecnológico hacia la inteligencia artificial. Algunos de los grandes avances son los siguientes:

- Diseño responsive.
- Web multimedia.
- Aplicaciones inteligentes.
- Web semántica más inteligencia artificial.
- Impulso a la Web 3D.
- Participación más activa en la red.
- CCS3.

Por último, tras la incorporación en el cine de la tecnología 3D, en CCS3 ya se puede realizar algo en tercera dimensión. HTML5 comienza a incorporar importantes avances en este sentido.

## 3.4. Tecnologías usadas en aplicaciones web

Actualmente, la mayoría de las aplicaciones web del mercado usan páginas dinámicas, que se ejecutan en el servidor web y se visualizan en el cliente (navegador). Como se comentó anteriormente, existen páginas de contenido estático y dinámico; normalmente, cuando existe contenido dinámico se ejecuta código tanto en el cliente como en el servidor. Se va a diferenciar el código que se ejecuta en estos dos nodos:

### 3.4.1. En el lado servidor

1. *CGI (Common Gateway Interface)*: al principio de Internet, el servidor solamente podía ejecutar programas del tipo C, Perl y líneas de comando Powershell. Estas instrucciones eran ejecutadas por el sistema operativo y se transmitían al navegador mediante el CGI.
2. *ASP.NET (Active Server Pages)*: es un framework de desarrollo libre de Windows orientado a objetos. Aunque existen versiones para Linux y Unix.
3. *Java*: existe un gran grupo de tecnologías desarrollado por Sun Microsystems que se pueden ejecutar en el servidor, y algunas son JSF (JavaServer Faces), JSP (JavaServer Pages) y los servlets. De estas tres, la que más se usa es JSP, ya que es la más extendida y es compatible con casi todos los servidores web.
4. *Ruby*: es un lenguaje interpretado de propósito general, también es dinámico y flexible. Además es de alto nivel, y lo más importante es que es software libre y multiplataforma.
5. *Perl*: generalmente este lenguaje de programación se ejecuta en el servidor. Apache, por ejemplo, tiene un módulo que permite ejecutar programas de este tipo. Toma

muchas características del lenguaje C y se caracteriza por su destreza a la hora de procesar texto.

6. *PHP*: es un lenguaje de propósito general del desarrollo backend más usado y popular que existe en el mercado. Es útil en el desarrollo de aplicaciones y motor de Wordpress, Drupal, Magento, Joomla, etc. Además, existen frameworks muy potentes como Laravel o Symfony. Por último, una de las mejores ventajas es que accede fácilmente a bases de datos como Oracle o Mysql, y también a bases de datos NoSQL como MongoDB. PHP tiene un inconveniente, ya que en algunas funcionalidades tiene agujeros de seguridad si no se toman las medidas oportunas.
7. *Python*: es un lenguaje interpretado muy poderoso, fácil de aprender y de alto nivel. Se usa en multitud de aplicaciones y sobre todo en aplicaciones de seguridad. Es un lenguaje orientado a objetos, que destaca por su tipado dinámico.
8. *Javascript*: es un lenguaje ligero, interpretado y orientado a objetos. Para aprenderlo es preciso un conocimiento básico de HTML y CSS. Interviene sobre todo en el contenido dinámico y en la interacción con el usuario. Permite controlar archivos de multimedia, creación de imágenes animadas, animación en 3D. Destaca por su aportación a la mayoría de los ámbitos de la tecnología.



#### PARA SABER MÁS

El lenguaje R es uno de los más usados en tecnología web debido al crecimiento exponencial del Big Data.

### 3.4.2. En el lado cliente

1. *HTML y CSS*: HTML es el lenguaje de marcas que se inventó junto con el navegador y el medio para transmitir información entre el cliente y el servidor. Posteriormente, surgió CSS (*Cascading Style Sheets*) que permitirá diseñar gráficamente la página HTML como tal. Posee una especificación que parte del W3C. Comenzó en la versión 1 y ya se ha publicado la versión 3.
2. *Javascript*: comentado anteriormente, se puede usar tanto en el cliente como en el servidor. Es uno de los más extendidos porque es soportado por la mayoría de los navegadores.
3. *VBScript*: es la competencia de Java que hizo Microsoft. Este lenguaje ha entrado en desuso, ya que no es compatible con la mayoría de los navegadores, solamente con Internet Explorer. Es un lenguaje interpretado por Windows Scripting Host de Microsoft. Se basa en el lenguaje de programación Visual Basic.

### 3.4.3. En ambos

1. *DHTML*: no es un lenguaje de programación como tal sino el conjunto de todos los elementos de una página, como, por ejemplo, el fondo, posiciones de controles, cajas, contenido dinámico, etc. Se ejecuta tanto en el servidor como en el cliente.

2. **XML:** es una tecnología relacionada con lenguajes de marca, y que permite compartir datos, incluso formar parte de una base de datos. Estos ficheros .xml se centran en la configuración de la mayoría de las aplicaciones y de los servidores web y de aplicaciones.



### Actividad propuesta 3.1

Describe en un cuadro cuáles son las tecnologías empleadas en el servidor y en el cliente.

## 3.5. Servidores y aplicaciones libres y propietarias

En la actualidad, existe una serie de plataformas y servidores libres para poder programar sobre cualquier lenguaje de los vistos anteriormente, tanto en el cliente como en el servidor. Por otro lado, en menor medida, pero también disponibles, están los servidores propietarios en los que se puede programar cualquier aplicación con base en cualquier lenguaje.

En esta sección se van a nombrar las más representativas plataformas en las que los desarrolladores programan sus aplicaciones. Para ello, se van a definir los componentes que forman la arquitectura web:

- a) *Sistema operativo:* es fundamental porque es el software base sobre el cual se sustentan los demás componentes. Los principales son Windows, Linux, Unix y Mac.
- b) *Servidor web:* es el servicio que va a atender las peticiones de los clientes y les va a responder lo más pronto posible. Principalmente, existen dos tipos de páginas, estáticas y dinámicas.
- c) *Bases de datos:* es la principal fuente de información de la que se nutre el servidor web para mostrar al cliente la información que solicita. Este componente es básico en cualquier portal de hoy en día. Existen dos grandes grupos, SQL y NoSQL.
- d) *Lenguaje de programación:* es el lenguaje que se encarga de la lógica de la aplicación. Tiene como función recibir las peticiones de los clientes, consultar la información en la base de datos y responder a las peticiones. Dependiendo de la aplicación, se pueden usar varios lenguajes de programación en la aplicación delimitando sus funciones. La clave de que funcione todo correctamente es el encapsulamiento y la modulación de los componentes de la aplicación.

Teniendo en cuenta los cuatro componentes anteriores, existen dos plataformas clave para trabajar en el desarrollo de una aplicación. Se denominan LAMP y WISA, y son conocidas por su popularidad y su uso en gran parte del entorno productivo empresarial.

La plataforma LAMP es de software libre e incluye los cuatro componentes descritos anteriormente, de ahí su popularidad y su crecimiento exponencial en el desarrollo de aplicaciones. Sus iniciales provienen de los siguientes componentes:

- Linux: sistema operativo.
- Apache: servidor web.

- MySQL: base de datos.
- PHP: lenguaje de programación, aunque se pueden usar otros como Python.

La plataforma WISA es de software propietario e incluye los cuatro componentes descritos anteriormente, de ahí su popularidad y su crecimiento exponencial en el desarrollo de aplicaciones. Sus iniciales provienen de los siguientes componentes:

- Windows: sistema operativo.
- IIS (*Internet Information Services*): servidor web.
- SQL: base de datos.
- ASP: lenguaje de script en el servidor.

### Actividad propuesta 3.2



Instala una plataforma en Windows y otra en Linux, e intenta navegar y configurar correctamente en las dos plataformas.

Existen algunas más, que es necesario mencionar y que actualmente se usan en el módulo DAW de los ciclos de informática y en las empresas de programación de aplicaciones web, como es XAMPP y Scotch Box.

La plataforma XAMPP es una distribución de Apache sin coste, multiplataforma y muy fácil de instalar. Sus iniciales son las siguientes:

- X: cualquier sistema operativo.
- Apache: servidor web.
- MariaDB/MySQL: base de datos.
- Perl, PHP: lenguaje de código abierto del servidor.

WWW

### Recurso web

1. Página de Apache Friends donde podrás saber más y descargar XAMPP.
2. Página de Scotch Box 3.5 en la que podrás profundizar más sobre esta distribución de LAMP.



1.



2.

La plataforma Scotch Box es una distribución de LAMP, multiplataforma, muy fácil de instalar. Entre los componentes, están los siguientes:

- Ubuntu 17.10.
- PHP 7.2.
- Apache 2.4.29.
- Nginx 1.13.8.
- RVM 2.5.0.
- NVM 8.94.

### 3.6. Protocolo HTTP vs HTTPS

Como se ha comentado anteriormente, el protocolo por antonomasia es HTTP y el seguro es HTTPS. En este apartado se entrará en profundidad en cada uno de ellos.

HTTP es un protocolo de la capa de aplicación que escucha por el puerto 80 basado en el modelo cliente-servidor. Se basa en TCP, por lo tanto, es orientado a conexión y escucha las peticiones de los clientes. Una vez aceptada la conexión, se encarga de mantenerla y garantizar la transferencia de datos sin errores.

Por otro lado, HTTPS es un protocolo de la capa de aplicación que escucha por el puerto 443, y también se basa en el modelo cliente-servidor, pero en modo seguro. Se basa en TCP, es orientado a conexión y encripta los datos para asegurar una transmisión segura entre los dos extremos. Para realizar esta encriptación necesita de certificados, siempre y cuando sean válidos tales certificados. Sí es cierto que al cifrar la información en el servidor y descifrarla en el cliente se pierde tiempo de computación. Aunque solamente se necesite cifrar cierta información.

Cualquier servidor web medio serio, por ejemplo, Apache, posee la acción de emitir certificados. En este entorno, la parte fundamental son los navegadores, ya que deben validar dichos certificados a partir de entidades certificadoras. Es necesario tener una entidad que sea fiable, por ejemplo, la FNMT que es la encargada en España de suministrar los certificados a los ciudadanos.

El protocolo HTTPS usa la encriptación SSL (*Secure Socket Layer*) que encripta datos sensibles (actualmente se le llama TLS). Un certificado SSL o TLS se instala en el servidor; la indicación de que es seguro es porque a http le sigue una s y se transforma en https://.

Se va a explicar a continuación cómo funciona el protocolo HTTPS cuando se teclea en el navegador:

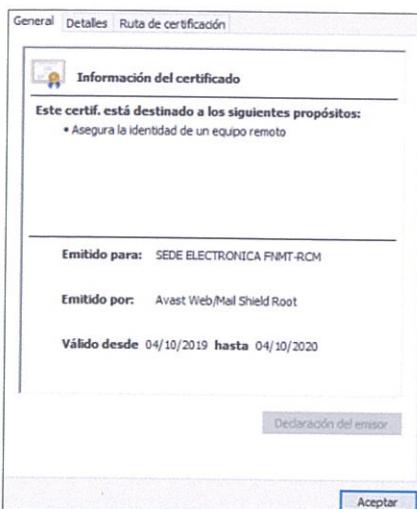
1. El usuario teclea una dirección web en el navegador https://www.sede.fnmt.gob.es/.
2. Se traduce el dominio DNS a la IP correspondiente.
3. Se busca en el servidor web la IP de la página solicitada por el puerto TCP 443.
4. Antes de transferir la información al navegador del cliente, se negocia mediante TLS, que consiste en el envío del certificado al cliente, y el cliente debe aceptar este certificado. Una vez aceptado, se usará un canal cifrado con la información.
5. Por último, se realiza la petición HTTP y se envía la respuesta al cliente.



#### Actividad propuesta 3.3

Navega por Internet y teclea una página que posea el protocolo HTTPS. Observa el certificado que posee esa página web, cualquier banco de España puede ser un ejemplo.

Como se muestra en la figura 3.2, el certificado es válido.



**Figura 3.2**  
Certificado

### 3.7. Clasificación de servidores de aplicaciones del mercado actual

En el mercado actual tan cambiante, existen multitud de servidores preparados para ser instalados en cualquier plataforma. A continuación, se mostrarán algunos de los servidores web más usados, que permiten ejecutar en su entorno diferentes tecnologías:

- *Apache*: es uno de los servidores más usados en Internet, su nombre completo es Apache HTTP Server. Su primera versión fue en 1996, desde entonces ha sido un referente como servidor, pero tiene dos grandes competidores que son Nginx y Microsoft IIS. Tiene como ventaja que su desarrollo es código abierto, y es multiplataforma. Entre sus desventajas se encuentra su bajo rendimiento cuando se realizan multitud de peticiones por parte de los clientes.
- *Mac OS X Server*: es un servidor para Mac que incluye bastantes servicios, entre otros un servidor Apache que permite las conexiones desde clientes para dar servicio de páginas web. Además incluye la base de datos Mysql, servicio NAT, servidor DHCP, servidor VPN, PHP, y algunos más.
- *Nginx*: es un servidor de código abierto y gratuito que nació en 2004, y es uno de los más usados por las compañías en el mundo. Destaca por su alto rendimiento e incluye funciones como servidor proxy inverso HTTP, balanceador de carga, POP3 e IMAP. Se puede instalar en Windows, Linux y Unix. Es altamente escalable, por si fuera necesaria una ampliación, y consume muy pocos recursos para resolver las peticiones de los clientes.
- *LiteSpeed*: es menos usado que el anterior, pero es un software para aceptar peticiones mediante HTTP enfocado a entornos Linux. Soporta grandes cantidades de conexiones simultáneas, está a la altura de Nginx. Incluye un sistema de caché para aplicaciones como Magento y Wordpress. El único inconveniente es que la versión completa es de pago, pero es una magnífica alternativa a Nginx.

- *Microsoft IIS*: es el software de Microsoft conocido por Internet Information Services. Comenzó en Windows NT. Permite el procesamiento de páginas del tipo ASP, ASP.net, PHP o Perl. Está programado en C++, pero tiene una gran desventaja y es que solamente se puede instalar en Windows, con lo que conlleva una amenaza de seguridad constante. Además posee servicios como FTP y SMTP.
- *GWS*: es conocido como Google Web Server. Este servidor es un poco peculiar, ya que no es público y no se puede descargar desde ninguna URL. Este servidor es solo de Google, y se puede comprobar con el comando curl.
- *Sun Java System Web Server*: es un servidor orientado a la programación en Java con licencia BSD, que incluye entre otras características soporte para WebDA, JDBC, PHP... Es multiplataforma; salvo en Mac, en los demás sistemas operativos se puede instalar perfectamente.



### Actividad propuesta 3.4

Instala el servidor web Nginx, configura los parámetros más habituales y observa cómo se comporta tal servidor.

## 3.8. Instalación y configuración básica del servidor Apache

Dos de los servidores más usados por las grandes compañías y organismos son Apache e IIS, por ello vamos a tratarlos más en profundidad en Windows y Linux. De estos servidores se van a mostrar las instalaciones en Linux y Windows junto con la configuración para que funcionen correctamente.

### 3.8.1. Definición y características

Actualmente no nos imaginamos una aplicación web sin un servidor web, por ello se va a tratar en este apartado. Primero es necesario dimensionar los accesos que van a solicitarse por parte de los clientes. Por lo que sería bueno configurar el hardware necesario para que no existan cortes en el servicio o congestionamiento de la red.

#### TOMA NOTA



En esta fase de diseño es crucial prever la escalabilidad del servidor web que se va a configurar y, por otro lado, que también sea estable.

Apache HTTP Server es un software gratuito y de código abierto para plataformas Linux o Unix y Windows. Es desarrollado y mantenido por Apache Software Foundation. Las características de este servidor serían las siguientes:

- ✓ Autenticación y autorización.
- ✓ Funcionalidad de proxy.
- ✓ Filtrado inteligente.
- ✓ Hosts virtuales.
- ✓ Modularización.
- ✓ Seguridad mediante cifrado.
- ✓ Monitorización mediante archivos de logs.
- ✓ Páginas web estáticas y dinámicas.

### 3.8.2. Instalación y configuración del servidor Apache en Debian

La instalación es del servidor Apache llamado apache2 para SO Linux bajo la versión de Debian (Kali Linux). La instalación se puede realizar en una máquina virtual que posee una imagen de este sistema operativo o en una máquina física. La lista de directivas de apache2 es bastante amplia, se van a configurar las más importantes. El procedimiento de instalación es el siguiente:

1. Para comenzar, habría que tener conexión a Internet y actualizados los repositorios de la máquina virtual o física que vayamos a instalar el servidor web apache2, además de poner una dirección IP fija para que no haya problemas de conexión en un momento posterior. Una vez realizadas todas las acciones anteriores (comentadas en el servicio de DNS) se está en disposición de ejecutar la siguiente sentencia:

```
# apt-get install apache2
```

Un ejemplo de la ejecución del comando sería la de la figura 3.3. Como se puede observar, se van a instalar los módulos apache2-bin, apache2-data y apache2-utils.

```
root@osboxes: # apt-get install apache2
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
Los paquetes indicados a continuación se instalaron de forma automática y ya no
son necesarios.
  finger libboost-atomic1.62.0 libboost-chrono1.62.0
  libboost-program-options1.62.0 libboost-serialization1.62.0
  libboost-test1.62.0 libboost-timer1.62.0 libcgal12 libfcgi-bin libfcgi0ldbl
  liblwgeom-2.4-0 liblwgeom-dev libqca2 libqca2-plugins
  libqgis-analysis2.14.20 libqgis-core2.14.20 libqgis-customwidgets
  libqgis-gui2.14.20 libqgis-networkanalysis2.14.20 libqgis-server2.14.20
  libqgispython2.14.20 libqtwebkit libqwt6abil libsfrcgalib libspatialindex4v5
  python-cycler python-elixir python-functools32 python-matplotlib
  python-pyspatialite python-qgis python-qgis-common python-qt4-sql
  python-shapely python-subprocess32 qt4-designer rwho rwhod x11-apps xsllib
Utilice «apt autoremove» para eliminarlos.
Se instalarán los siguientes paquetes adicionales:
  apache2-bin apache2-data apache2-utils
Paquetes sugeridos:
  apache2-doc apache2-suexec-pristine | apache2-suexec-custom
Se instalarán los siguientes paquetes NUEVOS:
  apache2 apache2-data apache2-utils
```

**Figura 3.3**  
Instalación apache2

2. Despues, se inicia el servicio con el comando **service apache2 start** y, si todo es correcto, se puede comprobar con los siguientes comandos si el servicio está activo y está escuchando peticiones:

```
#service apache2 status
#ps -aux | grep "apache2"
#nmap -ST -O localhost (comprobación de escucha por el puerto 80)
```

```
root@osboxes:/etc# nmap -ST -O localhost
Starting Nmap 7.60 ( https://nmap.org ) at 2020-04-09 12:43 EDT
Nmap scan report for localhost (127.0.0.1)
Host is up (0.000076s latency).
Other addresses for localhost (not scanned): ::1
Not shown: 999 closed ports
PORT      STATE SERVICE
80/tcp    open  http
Device type: general purpose
Running: Linux 3.X|4.X
OS CPE: cpe:/o:linux:linux_kernel:3 cpe:/o:linux:linux_kernel:4
OS details: Linux 3.8 - 4.9
Network Distance: 0 hops
```

**Figura 3.4**  
Comprobación del servicio apache2

- Antes de continuar con la configuración de apache2, es necesario comentar algunos comandos útiles con relación al funcionamiento de este servidor en Linux:

- *apachectl start*: inicia el servicio de apache2.
- *apachectl stop*: para el servicio de apache2.
- *apachectl restart*: reinicia el servicio apache2.
- *apachectl status*: muestra el estado del servicio apache2.
- *apachectl graceful*: reinicia el servicio apache2 pero las conexiones abiertas las mantiene sin cerrarlas.
- *apachectl configtest*: ejecuta un test sintáctico para comprobar que los ficheros de configuración son correctos.



### Actividad propuesta 3.5

Instala el servicio Apache y practica con los comandos anteriores para ir asimilando la herramienta.

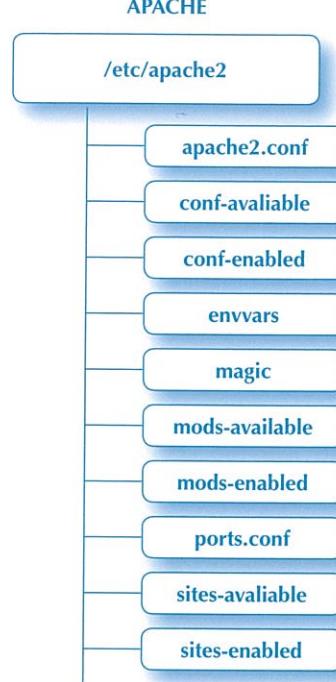
- A continuación, se va a analizar la estructura que tiene apache2 cuando se instala, que es la que se puede ver en la figura 3.5. Se explicarán cada uno de los directorios y ficheros de configuración que posee apache2 y se realizarán las configuraciones básicas para que se observe cómo trabaja este servidor. Además, se darán unas nociones básicas de seguridad para que minimicemos la amenaza desde el exterior. Se comenta la estructura de la forma siguiente:

- *apache2.conf*: es el fichero de configuración principal del servidor apache2. Contiene las variables globales. Cualquier cambio en este fichero implicaría reiniciar el servicio.
- *conf-available*: este directorio contiene configuraciones adicionales que están asociadas a un módulo en particular. Las configuraciones no están activas.

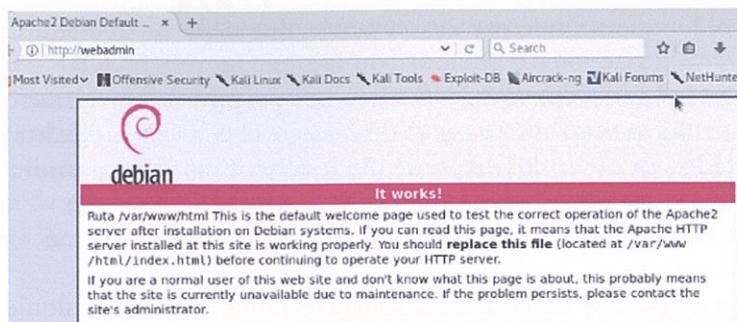
- *conf-enabled*: este directorio contiene enlaces al directorio anterior para poder activar las configuraciones que contiene.
  - *envvars*: es un fichero en el que se definen variables de entorno como APACHE\_RUN\_USER, APACHE\_RUN\_GROUP, etc., que en principio no es necesario modificar.
  - *magic*: es un fichero de configuración de tipo MIME, que permite configurar el tipo de medio del contenido que va a visualizar en el servidor web, puede ser del correo electrónico o de los servidores web. Algunos ejemplos pueden ser:
    - audio/x-mod.
    - Application/java
    - Image/x-coreldraw
  - *mods-available*, *mods-enabled*: estos directorios son similares a los sites, pero para módulos que se pueden acoplar al servidor web.
  - *ports.conf*: este fichero siempre está incluido en el fichero apache2.conf y permite configurar los puertos y las IP por las que escucha el servidor. El puerto por defecto es el 80.
  - *sites-available*: este directorio contiene los ficheros de los hosts virtuales definidos en el servidor, que pueden ser diferentes. Estos sitios están disponibles pero no activos.
  - *sites-enabled*: este directorio es similar al anterior pero las definiciones de hosts virtuales que se están usando, normalmente son enlaces simbólicos a los ficheros que se encuentran en sites-available. Por defecto, tiene el directorio /var/www/html, que es donde se almacena la página principal de Apache.
5. Se va a configurar la página de inicio de apache2, que cuando se instala el servicio se encuentra en /var/www/html/index.html. Cuando se teclea en el navegador http://localhost o http://10.0.2.15 (IP del equipo) o http://webadmin (creada en el fichero hosts previamente) es la página que se va a mostrar. Se va a modificar el fichero index.html para saber que estamos en ese directorio y luego se configurará en otro directorio. Cuando se inicia el servicio con el comando siguiente, se obtiene la página de inicio (figura 3.6).

```
#service apache2 start
```

Este apartado está relacionado con dos ficheros de configuración, que son 000-default.conf y apache2.conf. La configuración de ambos es lo que permite el acceso de un usuario desde el exterior y siempre que esté autorizado en caso de que existiera un firewall. Por ello, se recomienda configurar una DMZ pública y configurar el servidor, separándolo de nuestra Intranet en caso de una organización. Se va a comentar el fichero 000-default.conf (figura 3.7).



**Figura 3.5**  
Estructura de directorios apache2



**Figura 3.6**  
Página de inicio del servidor Apache

```
GNU nano 2.8.7           Fichero: 000-default.conf

<VirtualHost *:80>
    ServerAdmin webmaster@localhost
    DocumentRoot /var/www/html

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

**Figura 3.7**  
Fichero virtual host por defecto

Este fichero es un sitio virtual, que se configura en el servidor web, donde posee varias directivas pero la más importante es DocumentRoot. Esta directiva permite configurar una ruta donde se ubicará el fichero index.html. Para ello se ha creado un host virtual que escucha por el puerto 80 con la directiva <VirtualHost \*:80>.

Para habilitar el sitio es necesario usar el comando *a2ensite* y para deshabilitarlo es el comando *a2dissite*. Aunque los hosts virtuales se tratarán en el próximo tema, es necesario comentarlo aquí para la configuración básica de Apache. La otra directiva relacionada con esta configuración es en el fichero apache2.conf (figura 3.8).

El primer bloque (<Directory />) deniega todos los accesos, esto se realiza como método de seguridad para que nadie acceda al sistema de ficheros del sistema operativo. Se queda por defecto esta parte, que tiene las directivas:

- *Options FollowSymLinks*: que permite los enlaces simbólicos a otros directorios.
- *AllowOverride None*: ignora el fichero .htaccess para que no existan demasiadas llamadas a este fichero.
- *Require all denied*: deniega el permiso a todo el sistema de ficheros.

```
<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>
```

**Figura 3.8**  
Fichero apache2.conf

Y en el bloque <Directory /var/www>, que tiene las siguientes directivas:

- *Options Indexes FollowSymLinks*: Indexes permite la visualización del directorio en caso de que no exista en el directorio el fichero index.html o los patrones declarados en el servidor Apache. FollowSymLinks que permite los enlaces simbólicos a otros directorios.
  - *AllowOverride None*: ignora el fichero .htaccess para que no existan demasiadas llamadas a este.
  - *Require all granted*: da permiso a todo el mundo y desde cualquier IP a directorio /var/www.
6. Se va a configurar un sitio distinto que se llamará sitio.conf. Este fichero va a tener una ruta distinta para ubicar el fichero index.html que es /home/web/www. Para ello se van a modificar los dos ficheros anteriores, configuración que sería como se muestra en las figuras 3.9 y 3.10.

```
<Directory /home/web/www>
    Options Indexes FollowSymLinks
    AllowOverride None
    Require all granted
</Directory>
```

**Figura 3.9**  
Fichero apache2.conf II

```
GNU nano 2.8.7                               Fichero: sitio.conf
<VirtualHost *:80>
    ServerAdmin webmaster@localhost
    DocumentRoot /home/web/www

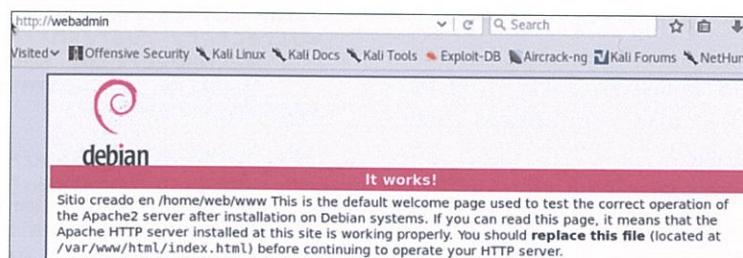
    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

**Figura 3.10**  
Fichero sitio.conf

Una vez modificados tales ficheros, se realizan los siguientes comandos: deshabilitar el sitio 000-default.conf y habilitar el sitio sitio.conf. Por último, cargamos de nuevo la configuración de Apache:

```
#a2dissite 000-default
#a2ensite sitio
#service apache2 reload
```

Y se obtendrá la pantalla de la figura 3.11, en la que se puede comprobar que funciona correctamente el sitio creado:



**Figura 3.11**  
Comprobación del sitio

7. Se va a configurar que el servidor escuche por otro puerto. Para ello se configurará en el fichero ports.conf, por ejemplo, el puerto 8080 (figura 3.12).

Si se observa el fichero, se ha creado la línea Listen 8080, y ya escucharía el servidor por el puerto 8080. Pero se necesitaría cambiar o modificar el fichero de sitio para que escuche por tal puerto, si no el servidor web daría un error de acceso prohibido. Si se configura de esta forma el fichero sitio.conf, el servidor escucharía por los dos puertos (figura 3.13).

```
Listen 80
Listen 8080

<IfModule ssl_module>
    Listen 443
</IfModule>

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

**Figura 3.12**  
Fichero ports.conf

```
<VirtualHost *:80>
    ServerAdmin webmaster@localhost
    DocumentRoot /home/web/www

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

<VirtualHost *:8080>
    ServerAdmin webmaster@localhost
    DocumentRoot /home/web/www

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

**Figura 3.13**  
Fichero sitio.conf

Con esto quedaría la configuración funcionando, después de reiniciar el servicio Apache de la siguiente forma:

```
#service apache2 restart
```

8. Por último, se van a detallar las directivas de configuración del fichero principal de Apache, apache2.conf:

- *ServerRoot*: es el directorio raíz donde está instalado el servidor Apache. Por defecto es /etc/apache2.
- *ServerName*: el nombre DNS que indica el nombre del servidor web, que en nuestro caso es webadmin. Sería necesario registrarla en nuestro DNS o, en su defecto, en el fichero /etc/hosts. Se puede indicar también en el fichero de host virtual.
- *ServerAdmin*: es el correo electrónico del administrador del servidor web, por si es necesario contactar con él.
- *Timeout*: permite especificar el tiempo que espera Apache para cerrar las conexiones con un cliente determinado.
- *KeepAlive*: si está activado permitirá las conexiones persistentes, es decir, si el servidor web deja abierta la conexión con un cliente para futuras conexiones. Tiene dos valores, on y off.
- *MaxKeepAliveRequests*: delimita el número de peticiones permitidas por una conexión persistente. Por defecto son 100; si se quiere poner infinito se puede poner 0, pero no es recomendable por rendimiento y seguridad.
- *KeepAliveTimeout*: el número de segundos que el servidor esperará para la próxima petición del mismo cliente.

- *HostnameLookups*: el servidor resuelve automáticamente la IP de cada conexión. Esto provoca que el servidor realice consultas al DNS para ejecutar la operación.
- *LogLevel*: especifica en el log el nombre DNS de los clientes o la IP de los mismos. Por defecto es off, y es la mejor opción para obtener la trazabilidad de las conexiones de los clientes.
- *Include*: esta directiva especifica la inclusión de otros ficheros de configuración, como puede ser ports.conf.
- *AccessFileName*: permite incluir un fichero con adicionales directivas de configuración.
- *IncludeOptional*: incluye otros ficheros de configuración, como pueden ser hosts virtuales, etc.



#### PARA SABER MÁS

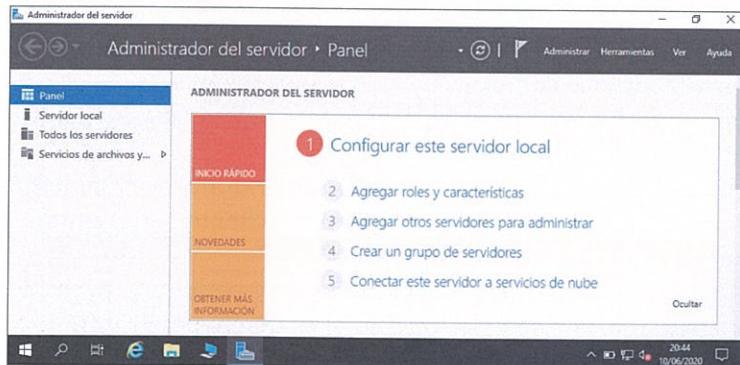
Se recomienda consultar este enlace de la página web de Apache para ampliar la información sobre las directivas.



### 3.8.3. Instalación y configuración del servidor IIS en Windows Server

La instalación es sobre el servicio web llamado IIS para SO Windows Server 2019. La instalación se puede realizar en una máquina virtual que posee una imagen de este sistema operativo o en una máquina física. Se configurará de forma simple para que se observe el funcionamiento de este servicio. El procedimiento de instalación es el siguiente:

1. Para comenzar, habría que configurar una dirección IP fija para que no haya problemas de conexión en un momento posterior. Una vez realizada la acción anterior, es necesario pulsar el inicio de Windows y, posteriormente, seleccionar Administrador del servidor. Se visualizará la figura 3.14.



**Figura 3.14**  
Administrador del servidor

2. A continuación, se pulsará en *Agregar roles y características*, y se observará una pantalla como la de la figura 3.15.
3. Se pulsará en Siguiente, y se observará una pantalla como la de la figura 3.16.

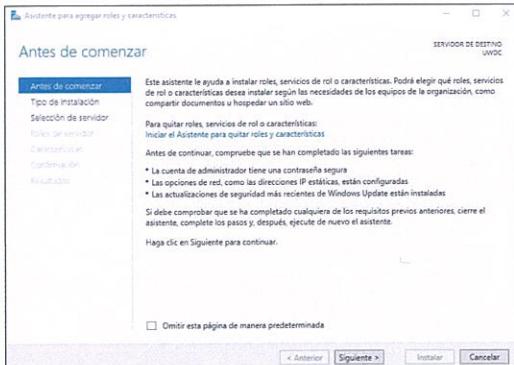


Figura 3.15  
Roles

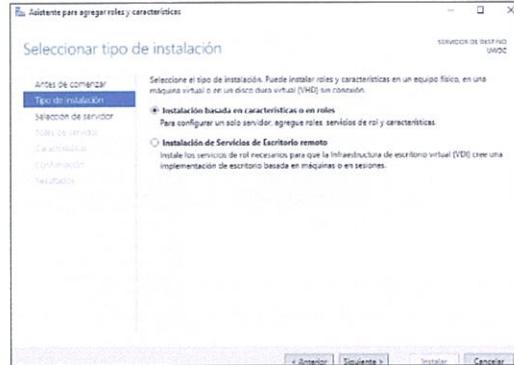


Figura 3.16  
Tipo de instalación

4. Se pulsará en Siguiente y se observará una pantalla como la de la figura 3.17.
5. En la figura 3.17 se selecciona el servidor (UWDC en nuestro caso) y se pulsa Siguiente. Se mostrará una imagen como la de la figura 3.18.

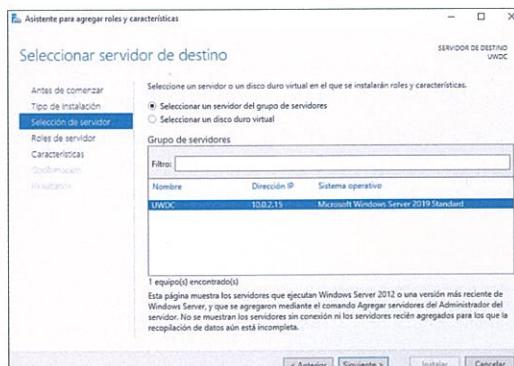


Figura 3.17  
Servidor

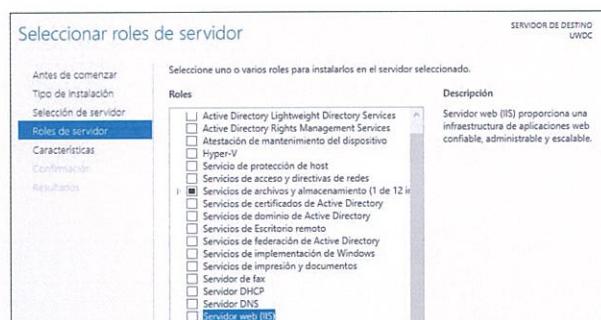
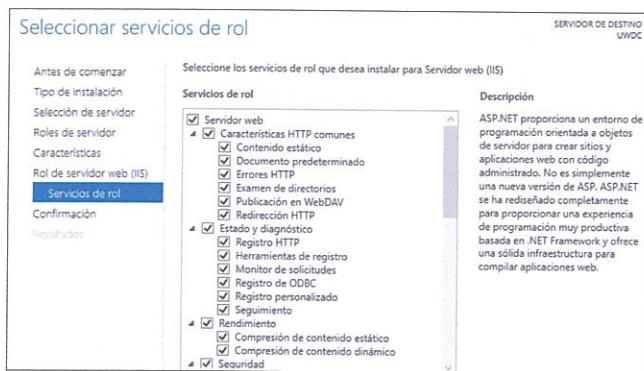


Figura 3.18  
Activación IIS

6. Una vez configurado el servidor web (IIS) es necesario activar una serie de características para que este funcione correctamente (por ejemplo, publicar aplicaciones), como son las siguientes:
  - Características de .NET Framework 3.5.
  - Características de .NET Framework 4.7.

Al seleccionar la característica de .NET Framework 3.5 e instalarla, puede fallar y aparecer un mensaje del tipo *0x800F0950*, que se soluciona introduciendo la imagen o CD del sistema operativo Windows 2019 server y tecleando la ruta alternativa cuando se instalen características.

Después, la instalación pasaría a otra pantalla, en la que sería necesario seleccionar todas las características del servidor web (autenticación, páginas estáticas, dinámicas, etc.) para que funcione correctamente. Y se observará una imagen como la de la figura 3.19:

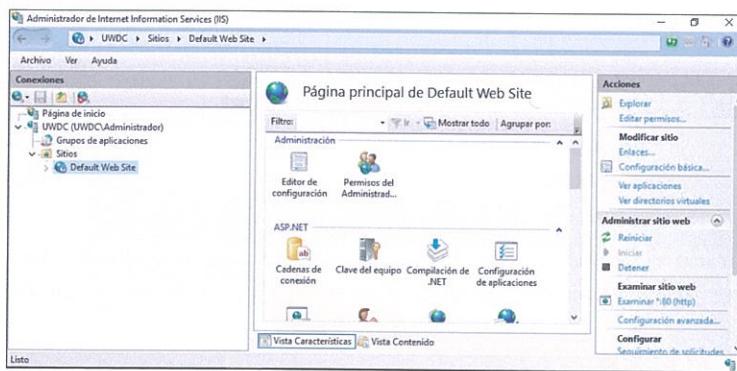


**Figura 3.19**  
Activación IIS

### Actividad propuesta 3.6

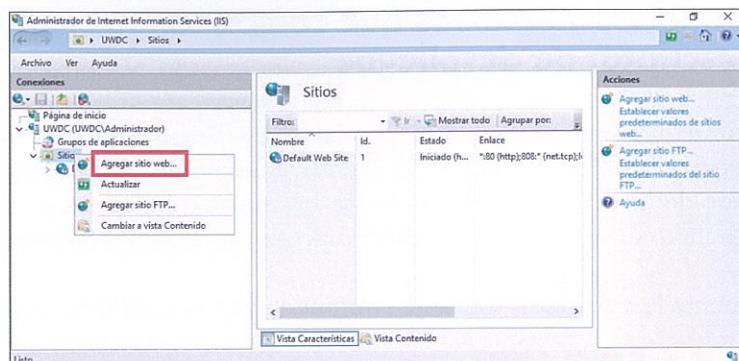
Instala el servicio IIS en Windows Server 2019 o en Windows 10 y visualiza las diferentes opciones que posee.

7. Es el momento de teclear en el navegador http:// localhost o la IP del servidor http://10.0.2.15 y se mostrará la página inicial del servidor IIS. A continuación, se va a crear un sitio o host virtual. Para que se observe la configuración, se parte de la página de administración de IIS, tecleando en el inicio de Windows IIS y se observará la pantalla de la figura 3.20.



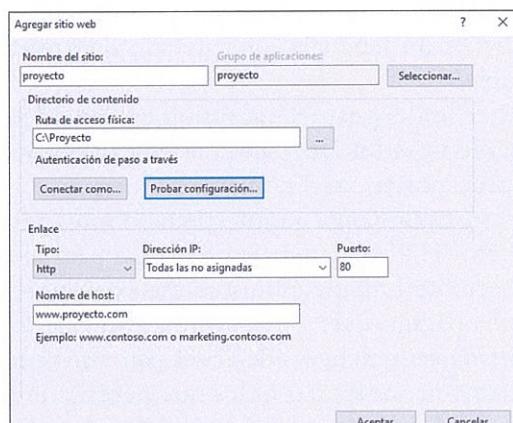
**Figura 3.20**  
IIS

8. Se puede crear un nuevo sitio o usar el sitio o host virtual que viene por defecto. Para que se observe bien la creación de un nuevo sitio, se va a realizar desde el principio y en otra ruta distinta. Para ello se creará una nueva carpeta llamada C:\Proyecto, si el servidor tuviera varias unidades, pues se crearía en la unidad asignada a datos. Si se observa la figura 3.20, se tiene que seleccionar *Sitios* y pulsar el botón derecho para *Agregar sitio web*. Se obtiene la pantalla de la figura 3.21.



**Figura 3.21**  
Sitio web

- Después, se crearán los parámetros que necesita el sitio web, como son el nombre del sitio, la ruta donde se encuentra (C:\Proyecto), el nombre del host. Además, se puede configurar la IP por la que escucha, el puerto y el protocolo (figura 3.22).



**Figura 3.22**  
Configuración del sitio web

- Por último, se necesitaría agregar la IP del servidor (10.0.2.15) en el servidor DNS o en el fichero host. También es necesario en la ruta C:\Proyecto crear un fichero index.html para comprobar que se está accediendo a esta ruta. Se ha realizado en el fichero host de la forma:

```
10.0.2.15 www.proyecto.com
```

Y, posteriormente, en el navegador tecleamos <http://www.proyecto.com>:



**Figura 3.23**  
Comprobación del sitio web

### 3.9. Estructura y recursos que componen una aplicación web. Descriptor de despliegue

Las aplicaciones web se basan en la versión Java EE 8 Web y en un servidor de aplicaciones que deben especificar de alguna forma cómo se van a estructurar los ficheros para que la aplicación se despliegue de forma óptima. Para ello, se necesita un fichero que se denomina *descriptor de despliegue* (fichero .xml) que especificará cómo se desplegará la aplicación y los recursos en el servidor.

Físicamente la aplicación web está compuesta por una información que se puede resumir:

- a) Servlets: son módulos escritos en Java que se utilizan en un servidor para potenciar sus capacidades de respuesta y que se ejecuta sobre el protocolo HTTP.
- b) Páginas jsp: tecnología de programación que permite crear páginas web dinámicas usando HTML y XML.
- c) PHP: lenguaje de propósito general usado en el backend.
- d) Perl: generalmente este lenguaje de programación se ejecuta en el servidor. Apache, por ejemplo, tiene un módulo que permite ejecutar programas de este tipo.
- e) Ruby: es un lenguaje interpretado de propósito general, que también es dinámico y flexible.
- f) ASP: es un lenguaje de scripting del lado del servidor creado por Microsoft.
- g) ASP.NET: es un lenguaje que está diseñado para trabajar con IIS y está escrito para soportar por el .net Framework.
- h) Phyton: es un lenguaje interpretado muy poderoso, fácil de aprender y de alto nivel.
- i) Ficheros HTML: lenguaje de marcas para mostrar información sobre el navegador.
- j) Ficheros de hoja de estilos: se usan con páginas HTML para hacer más fácil el diseño de una página web.
- k) Ficheros .java: ficheros de clases para programar clases en Java.
- l) Ficheros de sonidos: sonidos que necesite la aplicación web.
- m) Ficheros de imágenes: fondos para páginas web, imágenes para hacer más ameno el entorno.

Toda esta información y la que se necesite se puede empaquetar y ejecutar de forma ordenada mediante el fichero descriptor de despliegue.

WWW

**Recurso web**

Consulta la página web de Oracle para ampliar la información sobre Java EE 8.



El servlets es un módulo de Java que se encarga de una función específica dentro de un servidor web. La última versión es la 4.0, que incluye una serie de características novedosas, como puede ser el uso de HTTP 2.0, que permite operaciones push. Esto significa que el servidor puede enviar recursos al cliente sin que este realmente los necesite. Como se puede observar en la figura 3.24:

Una de las partes importantes de la aplicación web es el servlet, como se observa en la figura 3.24. Para ello se necesita un servidor web de aplicaciones como pueden ser Apache Tomcat, GlassFish Server, JBoss, Jetty, etc. Para controlar que no haya errores o colisiones entre una aplicación y otra se asocia un contexto para cada aplicación.

Cuando se crea un proyecto de una aplicación web (Netbeans) se crea el directorio raíz del proyecto, y a partir de la raíz se crean todos los directorios y ficheros importantes de la aplicación. Dos de los directorios importantes son META-INF y WEB-INF, que cuelgan de la carpeta web y esta del directorio, cuyo nombre es el nombre de la aplicación. A continuación, se van a detallar las carpetas más importantes de una aplicación web:

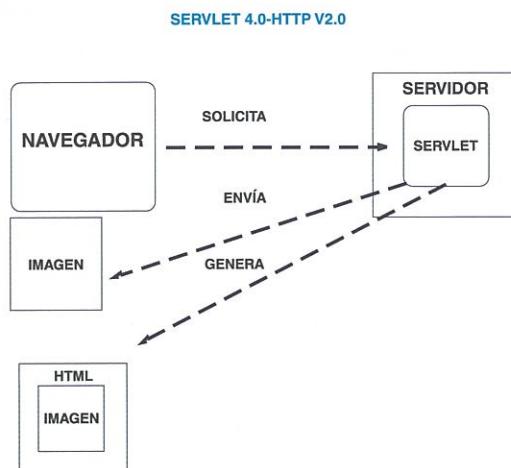
- WEB-INF: es un área privada de la aplicación donde no se puede acceder directamente desde el navegador. Aquí se almacenan los componentes de la configuración del archivo .war. Como, por ejemplo, la página de inicio, la ubicación de los servlets, así como otros componentes de la aplicación. Pero el más importante de todos es el fichero web.xml o descriptor de despliegue.
- WEB-INF/clases/: tiene como componentes las clases Java usadas en el archivo .war y usualmente se encuentran los servlets codificados.
- WEB-INF/lib/: en este directorio se ubicarán los ficheros .jar usados por la aplicación, por ejemplo, para conectarse a una base de datos, o para usar Hibernate o cualquier otra librería que requiera la aplicación.
- META-INF: es un directorio privado de la aplicación y contiene normalmente un fichero denominado context.xml que permite definir el contexto de la aplicación.

### 3.9.1. Archivos war

El archivo war es una de las partes más importantes de la aplicación web, ya que es el método para encapsular una aplicación y prepararla para la distribución en los servidores. Estos servidores deben ser compatibles con servlets y páginas jsp, o cualquier lenguaje de programación usado en la aplicación web. Su nombre procede de *Web Application Archive* (archivo de aplicación web).

Este archivo se puede generar de muchas formas y a través de cualquier IDE de programación, por ejemplo, con la herramienta jar se puede generar, o con la herramienta Netbeans 8.2. Solamente seleccionando la opción *Ejecutar* y, dentro de esta, la opción *Limpiar y Generar Proyecto*. A continuación, en el sistema de archivos de la aplicación web y concretamente en el directorio dist se generará un archivo cuyo nombre será *nombreproyecto.war*.

El contenido de este fichero puede ser el siguiente:



**Figura 3.24**  
Protocolo HTTP v2.0

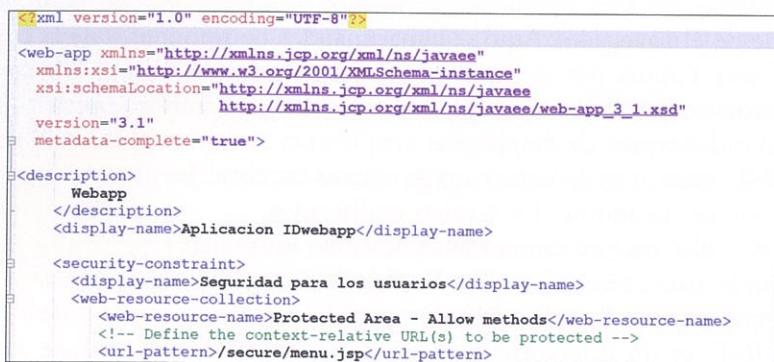
- ✓ Servlets, JSP o cualquier lenguaje de programación elegido para tal fin.
- ✓ Información web: XML, HTML, sonidos, vídeos, etc.
- ✓ Recursos web.

### 3.9.2. Descriptor de despliegue

El descriptor de despliegue es un fichero .xml que describe cómo se ha de desplegar una aplicación web, un módulo o cualquier componente. Al ser un fichero .xml es fácilmente modificable sin cambiar nada el código fuente.

Es necesario para cualquier aplicación web crear un descriptor de despliegue, en caso contrario, la aplicación no funcionará, es como el patrón a seguir.

Cualquier aplicación web debe contener un descriptor de despliegue en la carpeta WEB-INF/web.xml. Se verá un ejemplo de TOMCAT (servidor de aplicaciones), que contendrá un descriptor de despliegue general en la carpeta /apache-tomcat-8.5.5/conf/web.xml. Un ejemplo de descriptor de despliegue de una aplicación puede ser algo parecido al de la figura siguiente:



```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
        http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"
    version="3.1"
    metadata-complete="true">

    <description>
        Webapp
    </description>
    <display-name>Aplicacion IDwebapp</display-name>

    <security-constraint>
        <display-name>Seguridad para los usuarios</display-name>
        <web-resource-collection>
            <web-resource-name>Protected Area - Allow methods</web-resource-name>
            <!-- Define the context-relative URL(s) to be protected -->
            <url-pattern>/secure/menu.jsp</url-pattern>
        </web-resource-collection>
    </security-constraint>
</web-app>
```

**Figura 3.25**  
Descriptor de despliegue

Los servlets deberían estar entre las etiquetas `<web-app>` y `</web-app>`, por ejemplo, algo parecido a lo siguiente:

```
<servlet>
    <servlet-name>IDwebapp</servlet-name>
    <servlet-class>package.IDwebapp.Login</servlet-class>
</servlet>
```

Por último, habrá que arrancar TOMCAT, y para acceder al servlet habría que teclear la siguiente URL:

`http://localhost o IP:8080/IDwebapp(Servlet)`

## Resumen

- En este tercer capítulo se ha explicado todo lo relacionado con el servidor web y, más concretamente, la instalación de Apache sobre la distribución de Linux Debian y en Windows.
- Se ha comenzado detallando la evolución histórica de los servicios web, desde que Tim Berners-Lee inventó el primer navegador hasta nuestros días.
- Posteriormente, se entra en profundidad en los fundamentos, los protocolos de HTTP y HTTPS del servidor web, así como los servidores que existen en el mercado actual. Los más usados hoy son Apache y Nginx.
- En un paso posterior, se ha analizado la arquitectura web, que consta de cliente, servidor y base de datos.
- También se ha hablado de la evolución de la tecnología web desde sus principios, cuando aparecieron los primeros navegadores hasta nuestros días.
- Es importante nombrar las tecnologías usadas tanto en el cliente como en el servidor. Tales tecnologías permiten la interacción entre el servidor y los usuarios.
- Se ha desgranado el servidor Apache. Se ha detallado la instalación, la configuración básica, la estructura de directorios y ficheros que posee apache2 y algunas directivas básicas. Tales directivas, que forman parte de apache2, permiten que dicho servidor funcione de una forma correcta.
- Por último, se ha explicado la estructura que posee una aplicación web, y una primera aproximación de cómo se despliega una aplicación web a partir del descriptor de despliegue y la función del archivo .war.

## Supuestos prácticos

1. Aprovechando la instalación de DNS, se podría poner una máquina definida como webexterno.com que permita conexiones desde el rango de IP 10.2.2.0-10.2.2.254. Demuéstralos con pantallas.
2. Comprueba si los navegadores se comportan igual en la instalación y funcionamiento de Apache, por ejemplo, Opera, Firefox, Google Chrome e Internet Explorer.
3. Realiza una investigación de los diferentes servidores web que existen en el mercado y observa cuándo se adaptan a nuestro entorno productivo empresarial.
4. Comprueba el funcionamiento de la directiva Options Indexes dentro del fichero de configuración apache2.conf.
5. Activa un sitio que posea la ruta /web/publico. En este fichero es necesario incluir un fichero index.html que ponga la ruta y el nombre de la persona que lo haga. Hay que modificar los ficheros apache2.conf y crear un sitio nuevo dentro del directorio sites-available que se llame publico.conf. Demuéstralos con pantallas.



## Ejercicios propuestos

1. Instala y configura el servicio de apache2 en el sistema operativo Linux donde la máquina tenga como IP 192.168.1.15. Además, sería necesario configurar que Apache escuche por el puerto 8080.
2. Instala y configura el servicio de apache2 en el sistema operativo Windows donde la máquina tenga como IP 192.168.1.15. Además, sería necesario configurar que Apache escuche por el puerto 8080.
3. Realiza la instalación y configuración básica del servidor Nginx con la IP 192.168.1.15 y que escuche por el puerto 8090.

## ACTIVIDADES DE AUTOEVALUACIÓN

1. ¿De qué consta una aplicación web?:  
 a) Servidor y cliente.  
 b) Servidor, cliente y bases de datos.  
 c) Servidor y comunicación.  
 d) Servidor, cliente y protocolo.
2. Para comprobar que nuestro servidor Apache funciona correctamente hay que teclear (por defecto):  
 a) http://localhost.  
 b) http://localhost:8080.  
 c) http://127.0.0.1:8080.  
 d) Ninguna de las anteriores es correcta.
3. ¿Cuál es el fichero principal de Apache?:  
 a) apache.conf  
 b) confapache2.conf  
 c) apacheconf.conf  
 d) apache2.conf
4. Apache Server es un servidor:  
 a) Con licencia, de software libre que solo se usa en Linux.  
 b) Gratuito, pero es necesario pedir permiso para ampliar el código del servidor.  
 c) Gratuito, de software libre y multiplataforma.  
 d) Las respuestas a) y c) son correctas.

5. ¿Qué comando permite habilitar un site en Apache?:

- a) a2enconf.
- b) a2dissite.
- c) a2ensite.
- d) a2disconf.

6. ¿Cuál de estas tecnologías se usa solo en el servidor web?:

- a) CGI.
- b) Javascript.
- c) HTML y CSS.
- d) Ninguna de las respuestas anteriores es correcta.

7. ¿Cuál de estas tecnologías se usa en el cliente web?:

- a) CGI.
- b) HTML y CSS.
- c) Perl.
- d) ASP.NET.

8. ¿Qué fichero de configuración se usa para configurar los puertos por los que escucha el servidor web?:

- a) apache2.conf.
- b) ports.conf.
- c) portsapache2.conf.
- d) puerto.conf.

9. ¿Qué directorio de Apache almacena los sitios activos?:

- a) sites-enable.
- b) sites-available.
- c) sites-enabled.
- d) Ninguna de las respuestas anteriores es correcta.

10. ¿Cuáles son los servidores web más usados en el mundo?:

- a) Nginx y Apache.
- b) IIS y GWS.
- c) Apache y GWS.
- d) GWS y LiteSpeed.

### SOLUCIONES:

1. **a** **b** **c** **d**  
2. **a** **b** **c** **d**  
3. **a** **b** **c** **d**  
4. **a** **b** **c** **d**

5. **a** **b** **c** **d**  
6. **a** **b** **c** **d**  
7. **a** **b** **c** **d**  
8. **a** **b** **c** **d**

9. **a** **b** **c** **d**  
**10. a** **b** **c** **d**

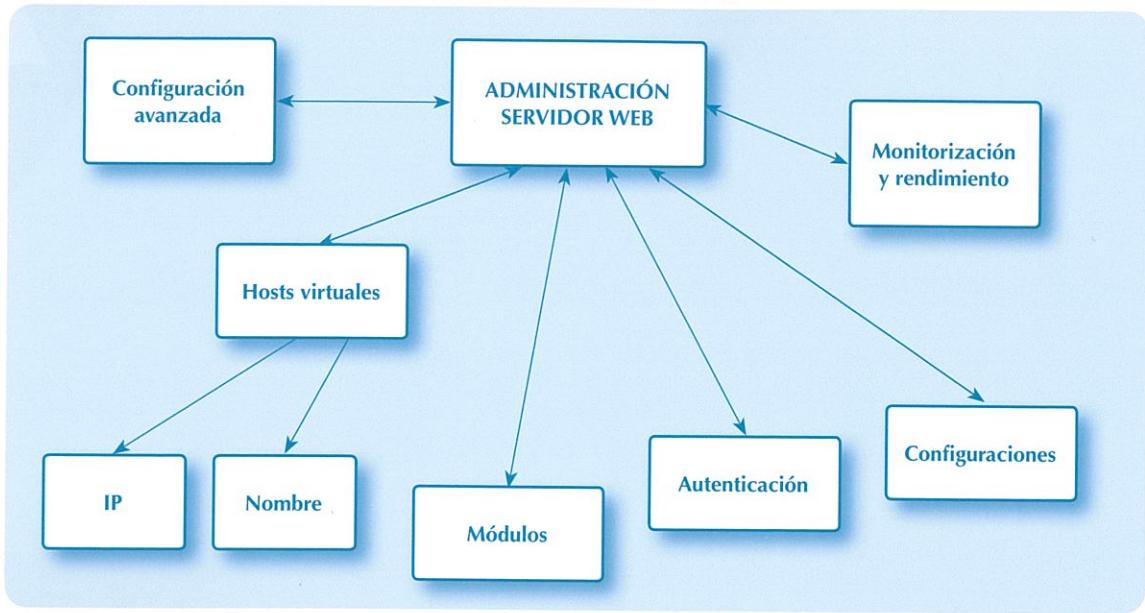


# Administración de servidores web

## Objetivos

- ✓ Reconocer los parámetros de administración más importantes del servidor web.
- ✓ Ampliar la funcionalidad del servidor mediante la activación y configuración de módulos.
- ✓ Crear y configurar sitios virtuales.
- ✓ Establecer los mecanismos de autenticación y control de acceso al servidor.
- ✓ Obtener e instalar certificados digitales.
- ✓ Constituir los mecanismos para asegurar las comunicaciones entre el cliente y el servidor.
- ✓ Realizar pruebas de funcionamiento y rendimiento del servidor web.
- ✓ Llevar a cabo los ajustes necesarios para la implantación de aplicaciones en el servidor web.

## Mapa conceptual



## Glosario

**Core.** Es un software que constituye la parte fundamental de un servicio o sistema operativo y que se ejecuta en modo administrador, también llamado núcleo o kernel.

**Directiva.** Es una palabra reservada dentro de un fichero de configuración que va acompañada de parámetros para realizar una función específica.

**Paquetes Linux.** Son programas que se pueden instalar mediante el comando apt-get en las distribuciones Linux y que forman parte del repositorio que posea cada distribución.

**Parámetro.** Es una palabra o una opción que personaliza un fichero de configuración de cualquier programa o servicio para realizar una función específica.

**Pid (process identifier).** Identificador de un proceso en un sistema operativo.

## 4.1. Introducción

Antes de comenzar a entrar en materia, este tema está estrechamente relacionado con el anterior pero será más práctico de cara al usuario final. En esta unidad se explicará en profundidad todo lo relacionado con la administración de un servidor web. Primero de todo, se comenzará con la configuración avanzada del servidor web, para posteriormente pasar a configurar los hosts virtuales y cómo se usan. También se hablará de los módulos adicionales que posee Apache como tal. Por otro lado, se configurará la autenticación y el control de acceso. Se explicará la seguridad en el servidor Apache mediante el protocolo HTTPS y certificados. Y, por último, se configurarán hosts virtuales en el sistema operativo Windows.

## 4.2. Configuración avanzada del servidor web

En la unidad anterior se explicó la instalación del servidor Apache 2.4 y algunas nociones básicas, en esta sección se explicarán en profundidad las opciones y directivas más importantes de este servidor web.

La configuración de Apache se realiza a través de ficheros de texto o ficheros .conf, los cuales contienen directivas que se pueden configurar fácilmente. Una vez configurada la directiva correspondiente es necesario reiniciar el servicio de Apache y los cambios realizados se llevarán a efecto. Se pueden agrupar en cinco categorías, que se muestran a continuación:

1. *Directivas de control de funcionamiento*: permiten controlar el flujo de trabajo del servidor de forma global.
2. *Parámetros del servidor web*: son parámetros globales para todos los hosts virtuales configurados. Se puede especificar algún parámetro en particular en cada host virtual.
3. *Configuración de hosts virtuales*: permite crear nombres de dominio diferentes para acceder al servidor desde IP diferentes y puertos diferentes. Se verá en profundidad en un apartado posterior.
4. *Instalación y configuración de módulos*: una de las partes más importantes de Apache, que destaca por su modularidad. Se verá en un apartado posterior.
5. *Monitorización y rendimiento del servidor*: esta parte se encarga de controlar los accesos al servidor, hosts virtuales, y monitoriza las conexiones y demás problemas o fallos del servidor.

### 4.2.1. Directivas de control del servidor Apache

En este apartado se explicarán las directivas de control que permiten controlar el funcionamiento del servidor y se localizan en el fichero apache2.conf. Antes de comenzar a detallar las más importantes, sería necesario realizar una copia del fichero para que en caso de error se pueda volver hacia atrás y no se tenga ningún problema en iniciar el servidor. La lista es la siguiente:

- *ServerRoot*. Es la ruta de directorios raíz a partir de la cual cuelga toda la configuración del servidor Apache.

- *ServerName*. Es el nombre por el cual se identifica Apache, es necesario definirlo en el fichero de hosts para que resuelva la IP o crear un registro DNS en el servicio DNS.
- *DefaultRuntimeDir \${APACHE\_RUN\_DIR}*. Es el directorio donde el servidor Apache crea ficheros en tiempo de ejecución, como bloqueos, memoria compartida, etc. Para ello existe un fichero denominado /etc/apache2/envvars, donde se crean todas las variables de entorno que empiezan por \$.
- *PidFile \${APACHE\_PID\_FILE}*. Es la directiva encargada de crear el id del proceso del servidor Apache y apunta a la variable de entorno APACHE\_PID\_FILE.
- *Timeout 300*. Es el tiempo en segundos que el servidor espera antes de enviar un fallo a una solicitud. Lo ideal es dejarlo como viene configurado, aunque dependiendo de las necesidades se puede aumentar o disminuir.
- *KeepAlive On*. Permite habilitar las conexiones persistentes mediante el protocolo HTTP. Permite enviar varias solicitudes en la misma conexión TCP.
- *MaxKeepAliveRequests 100*. Es el número de solicitudes que se permiten en una sola conexión. Si se coloca un 0 serían infinitas solicitudes, lo que puede llegar a saturar el servidor. Lo ideal es el número que viene por defecto. Esta directiva no tiene sentido si la anterior directiva está colocada a Off.
- *KeepAliveTimeout 5*. Es el número de segundos que espera entre solicitud y solicitud de la misma conexión del mismo cliente.
- *User \${APACHE\_RUN\_USER}*. Es el usuario que ejecutará Apache que debe tener permisos de administrador o root, se basa en el fichero envvars para coger el valor de la variable APACHE\_RUN\_USER.
- *Group \${APACHE\_RUN\_GROUP}*. Es el grupo al que pertenece el usuario que ejecutará Apache, se basa en el fichero envvars para coger el valor de la variable APACHE\_RUN\_GROUP.
- *HostnameLookups Off*. Esta directiva tiene tres posibles valores On, Off y Double. Si se configura a On el servidor Apache resuelve automáticamente las direcciones IP para cada conexión. Este proceso sobrecarga el procesamiento del servidor, ya que es necesario consultar un servicio DNS para resolver cada petición. Si está a Double realiza las búsquedas directas e inversas, añade más sobrecarga. Por lo que su valor por defecto e ideal es Off.
- *ErrorLog \${APACHE\_LOG\_DIR}/error.log*. Es el registro de log de los errores del servidor. Por defecto, la ruta donde se almacena es /var/log/apache2, se puede modificar en el fichero de variables de entorno /etc/apache2/envvars.
- *LogLevel warn*. Establece los detalles que tendrá el log de errores del servidor. Los diferentes valores están ordenados por nivel de detalle desde el menor detalle hasta los más detallados. El orden es emerg, alert, crit, error, warn, notice, info o debug. Al menos es necesario configurar el valor de crit, de ahí para arriba.
- *IncludeOptional mods-enabled/\*.load*. Incluye el fichero de módulos de configuración activados.

### Actividad propuesta 4.1



Instala el servidor Apache en Linux y prueba las directivas anteriores como ServerName, ServerRoot, ErrorLog, etc.

- *IncludeOptional mods-enabled/\*.conf*. Incluye el fichero de módulos de configuración habilitados.
- *Include ports.conf*. Incluye el fichero de configuración de los puertos de escucha de los hosts virtuales y del servidor.
- *AccessFileName .htaccess*. Directiva que permite dar acceso a un directorio configurando tal fichero.
- *IncludeOptional conf-enabled/\*.conf*. Incluye el fichero de configuraciones habilitadas para el servidor Apache.
- *IncludeOptional sites-enabled/\*.conf*. Incluye el fichero de sitios o hosts virtuales habilitados para el servidor Apache.

#### 4.2.2. Parámetros del servidor

El objetivo de esta sección es configurar los valores del servidor principal, que permitirán a los hosts virtuales poseer una configuración por defecto. Las directivas son las siguientes:

- ✓ *ServerAdmin*. Este parámetro sirve para suministrar una dirección de email del administrador del sistema al usuario que se conecta mediante un host virtual y será mostrada en las páginas de error generadas. Permitirá ponerse en contacto con el administrador del sistema.
- ✓ *ServerName*. Es el nombre del servidor o una dirección IP válida y un puerto opcional. Esta directiva se puede usar tanto en el fichero principal como en los hosts virtuales. La nomenclatura sería *ServerName domain-name | IP [:port]*.
- ✓ *ServerAlias*. Esta directiva permite establecer nombres adicionales y complementarios al host declarado en la directiva anterior. Su nomenclatura sería *ServerAlias hostname [hostname]...*
- ✓ *DocumentRoot*. Esta directiva establece el directorio a partir del cual es visible la ruta desde el servidor web. La nomenclatura es *DocumentRoot directory-path*.
- ✓ *DirectoryIndex*. Es la lista de recursos que se van a buscar cuando un cliente solicite un directorio. La nomenclatura es *DirectoryIndex disabled | local-url [local-url]...* Permite declarar varios tipos de ficheros como *index.html, index.jsp, index.htm, etc.*
- ✓ *ErrorDocument*. Es la directiva que permite establecer una página de error para que la visualice el cliente en caso de no poder atender la petición por la causa que sea. La sintaxis sería *ErrorDocument error-code document*.
- ✓ *AccessFileName*. Es la directiva que permite el acceso limitado a determinados directorios. Se verá en profundidad posteriormente.

#### Recurso web

www

Todas las directivas del servidor Apache 2.4 se pueden encontrar en su web.



### 4.3. Hosts virtuales. Creación, configuración y utilización

Hasta ahora se ha visto una conexión mediante un sitio o host virtual por el puerto 80 o 8080. En esta sección se va a comentar cómo definir varios nombres de dominio distintos, que pueden ubicarse en directorios diferentes y en puertos diferentes mediante la declaración de hosts virtuales basados en IP o en nombres o en servidores principales. Para declarar estos hosts virtuales sería necesario modificar el fichero host para poder declarar el nombre en cuestión, o declararlo en el servidor DNS, que ya se comentó antes como un registro DNS. En el fichero /etc/hosts sería de la siguiente forma:

```
GNU nano 2.8.7 Fichero: /etc/hosts
127.0.0.1      localhost
127.0.0.1      osboxes
10.0.2.15      webadmin
```

**Figura 4.1**  
Fichero hosts

Por lo tanto, si se necesita que un servidor web atienda peticiones de diferentes dominios, la solución es el uso de hosts virtuales que puedan estar asociados a dominios diferentes. Con esta solución se abre un abanico de posibilidades, las más usuales son las siguientes:

- Host virtual basado en nombre:* un servidor web que tiene una sola IP, y esta IP tiene asociados diferentes dominios. El contenido es el mismo, a nivel de páginas web, sin embargo, son nombres de acceso diferentes.
- Host virtual basado en IP:* otra opción sería un servidor web que tiene diferentes IP públicas, ya sean físicas o virtuales. Cada dirección web está asociada a una IP.
- Host virtual mixto:* esta es menos usual pero también se puede dar, es una mezcla de las dos opciones anteriores.

#### 4.3.1. Hosts virtuales basados en nombre

Se va a crear un host virtual basado en nombre desde cero por el puerto 80 para que se observe el procedimiento. Para ello se van a dar los datos siguientes:

- Crear un directorio que se llame empresa-daw en la ruta /var/www
- Este directorio debe permitir el acceso a las URL: www.empresa-daw.com y empresa-daw.local.
- Los logs se deben configurar de acceso y de error, que se denominan empresa-daw-access.log y empresa-daw-error.log, respectivamente. El log debe ser combined.

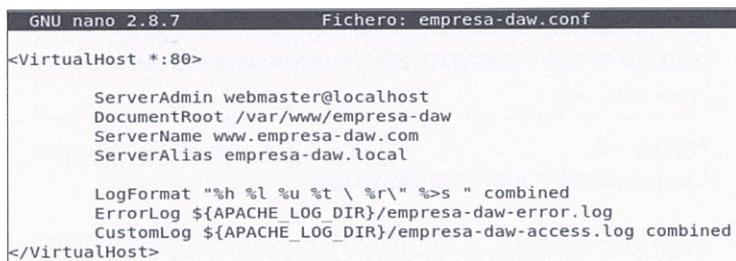
A continuación, se van a describir los pasos para llevar a cabo la configuración de un host virtual:

- Al principio se creará el directorio que pone en la ruta y se le dará permisos de lectura y ejecución (ya los tiene), tal y como se muestra en la figura 4.2.

```
root@osboxes:/var/www# ls
html
root@osboxes:/var/www# mkdir empresa-daw
root@osboxes:/var/www# ls -l
total 8
drwxr-xr-x 2 root root 4096 abr 14 06:46 empresa-daw
drwxr-xr-x 2 root root 4096 abr 12 14:20 html
root@osboxes:/var/www# cd empresa-daw
root@osboxes:/var/www/empresa-daw# ls
```

**Figura 4.2**  
Directorio empresa-daw

- Seguidamente se creará el fichero index.html para identificar que está funcionando el host virtual en la ruta anteriormente creada. Por otro lado, es necesario crear el fichero de configuración empresa-daw.conf (figura 4.3).



```
GNU nano 2.8.7          Fichero: empresa-daw.conf
<VirtualHost *:80>
    ServerAdmin webmaster@localhost
    DocumentRoot /var/www/empresa-daw
    ServerName www.empresa-daw.com
    ServerAlias empresa-daw.local

    LogFormat "%h %l %u %t \ %r\" %>s " combined
    ErrorLog ${APACHE_LOG_DIR}/empresa-daw-error.log
    CustomLog ${APACHE_LOG_DIR}/empresa-daw-access.log combined
</VirtualHost>
```

**Figura 4.3**  
Fichero empresa-daw.conf

La nomenclatura de VirtualHost es <VirtualHost IP[:puerto] [IP][:puerto]....> hay que poner mínimo una IP, que puede ser asterisco, que significa todas las IP. El puerto es opcional y se pueden poner más IP y puertos.

Es necesario habilitar este sitio con el comando #a2ensite empresa-daw. Se comentan las diferentes directivas que posee el fichero anterior:

- *ServerAdmin*: email del administrador del servidor web.
- *DocumentRoot*: ruta donde se ubicará el fichero index.html.
- *ServerName*: nombre del sitio web que albergará la información.
- *ServerAlias*: alias del nombre del sitio web.
- *LogFormat*: formato del fichero de log.
- *ErrorLog*: log de errores del sitio web.
- *CustomLog*: log de acceso al sitio web.

- En un paso posterior, se modifica el fichero /etc/hosts donde es necesario incluir la siguiente línea para que pueda resolver los nombres declarados en el fichero empresa-daw.conf. La línea es la siguiente:

10.0.2.15 (IP del servidor) www.empresa-daw.com empresa-daw.local

4. Es necesario también que en el fichero de configuración apache2.conf, el servidor web dé permiso para leer en el directorio /var/www. Esto se realiza con las líneas que se muestran en la figura 4.4.

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

**Figura 4.4**  
Directiva Directory

5. Por último, se reinicia el servicio Apache para que se lleven a efecto todos los cambios realizados, y se obtienen las siguientes figuras accediendo a los nombres www.empresadaw.com y empresa-daw.local:



**Figura 4.5**  
Comprobación empresadaw.com



**Figura 4.6**  
Comprobación empresadaw.local

Si en lugar de montar el host virtual por el puerto 80 lo hiciéramos por el puerto 8080 u otro puerto, la configuración sería la de la figura 4.7.

En este caso, permite conexiones por el puerto 80 y 8080 en este host virtual. Para ello también en la configuración del fichero ports.conf debe existir una configuración parecida a la de la figura 4.8.

```
<VirtualHost *:80 *:8080>
    ServerAdmin webmaster@localhost
    DocumentRoot /var/www/empresa-daw
    ServerName www.empresadaw.com
    ServerAlias empresadaw.local

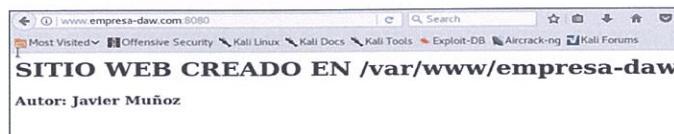
    LogFormat "%h %l %u %t \ %r\" %>s " combined
    ErrorLog ${APACHE_LOG_DIR}/empresadaw-error.log
    CustomLog ${APACHE_LOG_DIR}/empresadaw-access.log combined
</VirtualHost>
```

Listen 80  
Listen 8080

**Figura 4.7**  
Fichero host virtual

**Figura 4.8**  
Puertos de escucha

Con ello ya se tendría la configuración modificada, y si se reinicia el servicio de Apache, nos permitiría aceptar conexiones por el puerto 8080:



**Figura 4.9**  
Comprobación del puerto 8080



### Actividad propuesta 4.2

Crea un fichero de host virtual llamado daw.conf que permita el acceso a un directorio llamado /var/www/html. Crea una página index.html para comprobar que funcione correctamente. Además, debe escuchar por el puerto 80 y debe atender peticiones a la URL www.daw.es.

#### 4.3.2. Hosts virtuales basados en IP

Antes de realizar un ejercicio práctico se van a comentar algunas consideraciones. Para realizar este apartado se necesita de varias IP físicas o virtuales que están relacionadas con nuestro servidor web. En nuestro caso, se van a montar dos IP para comprobar que funciona correctamente esta opción del servidor Apache. La utilidad del host virtual basado en IP es para mostrar diferente contenido HTML en función de la IP.

Esta opción es menos utilizada que la anterior, porque en caso de que el servidor tenga acceso desde el exterior sería necesario contratar IP públicas. Si solamente el servidor web da servicio a nuestra Intranet, no sería necesario contratar IP.

#### TOMA NOTA



Esta opción es muy útil, por ejemplo, en caso de una compañía que tenga un portal web para el exterior y otro diferente para los empleados de la empresa. Esto es a lo que se denomina portal de Internet y portal de Intranet.

Se va a crear un host virtual basado en IP desde cero para que se observe el procedimiento. Para ello se van a dar los datos siguientes:

- Se parte de dos IP 10.0.2.15 y 10.0.2.16 que van a tener contenidos diferentes.
- Cada una de las IP va a la misma ruta /var/www pero a directorios diferentes.
- La IP 10.0.2.15 va al directorio *empresa-internet* y la IP 10.0.2.16 va al directorio *empresa-intranet*.

- El directorio empresa-internet tendrá los nombres www.empresa.com y empresa.es.
- El directorio empresa-intranet tendrá los nombres www.empresa-intranet.com y empresa-intranet.es.
- Los logs que se deben configurar de acceso y de error de Internet serán *empresa-access.log* y *empresa-error.log*, respectivamente. Y los de la Intranet serán *empresa-intranet-access.log* y *empresa-intranet-error.log*. El log debe ser combined en ambos casos.

A continuación, se van a describir los pasos para llevar a cabo la configuración de un host virtual basado en IP con el usuario root:

1. Al principio se va a configurar una interfaz de red virtual para tener una segunda IP (10.0.2.16) y se realiza como se ve en la figura 4.10.

Posteriormente, se ejecuta el comando ifup eth0:0 y a continuación ifconfig para comprobar que está levantada la interfaz virtual, y se obtiene la pantalla de la figura 4.11.

```
# Interfaz virtual
auto eth0:0
iface eth0:0 inet static
address 10.0.2.16
netmask 255.255.255.0
```

**Figura 4.10**  
Interfaz virtual

```
eth0:0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
          inet 10.0.2.16 netmask 255.255.255.0 broadcast 10.0.2.1
            ether 08:00:27:f6:51:97 txqueuelen 1000 (Ethernet)
```

**Figura 4.11**  
Ethernet eth0:0

2. Después, se crean los dos directorios empresa-internet y empresa-intranet, donde irán ubicados los dos ficheros index.html. La ruta será /var/www/empresa-internet y /var/www/empresa-intranet. Con los comandos siguientes:

```
#mkdir /var/www/empresa-internet
#mkdir /var/www/empresa-intranet
```

### Actividad propuesta 4.3



Crea una interfaz virtual mediante el fichero de configuración apropiado para las IP 192.168.1.15 y 192.168.1.16. Demuestra que funciona mediante el comando ping.

3. Seguidamente se crearán los ficheros index.html para identificar que está funcionando el host virtual en las rutas anteriormente creadas. Por otro lado, es necesario crear el fichero de configuración empresa.conf (figura 4.12).

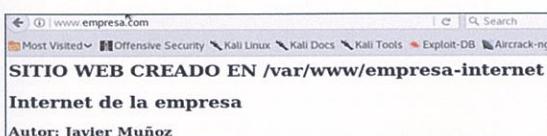
Es necesario deshabilitar este sitio con el comando `#a2dissite empresa-daw` creado en el anterior ejercicio y habilitar el nuevo sitio `#a2ensite empresa`. Es necesario comentar que se pueden crear en un fichero, que es nuestro caso, en ficheros diferentes, pero para ello sería necesario habilitar los dos ficheros. Las directivas ya se comentaron en el

anterior ejercicio. Solamente en este caso, en la etiqueta <VirtualHost 10.0.2.15:80> hemos puesto una IP y en la segunda etiqueta hemos puesto otra IP (10.0.2.16). Por último, queda comentar que se podrían también configurar varios puertos.

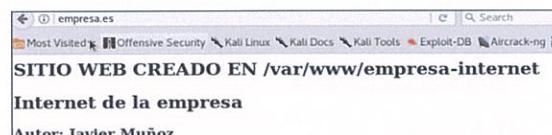
```
<VirtualHost 10.0.2.15:80>
    <VirtualHost 10.0.2.16:80>
```

**Figura 4.12**  
Host virtuales 80 y 8080

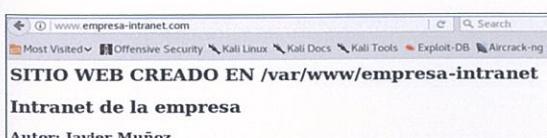
4. En un paso posterior, se modifica el fichero /etc/hosts donde es necesario incluir las siguientes líneas para que pueda resolver los nombres declarados en el fichero empresa.conf. La línea es la siguiente:
  - 10.0.2.15 (IP del servidor) www.empresia.com empresia.es
  - 10.0.2.16 (IP virtual) www.empresia-intranet.com empresia-intranet.es
5. Por último, se reinicia el servicio de Apache para que se activen todos los cambios realizados, y se obtiene la siguiente pantalla accediendo a los nombres www.empresia.com, empresia.es, www.empresia-intranet.com y empresia-intranet.es:



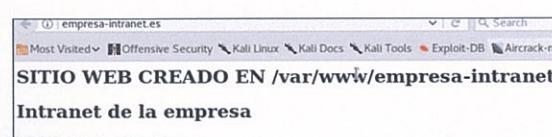
**Figura 4.13**  
Comprobación host virtual empresia.com



**Figura 4.14**  
Comprobación host virtual empresia.es



**Figura 4.15**  
Comprobación host virtual  
empresia-intranet.com



**Figura 4.16**  
Comprobación host virtual  
empresia-intranet.es



## TOMA NOTA

Cuando se están probando o creando hosts virtuales es necesario siempre limpiar la caché del navegador. El navegador podría mostrar una página anterior que esté almacenada en caché y no muestre la que realmente se está ejecutando en ese momento.

### 4.3.3. Host virtual mixto

En este apartado se van a comentar algunas configuraciones que son atípicas y que normalmente no se usan en un servidor web en producción.

```
<VirtualHost 10.0.2.15:80>
DocumentRoot /var/www/empresa-internet
ServerName www.empresa-daw.com
</VirtualHost>
<VirtualHost 10.0.2.15:80>
DocumentRoot /var/www/empresa-internet2
ServerName www.empresa-daw2.com
</VirtualHost>
<VirtualHost 10.0.2.16:80>
DocumentRoot /var/www/empresa-internet3
ServerName www.empresa-daw3.com
</VirtualHost>
```

Por el contrario, si deseamos que admita por todos los puertos podemos asignar el comodín, pero no es lo ideal. Las recomendaciones nos dicen que es necesario especificar bien los puertos por donde escucha el servidor y el host virtual. Esto contribuirá a controlar las posibles amenazas desde el exterior.

Otra configuración podría ser solamente la IP como la siguiente:

```
<VirtualHost 10.0.2.16>
DocumentRoot /var/www/empresa-internet3
ServerName www.empresa-daw3.com
</VirtualHost>
```

#### Actividad propuesta 4.4



Crea un virtual host mixto basándote en la configuración explicada anteriormente, y comprueba que funciona.

## 4.4. Módulos: instalación, configuración y uso

Una de las características más importantes de Apache es su modularidad, además de su estabilidad, disponibilidad y escalabilidad. Apache tiene una gran cantidad de funcionalidades, que si

estuvieran todas activas supondrían una carga excesiva para el servidor web. Como el servidor Apache es modular, esto es, solamente las funciones básicas son incluidas en el núcleo del servidor. Para ello, Apache divide sus funcionalidades en módulos que se activan según sean necesarios. Por defecto, el servidor carga los módulos en tiempo de compilación dinámicamente.

Tales módulos pueden ser compilados separadamente y añadidos con la directiva LoadModule. Como se explicó en el anterior capítulo, los módulos se dividen en dos directorios que cuelgan del principal y son los siguientes:

- /etc/apache2/mod-available: módulos que están disponibles con la instalación que existe.
- /etc/apache2/mod-enabled: módulos que están activos y que son enlaces simbólicos al directorio anterior. Estos módulos se cargarán la próxima vez que se reinicie Apache.

En el directorio *mod-available* se tienen dos tipos de ficheros con extensiones .conf y .load. El fichero .load, por ejemplo mod\_userdir.load, tiene la línea LoadModule userdir\_module /usr/lib/apache2/modules/mod\_userdir.so que permite cargar la librería de la ruta anterior. Y el fichero mod\_userdir.conf contiene la configuración de cómo actúa la directiva en el servidor web.

En nuestro caso, como se está trabajando con Debian, mediante el uso del comando *apachectl -M* podemos observar qué módulos se encuentran activos en nuestro servidor web (figuras 4.17 y 4.18).

```
root@osboxes:~# apachectl -M
Loaded Modules:
 core_module (static)
 so_module (static)
 watchdog_module (static)
 http_module (static)
 log_config_module (static)
 logio_module (static)
 version_module (static)
 unixd_module (static)
 access_compat_module (shared)
 alias_module (shared)
 auth_basic_module (shared)
 authn_core_module (shared)
 authn_file_module (shared)
```

**Figura 4.17**  
Módulos de Apache I

```
authz_core_module (shared)
authz_host_module (shared)
authz_user_module (shared)
autoindex_module (shared)
deflate_module (shared)
dir_module (shared)
env_module (shared)
filter_module (shared)
mime_module (shared)
mpm_event_module (shared)
negotiation_module (shared)
reqtimeout_module (shared)
setenvif_module (shared)
status_module (shared)
userdir_module (shared)
```

**Figura 4.18**  
Módulos de Apache II

Si se observa estas figuras, algunos módulos tienen la palabra static y otros shared. Los módulos pueden ser estáticos (static) o compartidos (shared). Un módulo estático se incluye dentro del binario de Apache, por lo que siempre estará disponible. Sin embargo, el módulo compartido se carga en tiempo de ejecución y sería necesario usar la directiva LoadModule en la configuración de Apache.

Los módulos en Apache se pueden encontrar de dos formas, compilados de forma individual como una biblioteca de acceso dinámico (mod\_user.so) o compilados dentro del ejecutable de apache2. Para conocer qué módulos incluye apache2 en tiempo de ejecución, ejecutamos el comando *apache2 -l* y obtenemos la pantalla de la figura 4.19.

```
root@osboxes:/etc/apache2/mods-enabled# apache2 -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
```

**Figura 4.19**  
Módulos en tiempo de ejecución

Para observar los restantes módulos que se pueden cargar en cualquier momento se ejecuta el comando ls /usr/lib/apache2/modules.

Se van a nombrar algunos módulos que son de interés, algunos de los cuales se probarán en este tema, como son los siguientes:

- *Módulo SSL*: permite implementar el protocolo de seguridad SSL.
- *Módulo LDAP*: permite la validación con el servicio de directorio.
- *Módulo PHP*: se puede ejecutar PHP en el servidor web.
- *Módulo UserDir*: permite que todos los usuarios tengan una página web en su directorio personal.
- *Módulo Security*: permite bloquear contenidos sobre la base de datos.
- *Módulo Proxy*: el servidor Apache se convierte en un proxy inverso.

Los comandos para activar y desactivar los módulos son los siguientes:

- ✓ a2enmod nombre\_módulo sin extensión. #a2enmod userdir
- ✓ a2dismod nombre\_módulo sin extensión. #a2dismod userdir

Suponed que se necesita un módulo que no está instalado, pues simplemente se busca si existe con el comando #apt-cache search apache2-mod.

Por último, existe otra opción si no disponemos de los comandos a2enmod y a2dismod. Solamente sería necesario crear un enlace simbólico desde el directorio mods-available al directorio mods-enabled.

Veamos un ejemplo con el módulo userdir. Se pueden crear los enlaces simbólicos de la siguiente forma:

```
ln -s /etc/apache2/mods-available/userdir.load /etc/apache2/mods-enabled/userdir.load
ln -s /etc/apache2/mods-available/userdir.conf /etc/apache2/mods-enabled/userdir.conf
```

Y para eliminarlos sería simplemente de la siguiente forma:

```
rm -f /etc/apache2/mods-enabled/userdir.*
```

Para instalar y desinstalar un módulo sería con los siguientes comandos:

```
apt-get install nombre-módulo.
```

```
apt-get remove nombre-módulo.
```



### Actividad propuesta 4.5

Practica con los comandos anteriores para deshabilitar módulos, habilitar módulos, desinstalar módulos e instalarlos.

## 4.5. Autenticación y control de acceso a directorios

Hasta ahora se ha permitido la navegación por parte de los usuarios por todo el host o sitio virtual que se ha creado, ya sea mediante páginas estáticas o dinámicas. Pero me surge la pregunta: ¿sería necesario controlar el acceso a ciertos directorios mediante la solicitud de usuario y contraseña?. Esta es la clave de este apartado. Se van a realizar algunas demostraciones de cómo se puede implementar esta configuración en el servidor web Apache.

Si un usuario se autentica en cualquier página actual, como puede ser su banco o una tienda virtual para comprar, es necesario validarse. Para ello el servidor web necesita de una base de datos o servicio de directorio. Con relación a la base de datos, puede ser Oracle, Mysql, MongoDB, y en cuanto a servicio de directorio, puede ser LDAP.

Para este tipo de casos es necesario pensar en dos conceptos, por un lado la autenticación y por otro el control de acceso:

- a) *Control de acceso*: es la acción que permite controlar el acceso a determinados dispositivos a través de su IP o una red de IP. El servicio Apache determina si se tiene acceso a un recurso o no mediante una directiva determinada.
- b) *Autenticación*: es una acción más específica y permite a Apache dar acceso a un recurso mediante la validación de un usuario y contraseña. Si es válido, el servicio Apache le dará acceso al recurso en cuestión. El método de control de acceso y autenticación son complementarios, esto es, se pueden usar y se deben usar al mismo tiempo para restringir al máximo el uso de ciertos recursos.

Apache proporciona varios esquemas para poder autenticar a usuarios, unos más seguros que otros. Se van a realizar algunas configuraciones interesantes; como ejemplo se definirán los siguientes métodos:

- *Método Basic*: este esquema consiste en que el cliente solicita un recurso del servidor y el servidor le solicita un usuario y una contraseña para darle acceso al recurso. Pero el usuario y la contraseña viajan en texto claro. Lo ideal en este caso es realizarlo con el protocolo SSL. El módulo usado para este método es mod\_auth\_basic, que en Debian corresponde al fichero auth\_basic.load.
- *Método Digest*: permite autenticar un usuario mediante su contraseña pero estos datos son trasmítidos cifrados mediante una función hash. Aunque esta acción no es la mejor opción, lo ideal sería almacenar la password en el servidor mediante la autenticación básica y encriptar la conexión mediante el módulo SSL. El módulo usado para este método es mod\_auth\_digest, que en Debian corresponde al fichero auth\_digest.load.

Por otro lado, si se necesita limitar los recursos a usuarios concretos o grupos de usuarios, se pueden usar los siguientes módulos:

- authz\_user: módulo que permitirá dar o denegar a uno o varios usuarios al recurso que se considere.
- authz\_groupfile: módulo que permitirá dar o denegar a un grupo de usuarios al recurso que se considere.

#### 4.5.1. Establecimiento del control de acceso

El control de acceso permite controlar el acceso por parte de un dispositivo de la red que desea acceder a un recurso determinado del servidor Apache. Este control se aplica sobre ficheros y directorios.

Este acceso se aplica dentro de la directiva de <Directory>, y dentro de ella se aplicará la directiva Require con sus diferentes opciones. A continuación, se mostrará una lista de las directivas más usadas:



**CUADRO 4.1**  
**Directiva Require**

Directiva	Descripción
Require all granted	Se permite el acceso a todo el mundo.
Require all denied	Se deniega el acceso a todo el mundo.
Require env env-var	Se permite el acceso solo a las variables de entorno definidas.
Require method http-method	Acceso a los métodos HTTP definidos.
Require expr expression	Acceso permitido al resultado verdadero de evaluar la expresión definida.
Requiere user userid	Solamente tienen acceso los usuarios definidos.
Require group group-name	Solamente tienen acceso los grupos definidos.
Require valid-user	Todos los usuarios validados tienen acceso al recurso.
Require ip	Los clientes que tengan como IP o rangos de IP tienen acceso al recurso.

Con algunos ejemplos de la tabla anterior se podrá comprobar su funcionamiento en los ficheros de configuración de Apache. Por ejemplo, si se quiere que nadie tenga acceso al directorio /var/www sería la definición como la siguiente:

```
<Directory "/var/www">
Require all denied
</Directory>
```

O que todo el mundo tenga acceso al directorio, tal y como se aprecia en la figura 4.20:

**Figura 4.20**  
Requiere all granted

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

Otra opción sería que solamente tenga acceso al directorio /var/www la red 192.168.10.0/24, esto significa que los equipos que tengan una IP en el rango 192.168.10.0 hasta 192.168.10.255 tendrían acceso. La implementación sería la siguiente:

```
<Directory "/var/www">
Require all denied
Require ip 192.168.10.0/24
</Directory>
```

**Recurso web**

Consulta la web de Apache para ampliar la información sobre la directiva Require con sus diferentes opciones.



#### 4.5.2. Autenticación básica

Se va a implementar una autenticación básica a partir de un host virtual anterior y con los siguientes datos:

- El directorio empresa-internet tendrá el nombre www.empresa.com que se ha configurado anteriormente.
- Ahora dentro del directorio /var/www/empresa-internet habrá uno que se llame rrhh y que solamente tendrá acceso el usuario admin y password rhat.123.

Para configurar tal autenticación se van a dar los siguientes pasos:

1. Se ha creado el fichero .htaccess dentro del directorio que se va a proteger, en nuestro caso, en la ruta /var/www/empresa-internet/rrhh (figura 4.21).

```
AuthType Basic
AuthName "Documentos de Empresa"
AuthUserFile "/etc/apache2/etc/.usuarios"
Require user admin
```

**Figura 4.21**  
Autenticación básica

Se va a analizar cada una de las opciones. AuthType es el tipo de autenticación, que en nuestro caso es básica. AuthName indica el texto que se visualizará en la ventana que solicita usuario y contraseña. AuthUserFile es el fichero de usuarios y contraseñas que permitirá validar al usuario. Require tiene varias opciones, en nuestro caso se ha puesto solamente que valide el usuario admin. También es válido indicar Require valid-user que permite validar cualquier usuario que esté en la lista del fichero.

2. Se podrá crear el fichero de usuarios que tendrá acceso al recurso rrhh, y para ello se usa el siguiente comando:

```
#htpasswd -c /etc/apache2/etc/.usuarios admin
```

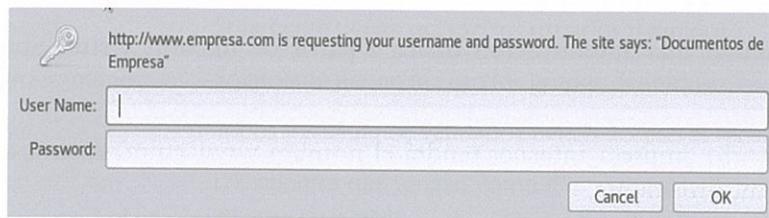
Este comando creará el fichero oculto .usuarios para dar más seguridad, y el usuario admin con la contraseña que pide la tarea. Otra opción sería poner la ubicación de este fichero en otra zona del servidor.

3. Ahora se procederá a configurar el host virtual de empresa para permitir el acceso al usuario a esta ubicación con la configuración que muestra la figura 4.22.

```
<Directory /var/www/empresa-internet/rrhh>
Options Indexes FollowSymlinks
AllowOverride All
Order allow,deny
Allow from all
</Directory>
```

**Figura 4.22**  
Configuración

La directiva que permitirá leer el fichero .htaccess será AllowOverride All. Posteriormente, se reinicia Apache con el comando #service apache2 restart y se obtendrá la pantalla de la figura 4.23.



**Figura 4.23**  
Pantalla de validación

4. Por último, se introduce el usuario que se comenta en la tarea, admin y password rhat.123, y se accederá perfectamente al recurso solicitado en el servidor web.



### Actividad propuesta 4.6

Configura una autenticación básica para el directorio /var/www/privado con la URL <http://www.móvil.com> y el usuario directiva y la password rhat.123. Comprueba mediante una página web que se accede a tal directorio.

#### 4.5.3. Autenticación de usuarios mediante LDAP

En este apartado se va a realizar la autenticación mediante un usuario del servidor de directorio de LDAP. Para llevar a cabo tal autenticación en Debian se pide la siguiente tarea:

- ✓ El directorio empresa-internet tendrá el nombre www.empresacom, que se ha configurado anteriormente.
- ✓ Ahora dentro del directorio /var/www/empresa-internet habrá uno que se llame LDAP y al que solamente tendrá acceso el usuario admin y password rhat.123.

Para configurar tal autenticación se van a dar los siguientes pasos:

1. Se han de habilitar dos módulos necesarios para realizar la autenticación con LDAP, y para ello se usarán los siguientes comandos:
  - a2enmod ldap.
  - a2enmod authnz\_ldap (aunque, si se habilita este, automáticamente se va a habilitar LDAP, porque depende de él).
2. Luego se va a proceder a configurar en el fichero empresa.conf para acceder al directorio rrhh. se habilitará este host virtual de nuevo para que se activen los cambios (figura 4.24).

```
<Directory /var/www/empresa-internet/rrhh>
AuthName "Área para administración"
AuthType Basic
AuthBasicProvider ldap
AuthLDAPBindAuthoritative Off
AuthLDAPURL "ldap://10.0.2.15:389/ou=usuarios,dc=cursolinux,dc=com"
Require valid-user
</Directory>
```

**Figura 4.24**  
Configuración LDAP

Como se puede observar en esta configuración de host virtual, hemos incluido la sección Directory con los siguientes campos:

- *AuthName*: permite poner un título a la ventana emergente que pide usuario y contraseña.

- *AuthType*: directiva que permite indicar qué tipo de autorización se va a usar, en nuestro caso, Basic.
- *AuthBasicProvider*: permite especificar el método de autenticación, que es LDAP.
- *AuthLDAPBindAuthoritative*: directiva que da la posibilidad de que otros módulos de autenticación puedan validar al usuario, en caso de fallo de LDAP.
- *AuthLDAPURL*: es la cadena LDAP que permite validar al usuario que se introducirá en la ventana emergente junto con la password. Se puede usar también en modo seguro.
- *Require valid-user*: permite validar a cualquier usuario que esté declarado en LDAP con su password correspondiente. Otra opción es poner un grupo de usuarios o un usuario específico.



**Figura 4.25**  
Comprobación LDAP

Esta configuración se puede realizar también en el fichero .htaccess del directorio que se va a limitar para que tenga un acceso limitado.

## 4.6. Certificados. Servidores de certificados

En esta unidad se han creado hosts virtuales para acceder al servidor web, así como carpetas para el acceso seguro. Pero en ningún momento se ha realizado una conexión segura mediante el protocolo HTTPS y basado en SSL (*Security Socket Layer*).



### TOMA NOTA

Si usamos este protocolo, la información irá cifrada y reducirá la amenaza de poder interceptar la información por parte de personal externo a nuestra organización.

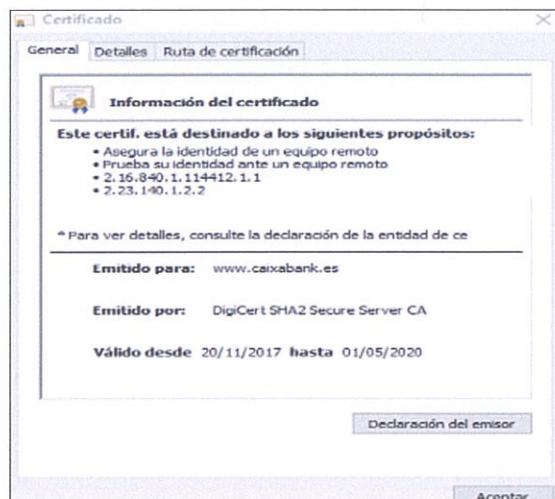
### 4.6.1. Módulo SSL para Apache

El método SSL permite establecer un canal seguro para la interacción entre el navegador del cliente y el servidor web a través de una red local o de Internet. La información irá encriptada y protegida e informará al usuario de que la conexión es segura. Se puede observar claramente porque en la barra del navegador aparece un candado que se puede pulsar o el protocolo https://. Si se pulsa el candado aparecerá el certificado y pulsando en él se observará una imagen como la de la figura 4.26.

El puerto que usa el método SSL es el 443, para diferenciarlo del puerto por defecto de http. El protocolo SSL es un método de clave pública de cifrado asimétrico. Esta comunicación se usa en portales web susceptibles de posibles ataques, como pueden ser la consulta de información confidencial, la banca online, la compra online, etc.

Todas estas comunicaciones tienen varios objetivos en común:

- ✓ Es necesario garantizar la confidencialidad de los datos que son transmitidos a través de Internet o de cualquier red. Es importante que la información sensible, como son los datos personales, no esté expuesta a entidades ajenas.
- ✓ Los portales de compra online o cualquier otra entidad deben cumplir las normas nacionales e internacionales de privacidad, seguridad e integridad de los datos.
- ✓ Todo tipo de empresa o entidad online debe garantizar la protección de la identidad de la persona y evitar a toda costa la suplantación de la misma.



**Figura 4.26**  
Certificado

#### 4.6.2. Servidor virtual seguro en Apache

Se va a demostrar con un host virtual, configurado anteriormente, la conexión segura mediante SSL. Para ello, se usará el host virtual del fichero empresa.conf que se ubicaba en el directorio /var/www/empresa-internet. Por lo tanto, nuestro objetivo es que tecleando `https://www.empresa.com` se visualice el fichero index.html y el servidor muestre que es un sitio seguro. Se comentan los pasos siguientes:

1. Se va a habilitar el módulo de SSL para Apache, si no está habilitado, de la siguiente forma:

```
#a2enmod ssl
```

2. En un paso posterior, se va a crear un directorio en el que se almacenarán los certificados con el siguiente comando:

```
#mkdir /etc/apache2/empresa/ssl
```

3. Despues, nos ubicamos en la carpeta anterior y creamos los certificados. El primero de ellos nos permite crear un certificado clave de SSL de 2048 bits de longitud. Nos pedirá una clave para un uso futuro; para generarla, se teclea el siguiente comando:

```
#openssl genrsa -des3 -out key 2048
```

4. Se crea el CSR (*Certificate Signing Request*) a partir de la clave privada creada. El comando solicitará una serie de datos, como la contraseña anteriormente indicada, la provincia, el país, etc. Se usa el siguiente comando para generar el fichero indicado:

```
#openssl req -new -key key -out empresa.csr
```

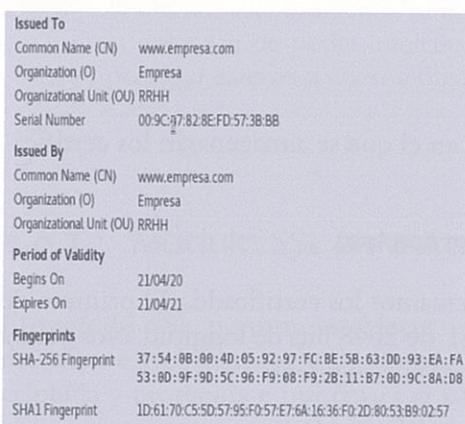
5. Se genera el certificado para nuestro sitio web. Nos pedirá de nuevo la clave para poder comprobar el fichero clave generado al principio. Se genera el certificado con el siguiente comando:

```
#openssl x509 -req -sha256 -days 365 -in empresa.csr  
-signkey key -out empresa.crt
```

6. Una vez generados los certificados, se va a configurar el host virtual con los siguientes parámetros:

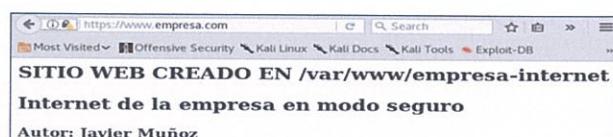
```
<VirtualHost 10.0.2.15:443>  
ServerAdmin admin@empresa.com  
DocumentRoot /var/www/empresa-internet  
ServerName www.empresa.com  
ServerAlias empresa.es  
SSLEngine on  
SSLCertificateFile /etc/apache2/empresa/ssl/empresa.crt  
SSLCertificateKeyFile /etc/apache2/empresa/ssl/key  
ErrorFile ${APACHE_LOG_DIR}/empresa-error-ssl.log  
CustomLogFile ${APACHE_LOG_DIR}/empresa-access-ssl.log  
LogFormat "%h %l %u %t \ %r\\" %>s" combined  
</VirtualHost>
```

7. Por último, se reiniciará el servicio de Apache con el comando *service apache2 restart*, y ya se está en disposición de comprobar que funciona perfectamente el acceso al host virtual de forma segura. Se mostrará, por un lado, el acceso al host virtual y, por otro, el certificado válido que nos indica que el sitio posee una conexión segura.



**Figura 4.27**

Certificado de la URL [www.empresa.com](https://www.empresa.com)



**Figura 4.28**

Comprobación [www.empresa.com](https://www.empresa.com)



## Actividad propuesta 4.7

Genera un certificado para la URL <https://www.móvil.com> y la ruta /var/www/privado. Comprueba que funciona mediante un página web index.html.

### 4.7. Pruebas de funcionamiento, monitorización y rendimiento del servidor web

A partir de que se haya instalado el servicio web es necesario mantenerlo para poder controlar que no tiene accesos malintencionados ni errores en los distintos hosts virtuales y en el core del servidor. Se van a tratar en este apartado los conceptos de registro y monitorización, las directivas que permiten controlar los registros de logs, acceso y errores. Por último, se harán pruebas de rendimiento del servidor mediante unos programas específicos.

#### RECUERDA

- ✓ Este apartado es crucial en el mantenimiento de un servidor web.

#### 4.7.1. Registro y monitorización

Esta tarea forma parte de una acción fundamental del mantenimiento del servidor Apache, que normalmente la realiza el administrador del sistema. Es necesario revisar los logs del servidor para comprobar si tienen, por ejemplo, errores de almacenamiento, de acceso no autorizado, etc.

En el fichero de configuración apache2.conf existen dos registros principales para controlar el servidor de los accesos y de los errores. Se va a explicar en qué consiste cada uno de ellos y se pondrá un ejemplo visual de los mismos.

- Registro de accesos: el fichero está en la ruta por defecto /var/log/apache2/access.log. En tal fichero se almacenan las peticiones solicitadas al servidor. De cada cliente registra la IP, la fecha y la hora de la petición, el tipo de petición, si es GET, POST, etc. Si se tienen varios hosts virtuales, lo ideal sería en cada uno configurar un registro de acceso para controlarlos por separado.

error.log	apache2.conf	empresa.conf	hosts	access.log
192.168.1.20 - - [08/Jul/2020:05:38:30 -0400] "GET / HTTP/1.1" 403 497 "-" "Mozilla/5.0 (X11; Linux x86_64; rv:52.0) Gecko/20100101 Firefox/52.0"				
192.168.1.20 - - [08/Jul/2020:05:38:30 -0400] "GET /favicon.ico HTTP/1.1" 403 496 "-" "Mozilla/5.0 (X11; Linux x86_64; rv:52.0) Gecko/20100101 Firefox/52.0"				
192.168.1.20 - - [08/Jul/2020:05:38:30 -0400] "GET /favicon.ico HTTP/1.1" 403 496 "-" "Mozilla/5.0 (X11; Linux x86_64; rv:52.0) Gecko/20100101 Firefox/52.0"				

**Figura 4.29**  
Ejemplo  
de registro  
de acceso

- b) Registro de errores: el fichero está en la ruta por defecto /var/log/apache2/error.log. En tal fichero se almacenan los errores provocados por una IP o por un error en la configuración del servidor y provoca que el servicio no pueda levantarse.

```
[Wed Jul 08 05:31:45.726999 2020] [ssl:emerg] [pid 1346:tid 140044876653696] AH02311: Fatal error initialising mod_ssl, exiting. See /var/log/apache2/empresa-error-ssl.log for more information
AH00016: Configuration Failed
[Wed Jul 08 05:34:56.059567 2020] [mpm_event:notice] [pid 1421:tid 140267720758400] AH00489: Apache/2.4.41 (Debian) OpenSSL/1.1.1d configured -- Resuming normal operations
[Wed Jul 08 05:34:56.059945 2020] [core:notice] [pid 1421:tid 140267720758400] AH00094: Command line: '/usr/sbin/apache2'
[Wed Jul 08 05:38:30.788764 2020] [authz_core:error] [pid 1422:tid 140267516552960] [client 192.168.1.20:57888] AH01630: client denied by server configuration: /home/web/www/
[Wed Jul 08 05:38:30.924363 2020] [authz_core:error] [pid 1422:tid 140267508160256] [client 192.168.1.20:57888] AH01630: client denied by server configuration: /home/web/www/favicon.ico
[Wed Jul 08 05:38:30.937921 2020] [authz_core:error] [pid 1422:tid 140267499767552] [client 192.168.1.20:57888] AH01630: client denied by server configuration: /home/web/www/favicon.ico
```

**Figura 4.30**  
Ejemplo de registro de error

Si se observa la figura 4.30, cada línea es un error registrado que está compuesto por la fecha y hora de cuándo se ha producido. Posteriormente, aparecerá el componente o módulo de Apache que ha generado el error, seguido de dos puntos y de la gravedad del mismo (emerg, notice, error, etc.). Se observará el pid (*process identifier*), esto es, identificador del proceso, y el tid (*thread identifier*), que significa el hilo del proceso. Después se puede visualizar la IP y el puerto, en caso de conexión del cliente [client 192.168.1.20:57888], y por último, la acción que se ha realizado. Por ejemplo, client denied by server configuration: /home/web/www/, se ha producido un acceso denegado del cliente cuya IP es 192.168.1.20 al recurso /home/web/www.

Como se ha comentado anteriormente, el mantenimiento de un servidor web como Apache es tan importante como configurarlo. Para realizar el mantenimiento y tener control de los accesos y los errores del servidor es necesario revisar y estudiar los logs que nos suministra el servicio Apache. Y con base en estos logs, tomar las medidas oportunas que permitan el buen funcionamiento del servicio.

Apache tiene un abanico de directivas que hacen posible la configuración de los archivos de logs. La información es almacenada por defecto en formato CLF (*Common Logon Format*). Este formato facilita el análisis de los logs, ya que todos los servidores web usan el mismo.

Otro método de análisis de logs sería el uso de scripts creados por el administrador, que permite, con un solo clic, comprobar si existe algún problema grave en el servidor. Se puede observar un tipo de acceso en formato CLF en la imagen 4.29 (cada línea es un acceso al servidor Apache).

```
192.168.1.20 - - [08/Jul/2020:05:38:30 -0400] "GET / HTTP/1.1"
403 497 "-" "Mozilla/5.0 (X11; Linux x86_64; rv:52.0)
Gecko/20100101 Firefox/52.0"
```

A continuación, se van a detallar los campos que componen una línea y pueden ser parametrizables. Cada campo que no posea valor se sustituirá por un guion. Se muestran algunos campos parametrizables en el siguiente cuadro:

**CUADRO 4.2**  
Campos del fichero de log

Campos	Descripción	Ejemplo de log
host(%h)	Indica la dirección IP del cliente que hizo la petición al servidor. Si está activada la directiva HostnameLookups, el servidor mostrará el nombre del host.	192.168.1.20
ident(%l)	Esta información está relacionada con la identidad de la máquina del cliente. Esta información es poco fiable, por lo que casi nunca se usa. El guion significa que la información no está disponible.	-
authuser(%u)	Identifica el usuario que solicita el recurso mediante la autenticación HTTP.	-
date(%t)	Muestra la fecha y hora en la que se ha realizado la petición del recurso al servidor.	[08/Jul/2020:05:38:30 -0400]
request(%r)	Indica la petición del cliente. Consta del método usado que es GET, el protocolo usado que es HTTP/1.1.	"GET / HTTP/1.1"
status(%s)	Es el código de estado que devuelve el servidor al cliente.	403
bytes(%b)	El número de bytes devueltos al cliente.	497

Los dos últimos de la fila están relacionados con el LogFormat, que se explicará en el siguiente apartado, pero permiten configurar las opciones de Referer y User-Agent. Estos campos están relacionados con el cliente que realiza la petición al servidor web Apache.

#### 4.7.2. Directivas para archivos de registro, log y error

En esta sección se detallarán las directivas que permiten controlar los registros de logs y de error del servidor Apache. Su objetivo final es controlar los accesos y el mantenimiento del servidor. Se pueden configurar de forma global o en particular dentro de cada configuración de los hosts virtuales que se crean. Se listan las más importantes directivas en el siguiente cuadro:

**CUADRO 4.3**  
Directivas de logs

Directivas	Descripción
LogFormat	Directiva que define el formato que tendrá el fichero de log. En esta directiva se podrán indicar los campos parametrizables detallados en el cuadro 4.2. Este formato de log será usado en la directiva TransferLog y CustomLog para poder customizar el fichero de log como se adapte a nuestras necesidades.
	[.../...]

**CUADRO 4.3 (CONT.)**

CustomLog	Directiva que permite almacenar en el fichero de log las solicitudes de acceso al servidor Apache. Esto es, cada petición HTTP será almacenada en este fichero de log, y además se permite personalizar los campos que pueda contener cada línea almacenada en el fichero de log.
CookieLog	Directiva que permite almacenar información sobre las cookies en un fichero. La ruta del fichero partirá del directorio que se configure en la directiva ServerRoot.
ErrorLog	Indica el nombre del fichero donde se almacenarán los mensajes de error generados por el servidor Apache. Se puede usar syslog en lugar del nombre del fichero siempre que el sistema lo soporte.
RewriteLog	Directiva que permite almacenar en un fichero los logs del servidor reescribiendo acciones realizadas.
ScriptLog	Establece un fichero para registrar los errores de script CGI. Si no se especifica esta directiva, no se creará ningún fichero de errores.
TransferLog	Directiva similar a la directiva CustomLog, con la excepción de que no permite especificar explícitamente el formato de log. Toma por defecto el formato definido en la directiva LogFormat.

**4.7.3. Pruebas de rendimiento del servidor web**

Para concluir esta sección se van a realizar una serie de pruebas de rendimiento sobre el servidor web con la herramienta ab (Apache Bench), que permite testear cómo reacciona el servidor web ante un número de peticiones y una serie de usuarios sobre una URL concreta alojada en nuestro servidor web. Por otro lado, existen bastantes herramientas gratuitas que permiten realizar pruebas de rendimiento. La herramienta ab permite medir el rendimiento del servidor web, y para que se observe cómo funciona y muestre los resultados, se va a comenzar con 1500 peticiones y 25 usuarios. Para ello se va a ejecutar el siguiente comando:

```
#ab -n 1500 -c 25 http://www.empresia.com/
```

Y se obtendrá la siguiente salida que permitirá visualizar el informe de la prueba que a partir de las 1500 peticiones y 25 usuarios se obtienen unos datos aceptables como el tiempo por solicitud de 5,234 milisegundos, con una media de 0,209 milisegundos de todas las solicitudes (figura 4.31).

Se va a incrementar la carga sobre el servidor web con una prueba de a partir de 150.000 peticiones y 50 usuarios, se obtienen unos datos

Server Hostname:	www.empresia.com
Server Port:	80
Document Path:	/
Document Length:	236 bytes
Concurrency Level:	25
Time taken for tests:	0.314 seconds
Complete requests:	1500
Failed requests:	0
Total transferred:	759000 bytes
HTML transferred:	354000 bytes
Requests per second:	4776.04 [/sec] (mean)
Time per request:	5.234 [ms] (mean)
Time per request:	0.209 [ms] (mean, across all concurrent requests)
Transfer rate:	2360.03 [Kbytes/sec] received
Connection Times (ms)	
	min mean[+/-sd] median max
Connect:	0 2 1.5 2 9
Processing:	1 3 1.4 2 10
Waiting:	0 2 1.4 2 8
Total:	3 5 1.8 4 11

**Figura 4.31**  
**Salida ab**

aceptables como el tiempo por solicitud de 8,752 milisegundos, con una media de 0,175 milisegundos de todas las solicitudes. Se puede concluir que el servidor web responde bien tanto a las peticiones como a los usuarios. También va a depender de la máquina hardware sobre la cual se instale el servidor web (figura 4.32).

```

Server Hostname: www.empresia.com
Server Port: 80

Document Path: /
Document Length: 236 bytes

Concurrency Level: 50
Time taken for tests: 26.257 seconds
Complete requests: 150000
Failed requests: 0
Total transferred: 75900000 bytes
HTML transferred: 35400000 bytes
Requests per second: 5712.70 [#/sec] (mean)
Time per request: 8.752 [ms] (mean)
Time per request: 0.175 [ms] (mean, across all concurrent requests)
Transfer rate: 2822.88 [Kbytes/sec] received

Connection Times (ms)
              min     mean[+/-sd] median     max
Connect:      0       4     1.3      4      19
Processing:   1       4     1.3      4      19
Waiting:      0       3     1.8      3      19
Total:        1       9     1.3      8      24

```

**Figura 4.32**  
Salida ab

## 4.8. Configuración de hosts virtuales en SO Windows

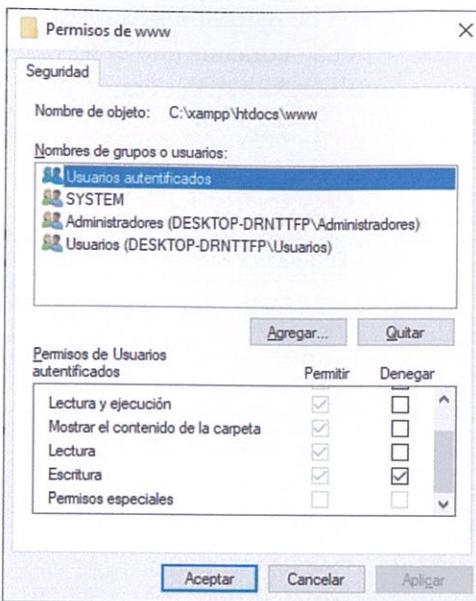
Se va a detallar la configuración de un host virtual en Windows. Como se observará, es parecida a la configuración que se ha realizado en el sistema operativo Linux, cambiando por ejemplo los procesos de Linux por servicios en Windows; las rutas de directorios también serían diferentes. Los datos serían los siguientes:

- La URL que se va a comprobar es www.empresia.com y que escuche por el puerto 90.
- La IP 192.168.1.3 que permitirá la resolución de la URL anterior.
- El directorio donde se va a ubicar la URL es c:\xampp\htdocs\www
- Los logs serán empresia.com-error.log y empresia.com-access.log

Los pasos para llevar a cabo la configuración de un host virtual son estos:

1. Al principio se creará el directorio que pone en la ruta y se le dará permisos de lectura y ejecución (ya los tiene). Se le deniega escritura, como se muestra en la figura 4.33.
2. Seguidamente se creará el fichero index.html para identificar que está funcionando el host virtual en la ruta anteriormente creada. Por otro lado, es necesario modificar el fichero de configuración httpd.conf con la siguiente línea, si no existiera:

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



**Figura 4.33**  
Permisos

3. Será necesario modificar el fichero httpd-vhosts.conf o cualquier otro que detalláramos en la línea anterior con las siguientes líneas:

```
<VirtualHost *:90>
    ServerAdmin webmaster@empresa.com
    DocumentRoot "C:/xampp/htdocs/www"
<Directory "c:/xampp/htdocs/www">
    Options +Indexes +Includes +FollowSymLinks +MultiViews
    AllowOverride All
    Require all granted
</Directory>
    ServerName www.empresa.com
    ErrorLog "logs/empresa.com-error.log"
    CustomLog "logs/empresa.com-access.log" common
</VirtualHost>
```

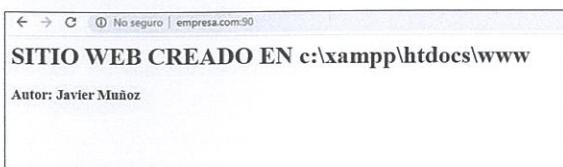
Por defecto, es el puerto 80, pero para nuestro ejemplo se configura por el puerto 90 y que no intervenga en otras configuraciones.

La nomenclatura de VirtualHost es <VirtualHost IP[:puerto] [IP][:puerto]....>. Hay que poner mínimo una IP, que puede ser asterisco, que significa todas las IP. El puerto es opcional y se pueden poner más IP y puertos.

4. En un paso posterior, se modifica el fichero C:\Windows\System32\drivers\etc\hosts, donde es necesario incluir la siguiente línea para que pueda resolver los nombres declarados en el fichero httpd-vhosts.conf. La línea es la siguiente:

192.168.1.3 (IP del servidor) www.empresa.com

5. Por último, se reinicia el servicio de Apache en Xampp y se accede a la página [www.empresa.com](http://empresa.com) con un navegador, por ejemplo Google Chrome, y se obtendrá una página como la de la figura 4.34.



**Figura 4.34**  
Comprobación

## Resumen

- En este capítulo se ha explicado cómo se administra un servidor web y todas las funcionalidades que posee.
- Se comenzó explicando los parámetros y las directivas de control más importantes del servidor, que permiten configurar el servidor de una forma óptima. Los primeros controlan el flujo del servidor y los segundos permiten distintas funcionalidades que configuran los hosts virtuales.
- En un segundo paso, se explican detalladamente los distintos hosts virtuales que existen y sus configuraciones en el sistema operativo Linux. Los dos más importantes son la configuración del host virtual basado en nombre y el host virtual basado en IP. El mixto se usa menos.
- Como se sabe, Apache es modular, y de ahí la importancia que posee de cara a la configuración de cualquier módulo, sin tener que compilar o ejecutar cada uno de sus componentes. Se basan en dos directorios mod-enabled y mod-available. Se explican los módulos más importantes. El usuario debe probar estos módulos para observar su funcionamiento.
- En un paso posterior, se profundiza en cómo se controla el acceso y la autenticación en los directorios indexados de Apache, junto con los certificados y el acceso seguro por parte de los usuarios. El acceso se puede realizar mediante una autenticación basic y digest.
- Después se analiza el servidor Apache con pruebas de funcionamiento y monitorización, así como una prueba de carga para comprobar cuántas conexiones soporta y usuarios al mismo tiempo, y su tiempo de respuesta. Este apartado es básico para controlar el funcionamiento del servidor y mejorar su funcionamiento del día a día.
- Por último, se configuró un host virtual en Windows. La configuración de Windows es prácticamente la misma que la de Linux. Es necesario tener en cuenta que en Windows se usan servicios, en lugar de procesos, como se hace en Linux.

## Supuestos prácticos

1. Aprovechando la instalación de Apache, comprueba el funcionamiento de algunos parámetros del servidor, como son los siguientes:
  - DirectoryIndex.
  - ErrorDocument.
  - DocumentRoot.
2. Comprueba algunas directivas del servidor Apache, como las siguientes:
  - ErrorLog.
  - Ports.conf (cambiar de puerto de escucha y comprobar que funciona).
  - ServerName.
3. Encuentra un módulo de Apache que no esté instalado, instálalo y comprueba que funciona.
4. Genera un error en el servidor Apache y comprueba que el fichero log se modifica notificando el error.
5. ¿Cómo se configura en Apache que solamente se acepten peticiones de la red 172.16.1.0-172.16.2.255 para un determinado host virtual basado en IP?
6. Explica cómo se configura un host virtual de forma segura, indicando los pasos que se han de realizar.



## Ejercicios propuestos

1. Una vez que está instalado Apache en Linux, configura un virtual host basado en nombre con las siguientes características:
  - a) Crea un directorio que se llame panarte en la ruta /var/www
  - b) Este directorio debe permitir el acceso a las URL: www.panarte.com y panarte.local con la IP 192.168.3.10
  - c) Los logs se deben configurar de acceso y de error, denominados panarte-access.log y panarte-error.log, respectivamente. El log debe ser combined.
  - d) El virtual host debe escuchar por el puerto 90.
  - e) Es necesario crear una página web para demostrar que funciona.
2. A partir de la instalación de Apache en Linux, configura un virtual host basado en IP con las siguientes características:
  - a) Se partirá de dos IP diferentes: 192.168.1.34 y 192.168.1.35
  - b) Crea un directorio que se llame cafe en la ruta /var/www
  - c) Este directorio debe permitir el acceso a partir de las URL: www.cafe.es (192.168.1.34), relacionada con Internet, y www.intranet.cafe.es (192.168.1.35), relacionada con Intranet.
  - d) Los logs se deben configurar de acceso y de error. Se denominan cafe-access.log y cafe-error.log, respectivamente. El log debe ser combined.

- e) El virtual host debe escuchar por el puerto 9090.
  - f) Es necesario crear una página web para cada IP para demostrar que funciona.
3. Configura un virtual host basado en nombre en el sistema operativo Windows a través de Xampp con los siguientes datos:
- a) Crear un directorio que se llame algodon en la ruta \var\www.
  - b) Este directorio debe permitir el acceso a partir de la ruta www.algodon.es
  - c) Los logs se deben configurar de acceso y de error. Se denominan algodon-access.log y algodon-error.log, respectivamente. El log debe ser combined.
  - d) El virtual host debe escuchar por el puerto 8080.
  - e) Es necesario crear una página web para demostrar que funciona.
4. Realiza una autenticación básica para el primer ejercicio. Solamente aquellos que conozcan el usuario rrhh y la password cat.t123 podrán acceder a /var/www/panarte/direccion.
5. Configura una autenticación mediante LDAP, creando un directorio activo (visto en capítulos anteriores) y la ruta /var/www/panarte/ldap. Tendrá acceso solamente el usuario rrhh y la password cat.123. El dominio y el administrador de LDAP es de libre designación y el usuario puede crear el que prefiera.

## ACTIVIDADES DE AUTOEVALUACIÓN

1. ¿Qué es una directiva dentro de los ficheros de Apache?:
  - a) Es un puerto de configuración que permite escuchar peticiones.
  - b) Es una palabra reservada que permite realizar una función específica dentro de la configuración de Apache.
  - c) Es un comando que se puede ejecutar en la línea de comandos.
  - d) Ninguna de las respuestas anteriores es correcta.
2. ¿Cuántos tipos de hosts virtuales se pueden configurar?:
  - a) 3.
  - b) 2.
  - c) 4.
  - d) 5.
3. ¿Qué comando permite listar los módulos cargados en Apache?:
  - a) apache -M.
  - b) apachect -M.
  - c) apachectl -m.
  - d) apachectl -M.
4. ¿Qué es un host virtual basado en nombre?:
  - a) Es un host virtual que posee varios nombres y varias IP asociados a los mismos.
  - b) Es una configuración que permite acceder solamente por un nombre.

- c) Es un host virtual que tiene asignado solamente una IP y varios nombres asociados que acceden por la misma IP.  
 d) Es un host virtual que tiene asignado solamente una IP y un solo nombre.

5. ¿Qué comando permite deshabilitar un site en Apache?:

- a) a2enconf.  
 b) a2dissite.  
 c) a2ensite.  
 d) a2disconf.

6. ¿Qué comando permite habilitar un módulo en Apache?:

- a) a2enconf.  
 b) a2dismod.  
 c) a2enmod.  
 d) a2disconf.

7. ¿Cuál es el fichero de configuración por defecto de XAMPP para configurar los virtual hosts?:

- a) httpd-vhosts.conf.  
 b) http-vhosts.conf.  
 c) httpd-vhost.conf.  
 d) httpd-hosts.conf.

8. ¿Qué directiva permite realizar una autenticación básica?:

- a) AuthName.  
 b) Auth.  
 c) AuthorizationType.  
 d) AuthType.

9. ¿Qué comando se usa para generar un certificado de 2048 bits?:

- a) keycert.  
 b) openssl.  
 c) sslopen.  
 d) Ninguna de las respuestas anteriores es correcta.

10. ¿Qué comando permite habilitar un site en Apache?:

- a) a2enconf.  
 b) a2dissite.  
 c) a2ensite.  
 d) a2disconf.

### SOLUCIONES:

1. **a** **b** **c** **d**

2. **a** **b** **c** **d**

3. **a** **b** **c** **d**

4. **a** **b** **c** **d**

5. **a** **b** **c** **d**

6. **a** **b** **c** **d**

7. **a** **b** **c** **d**

8. **a** **b** **c** **d**

9. **a** **b** **c** **d**

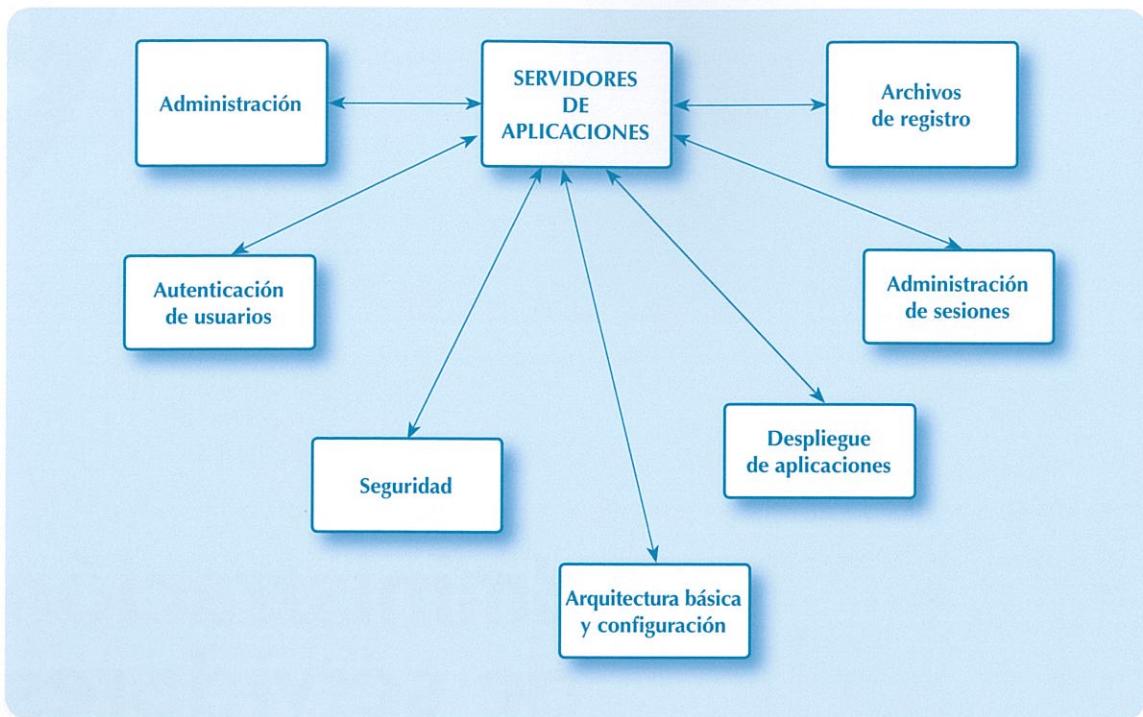
10. **a** **b** **c** **d**

# Administración de servidores de aplicaciones

## Objetivos

- ✓ Describir los componentes y el funcionamiento de los servicios proporcionados por el servidor de aplicaciones.
- ✓ Identificar los principales archivos de configuración y de bibliotecas compartidas.
- ✓ Configurar el servidor de aplicaciones para cooperar con el servidor web.
- ✓ Configurar y activar los mecanismos de seguridad del servidor de aplicaciones.
- ✓ Configurar y usar los componentes web del servidor de aplicaciones.
- ✓ Realizar los ajustes necesarios para el despliegue de aplicaciones sobre el servidor.
- ✓ Llevar a cabo pruebas de funcionamiento y rendimiento de la aplicación web desplegada.
- ✓ Elaborar documentación relativa a la administración y recomendaciones de uso del servidor de aplicaciones.

## Mapa conceptual



## Glosario

**Fichero .war (Web Application Archive).** Es un fichero comprimido que contiene todos los componentes web de una aplicación basada en Java y que se pueden ejecutar en un servidor de aplicaciones.

**Java ServerPages.** Es una tecnología basada en Java para crear páginas dinámicas y que se usan habitualmente en servidores de aplicaciones como Apache-Tomcat o GlassFish.

**JDK (Java Development Kit).** Es un kit de desarrollo que provee herramientas muy útiles para la implementación de programas en Java.

**JNDI (Java Naming Directory Interface).** Es una interfaz que permite localizar información en distintos directorios en plataformas distribuidas, por ejemplo LDAP.

**Realm.** Es el concepto que usa Tomcat para referirse a los dominios de seguridad, y que permite controlar la autenticación por parte de los usuarios.

**Servlets.** Son programas escritos en Java sin interfaz gráfica que permiten escuchar peticiones a partir del protocolo HTTP.

**SSL (Secure Socket Layer).** Es un protocolo de cifrado que se usa para garantizar la seguridad de las comunicaciones, sobre todo a través de Internet.

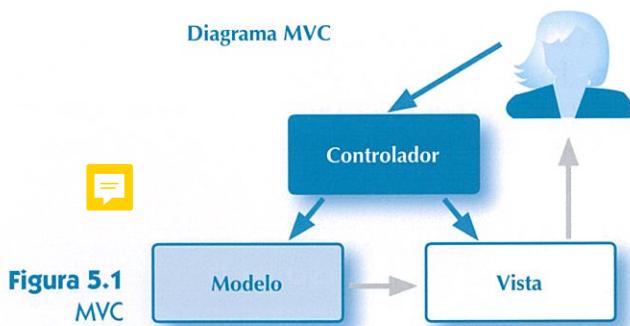
## 5.1. Introducción

El objetivo de este capítulo es, a su vez, el objetivo del libro, el despliegue de una aplicación en un servidor de aplicaciones. Para ello es crucial tener en cuenta algunos aspectos, como el descriptor de despliegue, el servidor de aplicaciones donde se va a desplegar la aplicación, la seguridad de la aplicación, cómo se van a autenticar los usuarios, la administración de sesiones, los registros de logs de la aplicación y todo lo relacionado con la misma.

A lo largo del capítulo se irán desgranando cada uno de los aspectos que posee un rol clave en el despliegue de la aplicación.

## 5.2. Arquitectura y configuración básica del servidor de aplicaciones

La arquitectura que se emplea en este tipo de aplicaciones es un patrón software que separa la lógica de negocio y los datos de la parte representativa que observa el usuario, los eventos y las comunicaciones entre los distintos componentes. A este modelo se le denomina *Modelo-Vista-Controlador (MVC)*, y se basa en la separación de módulos para su posterior mantenimiento y reutilización de código, además de la facilidad de detectar errores en caso de fallos. Es importante separar la representación de los datos (Vista) de la parte interna o modelo de datos, como, por ejemplo, la base de datos. Como se puede observar en la siguiente figura:



Los tres grandes componentes en los que se divide el modelo son los siguientes:

- Modelo*: es el componente que se encarga de representar la información con la que la aplicación trabaja. Por lo que su función es la de realizar las consultas y modificaciones con base en los privilegios definidos previamente en el análisis de requisitos de la aplicación. La petición llega por parte del controlador y este componente ejecuta la acción y la presenta a la *Vista*.
- Vista*: visualiza el modelo con un tipo de representación para interactuar con el usuario. Normalmente es la interfaz de la aplicación, ya que puede ser una aplicación web, aplicación de escritorio, o cualquier aplicación en cualquier entorno o sistema operativo.
- Controlador*: es el módulo más importante, como su propio nombre indica, que controla a los otros dos componentes. Responde a peticiones realizadas por el usuario (pulsar un botón de la interfaz), y partir de ahí se comunica con el modelo para ma-

nipularlo haciendo cambios en la vista. Se puede concluir que es el mediador entre el *Modelo* y la *Vista*.

Un servidor de aplicaciones es un software global que permite dar servicio a las aplicaciones que se publican en el mismo, dando soporte para la seguridad, para las transacciones, administración de sesiones, registro de logs, etc.

Existe un abanico de servidores de aplicaciones que son usados en el tejido productivo empresarial. Se van a comentar los más usados:

- ✓ *WildFly (JBoss Application Server)*: es un software implementado por JBoss y está desarrollado en Java. Está disponible para todos los sistemas operativos del mercado y está basado en los proyectos punteros de software open source. Si se analiza la estructura interna, está basado en dos núcleos principales:
  - *JBoss Module*: controla la carga de los recursos de la clase contenedor.
  - *Modular Service Container*: administra los servicios usados por el contenedor, por ejemplo, instala y desinstala los diferentes servicios.
- ✓ *GlassFish*: es un software de código abierto desarrollado por Sun Microsystems para la plataforma Java EE. Se encuentra disponible la versión GlassFish 5.0.1 con Java EE 8. Este proyecto open source proporciona un proyecto estructurado de desarrollo que permite tener disponibles las nuevas funcionalidades de Java.
- ✓ *JOnAs*: es un software libre indicado para servidores de aplicaciones desarrollado por el consorcio ObjectWeb.
- ✓ *Apache-Tomcat*: es un servidor de aplicaciones open source implementando Java Servlet, JavaServer Pages, Java Expression Language and Java WebSocket technologies. Estos módulos son desarrollados por Java Community Process. Oracle certifica con la versión de Java correspondiente. La versión estable de Apache-Tomcat es la 9.0.37 con JDK 8.

WWW

### Recurso web

Consulta las páginas web de GlassFish y Apache-Tomcat para ampliar la información y para descargar la última versión.



#### 5.2.1. Instalación del servidor de aplicaciones Apache-Tomcat

El servidor de aplicaciones que se va a instalar y configurar es Apache-Tomcat, por ser uno de los más usados en el mundo empresarial, también por su facilidad de uso y por su aceptación en la comunidad de desarrolladores de software.

Apache-Tomcat es un contenedor de Servlets que se puede usar para compilar y ejecutar aplicaciones realizadas en Java. Tomcat puede funcionar de manera autónoma pero usualmente se puede utilizar con otros productos para permitir una mayor compatibilidad con las distintas tecnologías del mercado, y de esta forma incrementar su funcionalidad.

Para que se observe un pequeño servlet, sería el siguiente:

```
public class Prueba extends HttpServlet {  
  
    // Maneja el método GET de HTTP para una página web  
  
    public void doGet (HttpServletRequest request, HttpServletResponse response)  
        throws ServletException, IOException {  
        PrintWriter salida;  
        String titulo = "Servlet de salida";  
  
        // Selecciona los contenidos y otros campos  
  
        response.setContentType("text/html");  
        // Luego escribe los datos de la respuesta  
        salida = response.getWriter();  
        salida.println("<HTML><HEAD><TITLE>");  
        salida.println(titulo);  
        salida.println("</TITLE></HEAD><BODY>");  
        salida.println("<H1>" + titulo + "</H1>");  
        salida.println("<P>Esto es una salida del Servlet.");  
        salida.println("</BODY></HTML>");  
        salida.close();  
    }  
}
```

Es crucial conocer cómo funciona un contenedor de servlets, ya que es el funcionamiento de Apache-Tomcat. Para ello se explica en la siguiente lista y gráfico:

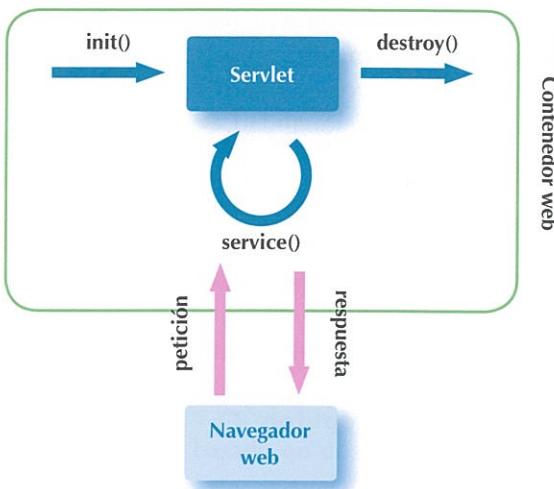
- El navegador web del cliente solicita una página al servidor HTTP.
- El contenedor de servlets procesa la petición y la asigna al servlet apropiado.
- El servlet elegido es el encargado de generar el texto de la página web y entregarla al contenedor de servlets.
- El contenedor devuelve la página web al navegador del cliente.

Por otro lado, además de servlets, se pueden ejecutar en la capa web más componentes, como, por ejemplo, Java Server Pages (JSP). Es un software que permite generar páginas web dinámicas principalmente (también genera otro tipo de documentos, como xml, etc.). Estos archivos .jsp se compilan y se transforman en un servlet.



### Actividad propuesta 5.1

Crea una página JSP para observar cómo funciona, por ejemplo, en el IDE Netbeans.



**Figura 5.2**  
Funcionamiento del contenedor de servlets

Después de estas ideas iniciales se va a proceder a la instalación de Apache-Tomcat en el sistema operativo Linux y configuración inicial para luego desplegar aplicaciones. La instalación se puede realizar desde el repositorio o manual. La instalación que se va a llevar a cabo es manual para que se observe toda la configuración completa; es la siguiente:

1. A lo largo del tiempo, Apache va sacando a la luz diferentes versiones de Tomcat; la versión última es la 10, pero está en fase de test alpha. Por lo tanto, se va a instalar la versión más estable, que es Tomcat 9, con la versión de JDK 8, ya que es compatible. Primero se empieza con la versión de JDK. Para ello se observa si existe alguna versión instalada con el siguiente comando. Y se obtiene la versión de la figura 5.3.

```
#java -version  
#javac -version
```

```
root@osboxes:/etc# java -version  
openjdk version "1.8.0_151"  
OpenJDK Runtime Environment (build 1.8.0_151-8u151-b12-1-b12)  
OpenJDK 64-Bit Server VM (build 25.151-b12, mixed mode)  
root@osboxes:/etc# javac -version  
javac 1.8.0_151  
root@osboxes:/etc#
```

**Figura 5.3**  
Versión de JDK actual

WWW

### Recurso web

Se recomienda consultar esta página web de Apache-Tomcat para observar la compatibilidad de Apache-Tomcat con JDK que nos permitirá instalar los dos componentes sin ningún problema.



2. Puede ser que no haya ninguna versión instalada de JDK. Tras esto es necesario descargar la versión 8 o posterior compatible con Apache-Tomcat. Para ello se selecciona la URL <https://www.oracle.com/java/technologies/javase/javase-jdk8-downloads.html> y la opción Linux de 64 bits. Para descargarse el JDK es necesario crear una cuenta en Oracle que automáticamente te permitirá descargar el fichero que en nuestro caso sería *jdk-8u261-linux-x64.tar.gz*.



El directorio /opt es para los paquetes de software que no son instalados en la ruta predeterminada.

3. Una vez descargado y colocado en el directorio /opt, se crea el directorio JDK y se descomprime con el siguiente comando:

```
#mkdir /opt/jdk  
#tar -zxf jdk-8u261-linux-x64.tar.gz -C /opt/jdk
```

4. Se va a configurar para que el sistema configure la versión 8.261 de JDK en el sistema por defecto. Para ello se teclean los siguientes comandos:

```
#update-alternatives --install /usr/bin/java java /opt/jdk/jdk1.8.0_261/bin/java 100  
#update-alternatives --install /usr/bin/javac javac /opt/jdk/jdk1.8.0_261/bin/javac 100
```

Por último, en la configuración de JDK se ejecutan los siguientes comandos, lo que permitirá usar la versión descargada en Apache-Tomcat y cualquier aplicación que use Java, como se puede observar en la figura 5.4.

```
#update-alternatives --config java  
#update-alternatives --config javac
```

5. Una vez terminada la instalación de JDK, se procede a instalar Apache-Tomcat. Primero hay que descargar la versión binaria para Linux desde la URL <https://tomcat.apache.org/download-90.cgi> y seleccionar Binary Distributions y Core.
6. Apache-Tomcat 9.0.37 se coloca en el directorio opt. Se crea un directorio llamado tomcat9 y se descomprime con el siguiente comando:

```
#mkdir /opt/tomcat9  
#tar xzf apache-tomcat-9.0.37.tar.gz -C /opt/tomcat9
```

```

root@osboxes:/opt# java -version
openjdk version "1.8.0_151"
OpenJDK Runtime Environment (build 1.8.0_151-8u151-b12-1-b12)
OpenJDK 64-Bit Server VM (build 25.151-b12, mixed mode)
root@osboxes:/opt# update-alternatives --config java
Existen 2 opciones para la alternativa java (que provee /usr/bin/java).

  Selección   Ruta                               Prioridad  Estado
* 0           /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/java  1081      modo automático
  1           /opt/jdk/jdk1.8.0_261/bin/java               100       modo manual
  2           /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/java  1081      modo manual

Pulse <Intro> para mantener el valor por omisión [*] o pulse un número de selección: 1
update-alternatives: utilizando /opt/jdk/jdk1.8.0_261/bin/java para proveer /usr/bin/java (java)
do manual
root@osboxes:/opt# java -version
java version "1.8.0_261"
Java(TM) SE Runtime Environment (build 1.8.0_261-b12)
Java HotSpot(TM) 64-Bit Server VM (build 25.261-b12, mixed mode)

```

**Figura 5.4**  
Configuración de JDK 1.8.0\_261

7. Posteriormente, se configuran las variables de entorno de JDK para que automáticamente se carguen en Apache-Tomcat cuando arranque el servicio. Para ello se teclean las siguientes líneas:

```

#echo 'export JAVA_HOME="/opt/jdk/jdk/jdk1.8.0_261"' > /etc/profile.d/tomcat9.sh
#echo 'export JRE_HOME="/opt/jdk/jdk/jdk1.8.0_261/jre"' >> /etc/profile.d/tomcat9.sh

```

El fichero /etc/profile.d/tomcat9.sh permitirá que cada vez que arranque la máquina física o virtual el componente de JDK se cargue sin ser necesario configurarlo de forma manual.

Ya se está preparado para arrancar Apache-Tomcat con el siguiente fichero. Se observará una pantalla como la de la figura 5.5.

```
#/opt/tomcat9/apache-tomcat-9.0.37/bin/startup.sh
```

```

root@osboxes:~# /opt/tomcat9/apache-tomcat-9.0.37/bin/startup.sh
Using CATALINA_BASE:   /opt/tomcat9/apache-tomcat-9.0.37
Using CATALINA_HOME:   /opt/tomcat9/apache-tomcat-9.0.37
Using CATALINA_TMPDIR: /opt/tomcat9/apache-tomcat-9.0.37/temp
Using JRE_HOME:        /opt/jdk/jdk1.8.0_261/jre
Using CLASSPATH:       /opt/tomcat9/apache-tomcat-9.0.37/bin/bootstrap.jar:/opt/tomcat9/apache-tomcat-9.0.37/bin/tomcat-juli.jar
Tomcat started.

```

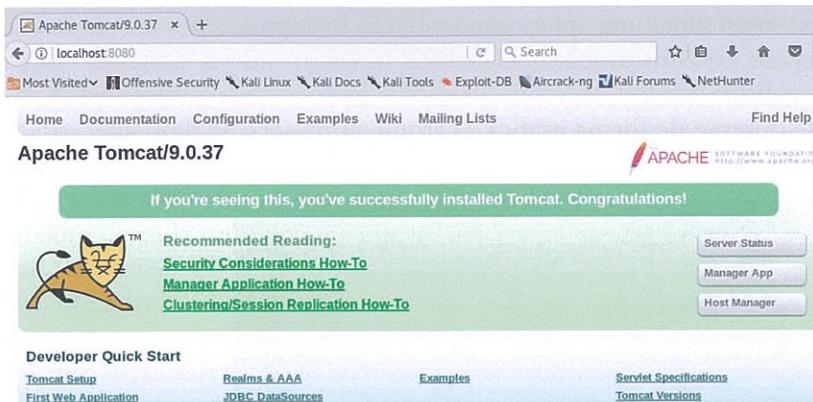
**Figura 5.5**  
Inicio de Apache-Tomcat

8. Por último, se puede comprobar que Apache-Tomcat está funcionando correctamente. Abriendo el navegador y tecleando <http://localhost:8080> o [http://IP\\_equipo:8080](http://IP_equipo:8080), se mostrará una pantalla como de la figura 5.6.

### Actividad propuesta 5.2



Instala JDK y Tomcat en una máquina virtual o física de la distribución Linux Debian y comprueba que funciona correctamente.



**Figura 5.6**  
Inicio de Apache-Tomcat



Se puede comprobar que no hay errores en la configuración de Apache-Tomcat 9.0.37 con el comando:

```
#/opt/tomcat9/apache-tomcat-9.0.37/bin/configtest.sh.
```

### 5.2.2. Arquitectura de Apache-Tomcat y variables de entorno

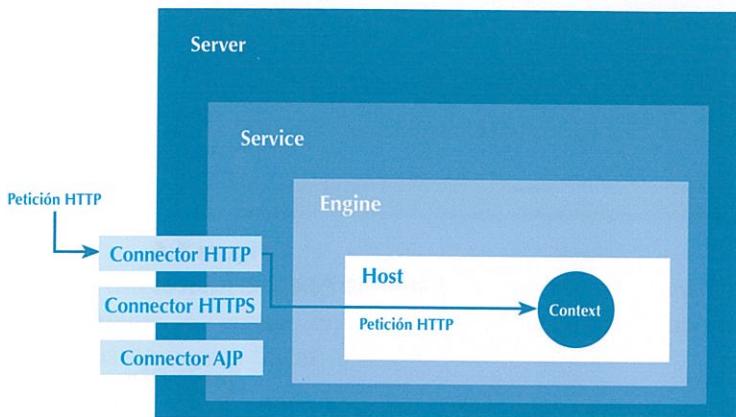
Una vez instalado el servidor de aplicaciones Apache-Tomcat se va a analizar su estructura para que el programador o usuario de este servicio pueda comprender el funcionamiento a grandes rasgos de la aplicación.

Apache-Tomcat usa una arquitectura que permite de forma lógica conectar sus diferentes componentes y que cada uno realice una función determinada en la estructura arbórea. Por ello se van a listar los componentes que posee este servidor de aplicaciones:

- a) *Server*: es el componente que representa el contenedor al completo y dentro de él se ejecutan los demás componentes. Posee una interfaz gráfica que prácticamente no se customiza.
- b) *Service*: es un componente intermedio que se comunica y que está dentro del componente server, que posee varias conexiones.
- c) *Engine*: es el servicio que permite procesar las peticiones que provienen desde el exterior y responder a tales solicitudes. Por ello, posee múltiples conectores que hacen posible resolver las peticiones.
- d) *Host*: es un nombre de red que puede poseer varios hosts o alias, como pueden ser www.daw.com o www.daw.es.
- e) *Connector*: es el encargado de controlar las comunicaciones con el cliente. Tomcat incluye varios conectores, como AJP Connector o HTTP Connector.

- f) *Context:* representa una aplicación web. Un host puede poseer varios o múltiples contextos pero con una única ruta.

Para que se observe de forma gráfica, la arquitectura sería la siguiente:



**Figura 5.7**  
Arquitectura Apache-Tomcat 9

### Actividad propuesta 5.3



Investiga cada uno de los componentes de la arquitectura Tomcat, defínelos y observa qué tags se encuentran dentro de ellos.

Anteriormente, se instaló Apache-Tomcat bajo la ruta /opt/tomcat9/apache-tomcat-9.0.37, que es el directorio que se ha descomprimido y descargado de Internet. A partir de esa localización se encuentran los siguientes directorios:

- ✓ *bin:* es el directorio que contiene los ficheros binarios y los scripts de inicio, tanto en sistema Windows como en Linux. Los dos ficheros importantes en Linux son startup.sh (inicio de Apache-Tomcat) y shutdown.sh (Parada de Apache-Tomcat).
- ✓ *conf:* este directorio es uno de los más importantes, ya que permite la configuración global del servidor de aplicaciones. A continuación, se listarán y explicarán cada uno de estos archivos, cruciales para el funcionamiento de Apache-Tomcat:
  - *catalina.policy:* la política de seguridad de Apache-Tomcat se realiza a partir de este fichero y está relacionada con Java. Una entrada en este fichero sería de la siguiente forma:

```
// Example policy file entry
```

```
grant [signedBy <signer>,] [codeBase <code source>] {
    permission <class> [<name> [, <action list>]];
};
```

- *catalina.properties*: es el fichero en el que se relacionan los ficheros .jar de Java para la clase Catalina. La información está relacionada con los paquetes de seguridad y las rutas de las clases cargadas. También contiene algunas configuraciones de las cadenas más usadas en la caché.
- *context.xml*: este fichero xml contiene la información de contexto común a todas las aplicaciones web que se ejecuten en Tomcat. También permite localizar el archivo web.xml de cada una de las aplicaciones.
- *jaspic-properties.xml*: es un fichero que permite definir un módulo de autenticación diferente a JAAS, pero con sus peculiaridades.
- *jaspic-properties.xsd*: es el esquema XSD que define si es válido el fichero anterior xml, esto es, su estructura, sus componentes, qué tipos de datos, etc.
- *logging.properties*: este fichero permite configurar las funciones de logging de todas las aplicaciones y de la actividad del servidor. Es crucial para poder solucionar cualquier error en caso de fallo del servidor o de una aplicación.
- *server.xml*: es el fichero principal de configuración de Tomcat, que contiene un gran número de parámetros, de ahí su complejidad. Por ello se van a numerar los parámetros que usan dentro de él:

**CUADRO 5.1**  
Parámetros del *server.xml*

Marca	Descripción	Ejemplo
<code>&lt;server&gt;</code> <code>&lt;/server&gt;</code>	Es la etiqueta principal que engloba a toda la configuración del fichero. Engloba a uno o más servicios, y contiene al atributo port, que permite definir el puerto por el que escucha.	<code>&lt;Server port="8005"</code> <code>shutdown="SHUTDOWN"&gt;</code>
<code>&lt;Listener .../&gt;</code>	Para definir las extensiones JMX ("Java Management Extensions") que serán usadas por Apache-Tomcat. Tienen un atributo principal llamado className.	<code>&lt;Listener className="org.apache.catalina.startup.VersionLoggerListener"/&gt;</code>
<code>&lt;GlobalNamingResources&gt;</code> ..... <code>&lt;/GlobalNamingResources&gt;</code>	Esta marca define los recursos tipo JNDI para usarlos globalmente en el servidor de aplicaciones. Usa la etiqueta Resource para especificar la localización.	<code>&lt;Resource name="UserDatabase"</code> <code>auth="Container"</code> <code>type="org.apache.catalina.UserDatabase"</code>  <code>description="User database</code> <code>that can be updated and saved"</code> <code>factory="org.apache.catalina.users.MemoryUserDatabaseFactory"</code> <code>pathname="conf/tomcat-users.xml" /&gt;</code>

[.../...]

## CUADRO 5.1 (CONT.)

<Service> .... </Service>	Esta marca permite agrupar uno o más conectores. Si se teclea Catalina se ejecutará el servidor como independiente.	<Service name="Catalina">
<Connector />	Representa las conexiones TCP desde el exterior que serán abiertas cuando arranca el servidor. Uno de los principales es el HTTPConnector.	<Connector port="8090" protocol="HTTP/1.1" connectionTimeout="20000" redirectPort="8444" />
<Engine> .... </Engine>	Se usan dentro de la marca Service o de Host. Se procesan las peticiones que llegan a la marca Connector y que la cabecera disponga el Host por defecto.	<Engine name="Catalina" defaultHost="localhost">
<Logger/>	Esta marca indicará dónde serán enviados los registros de logs.	<Logger className="org.apache.catalina.logger.FileLogger" directory="logs" prefix="localhost_log." suffix=".txt" timestamp="true"/>
<Host> ... </Host>	A partir de esta etiqueta se pueden definir varios hosts virtuales para atender las peticiones.	<Host name="localhost" appBase="webapps" unpackWARs="true" autoDeploy="true">
<Context> ... <Context>	Indicará la ruta a partir de la cual se encontrarán las aplicaciones ejecutadas en Tomcat. Normalmente se encuentran debajo del directorio webapps.	

- *tomcat-users.xml*: este fichero contiene los usuarios, contraseñas y roles que serán usados para acceder al servidor Apache-Tomcat. Existen algunas líneas con comentarios que pueden ser descomentadas para ponerlas en funcionamiento.
- *tomcat-users.xsd*: es el esquema XSD que define si es válido el fichero anterior xml, esto es, su estructura, sus componentes, qué tipos de datos, etc.
- *web.xml*: es un fichero estándar para las aplicaciones web, común a todas las aplicaciones web, ya que posee la configuración global a todas ellas.

- ✓ *lib*: este directorio contiene todos los ficheros .jar usados en el servidor que corresponden a Tomcat, a JSP, etc.
- ✓ *logs*: el directorio donde se almacenarán los logs de Catalina y de las aplicaciones.
- ✓ *temp*: es el directorio donde se almacenan los temporales del servidor Tomcat.

- ✓ *webapps*: directorio que contiene todas las aplicaciones web. Solamente existe una que se denomina ROOT.
- ✓ *work*: directorio de almacenamiento temporal de ficheros, por ejemplo, de compilación.



### Actividad propuesta 5.4

Crea un usuario admin y password admin.1234 en el fichero tomcat-users.xml y dale permisos de administrador de aplicaciones.

Con relación a las variables de entorno que usa Apache-Tomcat en el arranque para que pueda funcionar este servidor, se pueden comentar las siguientes:

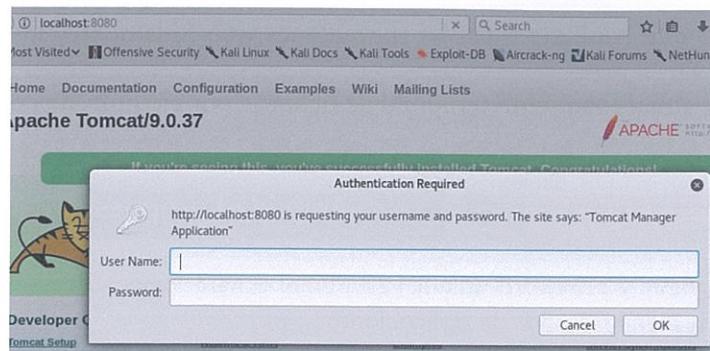
- CATALINA\_BASE: es el directorio que representa la configuración de una instancia de Tomcat, normalmente coincide con el valor de la variable CATALINA\_HOME. Solamente se diferencia en el caso de que existan varias instancias en la misma máquina.
- CATALINA\_HOME: indica el directorio raíz de la instalación del servidor. En nuestro caso es /opt/tomcat9/apache-tomcat-9.0.37, en Windows podría ser algo parecido a C:\Program Files\ tomcat9\apache-tomcat-9.0.37.
- CATALINA\_TMPDIR: es el directorio temporal de Apache-Tomcat donde se pueden almacenar ficheros de compilación, intermedios, etc.
- JRE\_HOME: almacena la ruta donde se encuentran los ejecutables de Java que usa Apache-Tomcat.
- CLASSPATH: es el conjunto de rutas que usa Apache-Tomcat, sobre todo los ficheros .jar.

## 5.3. Administrar aplicaciones web

Una vez explicada la arquitectura de Apache-Tomcat es el momento de administrar las aplicaciones web que soporta el servidor. Para ello, una vez iniciado el servicio de Tomcat se debe teclear en el navegador la siguiente URL: <http://localhost:8080/manager/html> o <http://IP:8080/manager/html> (la IP que aparezca al ejecutar el comando ifconfig). También, se puede acceder mediante la URL <http://localhost:8080> y después pulsando en el botón *Manager App*. De las dos formas se observará una pantalla como la de la figura 5.8.

Según se puede ver en la figura 5.8, está solicitando un usuario y una contraseña que es configurable dentro del fichero tomcat-users.xml. Por defecto, el acceso al administrador de aplicaciones está deshabilitado. Para entrar es necesario autenticar a un usuario. Para habilitar un usuario es necesario definirlo mediante el atributo username, password y roles. El rol que permite manejar las aplicaciones es manager-gui. Por lo tanto, si se añaden las siguientes líneas en el fichero de usuarios de Tomcat, esto permitiría administrar las aplicaciones del servidor:

```
<user username="tomcat" password="cat.123" roles="manager-gui"/>
```



**Figura 5.8**  
Aplicaciones

Una vez introducidas estas líneas en el fichero \$CATALINA\_HOME/conf/tomcat-users.xml, ya se puede introducir el usuario tomcat y la contraseña cat.123 que permitirá visualizar una pantalla como la de la figura 5.9.

Como se puede observar en la siguiente pantalla, se tiene una fila por cada aplicación desplegada con una serie de opciones (figura 5.10).

Path	Version	Display Name	Running	Sessions	Commands
/	None specified	Welcome to Tomcat	true	0	<a href="#">Start</a> <a href="#">Stop</a> <a href="#">Reload</a> <a href="#">Undeploy</a> <a href="#">Expire sessions</a> with idle ≥ 30 minutes
/docs	None specified	Tomcat Documentation	true	0	<a href="#">Start</a> <a href="#">Stop</a> <a href="#">Reload</a> <a href="#">Undeploy</a> <a href="#">Expire sessions</a> with idle ≥ 30 minutes

**Figura 5.9**  
Administrar aplicaciones

/examples	None specified	Servlet and JSP Examples		true	0	<a href="#">Start</a> <a href="#">Stop</a> <a href="#">Reload</a> <a href="#">Undeploy</a> <a href="#">Expire sessions</a> with idle ≥ 30 minutes
-----------	----------------	--------------------------	--	------	---	------------------------------------------------------------------------------------------------------------------------------------------------------

**Figura 5.10**  
Opciones de la aplicación

Las opciones son las siguientes:

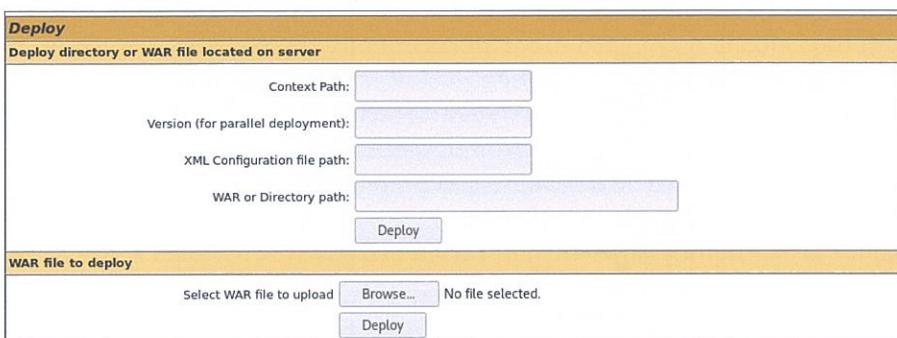
- ✓ *Path*: es el nombre del directorio a partir del cual se encuentra la estructura e información de la aplicación desplegada.
- ✓ *Version name*: es el número de versión de la aplicación, que va incrementando a medida que se van realizando modificaciones a la aplicación.
- ✓ *Display Name*: el nombre de la aplicación que se va a desplegar.
- ✓ *Running*: si está funcionando en el momento que se accede al administrador de aplicaciones. Tiene dos valores: true y false.

- ✓ **Sessions:** si existe alguna sesión abierta, indica el número con la aplicación.
- ✓ **Commands:** son los comandos que nos permiten controlar la aplicación. Existen cuatro comandos en la parte de arriba de la columna. En nuestro caso, la aplicación está iniciada, y permite pararla con el botón de *Stop*, recargarla mediante *Reload* y eliminar el despliegue de la aplicación con *Undeploy*. Por último, permite controlar el tiempo que una sesión puede estar en espera, y a partir del cual se elimina la sesión, en nuestro caso es de 30 minutos.

**RECUERDA**

- ✓ La opción *Undeploy* borrará la aplicación que exista con el nombre en el directorio de Tomcat; normalmente se llama webapps para el virtual host correspondiente.

En el siguiente apartado se puede observar el despliegue de una aplicación de forma automática y a partir de un fichero .war (figura 5.11).



**Figura 5.11**  
Despliegue de una aplicación

Si se observa, existen dos opciones para desplegar una aplicación, la primera forma indicando la ruta del contexto de la aplicación, la versión de la misma, la ruta del fichero .xml de la configuración y la ruta del directorio de la aplicación o el fichero .war.

La segunda opción sería subiendo al administrador de aplicaciones el fichero .war. El fichero .war se puede generar de varias formas. La primera de ellas sería ubicarse en el directorio de la aplicación y posteriormente ejecutar el siguiente comando:

```
#jar -cvf app.war *
```

Las opciones del comando .jar son las siguientes

\*c: crea un fichero sobre la salida estándar.

\*d: visualiza la información sobre el fichero creado, tamaño, modificación, etc.

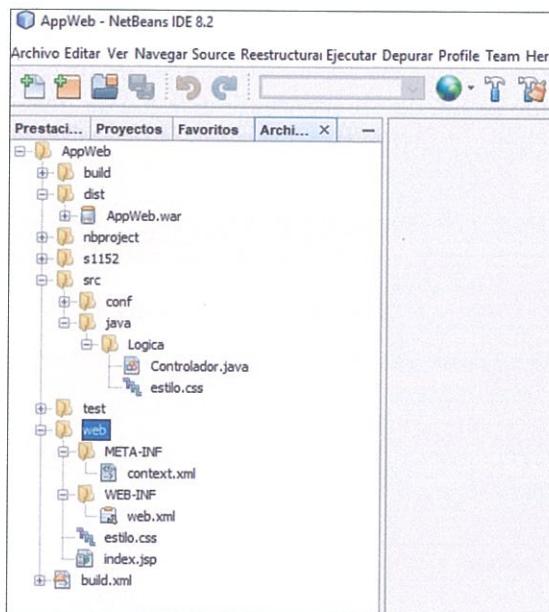
\*f: este argumento permite especificar el fichero .jar que se va a procesar.

## Actividad propuesta 5.5



Crea una aplicación de ejemplo o descárgala de alguna URL y desplégala previamente creada con el comando jar.

Otra forma sería a partir de un entorno de desarrollo de programación de Java, por ejemplo Netbeans IDE 8.2. Una vez que se tiene la aplicación web funcionando y probada, se ejecuta la opción *Ejecutar* y dentro de esta la opción *Limpiar y generar Proyecto*. Se generaría un fichero llamado como la aplicación terminado en .war dentro de la estructura de directorios de la aplicación en el directorio dist. Se puede observar en la figura 5.12.

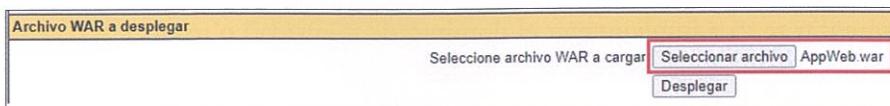


**Figura 5.12**  
Netbeans

Por último, se va a generar un fichero .war de la aplicación anterior y, a continuación, se va a desplegar para que se observe el funcionamiento. Se genera el fichero Appweb.war y se carga en la opción que existe en el administrador de aplicaciones en el apartado *WAR file to deploy*. Además, es necesario el permiso de admin-gui para poder desplegar aplicaciones.

```
<user username="tomcat" password="cat.123" roles="manager-gui, admin-gui"/>
```

Tal como se observa en la figura 5.13, y se pulsa el botón Deploy. Al desplegarla, se visualizará en el administrador de aplicaciones. Ahora, si se quiere comprobar que funciona, es necesario teclear la URL <http://localhost:8080/AppWeb/> o [http://IP\\_host\(192.168.1.20\):8080/AppWeb/](http://IP_host(192.168.1.20):8080/AppWeb/) y se observará una imagen como la de la figura 5.14:



**Figura 5.13**  
Despliegue



**Figura 5.14**  
URL aplicación

## 5.4. Autenticación de usuarios. Dominios de seguridad para la autenticación

El siguiente paso en la aplicación es segurizarla y permitir la autenticación de usuarios. Para ello Tomcat tiene varios métodos de autenticación, todos basados en el concepto de interfaz de Java llamado *realm* (dominio de seguridad). Este concepto engloba a los usuarios, contraseñas y roles que permitirán decidir quién accede y quién no accede a la aplicación. Tal concepto se configura en el fichero web.xml (explicado anteriormente).

Tomcat proporciona la configuración de este conjunto de roles, usuarios y contraseñas en el fichero tomcat-users.xml, pero se pueden implementar de otras formas, como almacenarse en una base de datos externa, almacenarse en un fichero .xml, comprobar la validación en el directorio LDAP o por el método de autenticación propio de Java denominado JAAS. Los seis plugins que soporta Apache-Tomcat son los siguientes:

- *JDBCRealm*: autenticación vía JDBC driver a una base de datos relacional.
- *DataSourceRealm*: autenticación vía JNDI JDBC DataSource a una base de datos relacional.
- *JNDIRealm*: autenticación mediante JNDI provider a un directorio activo LDAP.
- *UserDatabaseRealm*: se autentica mediante el acceso a la información almacenada en un recurso UserDatabase JNDI, normalmente un fichero .XML (tomcat-users.xml).
- *MemoryRealm*: se autentica mediante el acceso a la información almacenada en memoria, que es configurado desde un fichero .XML (tomcat-users.xml).
- *JAASRealm*: autenticación basada en un framework de Java llamado JAAS (Java Authentication & Authorization Service).

En esta sección se van a explicar el método de autenticación JAAS (propio de Java) y el método UserDatabaseRealm. Una de las características fundamentales de este servidor de aplicaciones es la autenticación HTTP. Este tipo de autenticación puede no ser segura al 100%, ya que el usuario y la contraseña introducidos por el usuario viajan por la red, y en estos tiempos que corren no es buena opción. Aunque siempre existe la posibilidad de implementar el protocolo seguro HTTPS.

Para comenzar se va a implementar el dominio de seguridad o realm UserDatabaseRealm, y para ello es necesario seguir las siguientes instrucciones:

1. Se parte de una aplicación de ejemplo llamada Bufete que posee una serie de carpetas y ficheros para gestionar la aplicación. Primero, es necesario modificar las siguientes etiquetas dentro del fichero llamado *web.xml*, en la carpeta WEB-INF de la aplicación:

- ✓ *<welcome-file-list>*: esta marca permite que se listen los ficheros iniciales de la aplicación, que normalmente se nombra como index con las extensiones .html, xhtml, .htm o .jsp.
- ✓ *<security-constraint>*: esta etiqueta especifica el acceso a los recursos de la aplicación, que en nuestro caso son carpetas y contenidos.
- ✓ *<http-method>*: esta etiqueta especifica uno de los métodos HTTP que permiten interactuar al cliente con el servidor. Entre los cuales están GET, POST, PUT, DELETE, HEAD, etc.
- ✓ *<auth-constraint>*: este tag define roles para acceder a los recursos.
- ✓ *<login-config>*: esta marca permite especificar el tipo de autenticación:
  - BASIC: con este valor no es necesario definir un formulario, ya que Tomcat visualiza una ventana para introducir usuario y contraseña.
  - FORM: la aplicación debe poseer un formulario diseñado por el programador cumpliendo una serie de reglas, que posteriormente se explicarán.

Un ejemplo de esta estructura se puede observar en las siguientes imágenes:

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
    http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"
  version="3.1"
  metadata-complete="true">
  <display-name>Bufete</display-name>
  <welcome-file-list>
    <welcome-file>index.html</welcome-file>
    <welcome-file>index.xhtml</welcome-file>
    <welcome-file>index.htm</welcome-file>
    <welcome-file>index.jsp</welcome-file>
  </welcome-file-list>

  <security-constraint>
    <display-name>Acceso a la directiva</display-name>
    <web-resource-collection>
      <web-resource-name>Area protegida</web-resource-name>
      <!-- Define los recursos protegidos -->
      <url-pattern>/secure/*</url-pattern>
      <!-- Métodos HTTP protegidos -->
      <http-method>DELETE</http-method>
      <http-method>GET</http-method>
      <http-method>POST</http-method>
      <http-method>PUT</http-method>
    </web-resource-collection>
    <auth-constraint>
      <role-name>DIRECTIVA</role-name>
    </auth-constraint>
  </security-constraint>
  <security-role>
    <role-name>DIRECTIVA</role-name>
  </security-role>
</web-app>
```

**Figura 5.15**  
Fichero I Web.xml

**Figura 5.16**  
Fichero Web.xml

```
<login-config>
    <auth-method>BASIC</auth-method>
    <realm-name>Ejemplo de autenticacion</realm-name>
    <form-login-config>
        <form-login-page>/secure/login.jsp</form-login-page>
        <form-error-page>/secure/error.jsp</form-error-page>
    </form-login-config>
</login-config>
```



### Actividad propuesta 5.6

Configura una aplicación con el método BASIC y comprueba que funciona, te permitirá ir adquiriendo conocimientos de la estructura de TOMCAT.

- En el siguiente paso es necesario modificar el fichero de usuarios, contraseñas y roles que se encuentran en la siguiente ruta tomcat9/conf/tomcat-users.xml. Se define un rol llamado DIRECTIVA, un usuario admin y una contraseña para entrar en la aplicación Bufete. Como se puede observar en la figura 5.17.

```
<?xml version="1.0" encoding="UTF-8"?>
<tomcat-users xmlns="http://tomcat.apache.org/xml"
               xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
               xsi:schemaLocation="http://tomcat.apache.org/xml tomcat-users.xsd"
               version="1.0">

    <!-- Administrar Tomcat -->
    <user username="tomcat" password="cat.123" roles="manager-gui,admin-gui"/>
    |   <!-- Aplicación Bufete -->
    <role rolename="DIRECTIVA"/>
    <user username="admin" password=".123456pt" roles="DIRECTIVA"/>
</tomcat-users>
```

**Figura 5.17**  
Fichero tomcat-users.xml

- Ahora se necesita comprobar en el fichero de tomcat/conf/server.xml que la siguiente línea está descomentada y activa:

```
<Realm className="org.apache.catalina.realm.LockOutRealm">
<Realm className="org.apache.catalina.realm.UserDatabaseRealm"
      resourceName=UserDatabase" />
</Realm>
```

La clase org.apache.catalina.realmLockOutRealm permite un mecanismo de bloqueo en caso de que se realicen demasiadas conexiones fallidas en un periodo de tiempo corto. Previene los ataques desde el exterior usando usuarios que no son válidos.

- Para otra configuración en el contexto de la aplicación, esto es, en el directorio Bufete/META-INF/context.xml, sería conveniente incluir las siguientes líneas:

```
<Context>
<Realm className="org.apache.catalina.realm.UserDatabaseRealm"
       resourceName=UserDatabase"/>
</Context>
```

Por último, sería necesario reiniciar Apache-Tomcat para que se lleven a efecto los cambios realizados en cada uno de los ficheros mediante los siguientes comandos:

```
#tomcat/bin/shutdown.sh
#tomcat/bin/startup.sh
```

Como se comentó anteriormente, se va a implementar otro de los dominios de seguridad o realm *JAASRealm* de la aplicación Bufete. Este dominio de seguridad es una implementación de la interfaz Tomcat Realm que sirve para autenticar usuarios usando el framework de autenticación y autorización JAAS.

*JAASRealm* permite al programador implementar un mecanismo de seguridad propio basado en las interfaces JAAS Login Module y JAAS Principal.

Y para ello es necesario seguir las siguientes instrucciones:

1. Hay que programar nuestras clases para LoginModule, User y Role, lo que permitirá validar al usuario y darle acceso a los recursos. Aunque en esta parte nos vamos a centrar en que el usuario valide y entre en la aplicación.
2. Una vez que estén implementadas las clases, se deberían compilar estas con el comando *java nombreclase.java*, que generará un fichero .class, que habría que poner en el CLASSPATH de Tomcat para que al arrancar el servidor de aplicaciones conozca de su ubicación.
3. Es necesario crear un fichero de configuración de JAAS llamado jaas.config para indicar qué Login Module sería necesario usar (puede existir más de uno). Sería recomendable poner este fichero en la ruta \$CATALINA\_HOME/conf. Un ejemplo de archivo sería el siguiente:

```
gestion{
PlainLoginModule required debug=true;
};
```

4. Otro paso posterior sería en la variable \$JAVA\_OPTS que permitiría indicar al servidor de aplicaciones dónde está el fichero de configuración anterior. Es necesario crear un script para poder ejecutarlo cada vez que se arranque Tomcat. Se denomina setenv.sh y debería tener la siguiente línea:

```
Export      JAVA_OPTS=-Djava.security.auth.login.config==$CATALINA_
BASE /conf/jaas.config
```

5. Dentro de la aplicación, es preciso indicarle cuál es el contexto, lo que se realiza a través del fichero \$CATALINA\_HOME/webapps/Bufete/META-INF/context.xml. Habría que incluir las siguientes líneas:

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<Context>
<Realm className="org.apache.catalina.realm.JAASRealm"
    appName="gestion"
    userClassNames="PlainUserPrincipal"
    roleClassNames="PlainRolePrincipal"
    />
</Context>
```

6. En un paso posterior es necesario indicarle a Tomcat las siguientes líneas dentro del motor (Engine) de Tomcat. La aplicación va a usar la autenticación Realm y sería de la siguiente forma:

```
<Realm className="org.apache.catalina.realm.JAASRealm"
    appName="gestion"
    userClassNames="PlainUserPrincipal"
    roleClassNames="PlainRolePrincipal"
    />
```

7. Posteriormente, hay que configurar el fichero \$CATALINA\_HOME/webapps/Bufero/WEB-INF/web.xml. Este fichero tiene dos partes bien diferenciadas, una relacionada con la autenticación y otra con la autorización. Para la parte de autenticación sería necesario poner las siguientes líneas:

```
<login-config>
    <auth-method>FORM</auth-method>
    <realm-name>Formulario de autenticacion</realm-name>
    <form-login-config>
        <form-login-page>/secure/login.jsp</form-login-page>
        <form-error-page>/secure/error.jsp</form-error-page>
    </form-login-config>
</login-config>

<welcome-file-list>
    <welcome-file>index.html</welcome-file>
    <welcome-file>index.xhtml</welcome-file>
    <welcome-file>index.htm</welcome-file>
    <welcome-file>index.jsp</welcome-file>
</welcome-file-list>
```

Y para la parte de autorización es necesario incluir las líneas relacionadas con los roles y las rutas donde serán aplicados tales roles, así como los métodos http que estén habilitados:

```
<security-role>
    <role-name>DIRECTIVA</role-name>
<security-role>
    <security-constraint>
        <display-name>Seguridad para RECURSOS</display-name>
```

```

<web-resource-collection>
    <web-resource-name>Area protegida</web-resource-name>
    <!-- Define the context-relative URL(s) to be protected -->
    <url-pattern>/secure/*</url-pattern>
    <!-- Métodos HTTP -->
        <http-method>DELETE</http-method>
        <http-method>GET</http-method>
        <http-method>POST</http-method>
        <http-method>PUT</http-method>
    </web-resource-collection>
    <auth-constraint>
        <!-- Roles asignados -->
        <role-name>DIRECTIVA</role-name>

    </auth-constraint>
</security-constraint>

```

8. A continuación, el fichero login.jsp, que debería contener dos campos obligatorios, que serían Usuario y Contraseña. Estos campos se deben denominar j\_username y j\_password y, además, la acción del formulario debe ser j\_security\_check con el método POST para mostrar los datos de identificación en la barra del explorador. Todo esto es para que se cumpla con el estándar JAAS. El formulario de inicio debería ser algo parecido al siguiente:

```

<form method="POST" action='<%= response.encodeURL("j_security_
check") %>' >
    <table border="0" cellspacing="5">
        <tr>
            <td align="right">Usuario:</td>
            <td align="left"><input type="text" name="j_username"></td>
        </tr>
        <tr>
            <td align="right">Clave:</td>
            <td align="left"><input type="password" name="j_password"></
td>
        </tr>
        <tr>
            <td colspan="2" align="center"><input type="submit" val-
ue="Enviar">
                <input type="reset" value ="Borrar"></td>
            </tr>
    </table>
</form>

```

9. Por último, después de todas las modificaciones, estaríamos preparados para parar Tomcat y arrancarlo de nuevo.

## 5.5. Administración de sesiones. Sesiones persistentes

En esta sección se va a explicar la administración de sesiones de las diferentes aplicaciones que se están ejecutando en un instante determinado en Apache-Tomcat y cómo se pueden configurar las sesiones persistentes para controlar el acceso malintencionado desde el exterior.

Para comprobar cómo funcionan las sesiones, se arranca Tomcat y se teclea la URL: `http://localhost:8080/manager/html` o `http://IP:8080/manager/html` (la IP que aparezca al ejecutar el comando `ifconfig`). Una vez dentro de la interfaz del servidor, se selecciona la opción *Manager App*, que solicitará un usuario y contraseña, que ya se definió antes (usuario `tomcat` y password `cat.123`), y se llegará a la pantalla del administrador de aplicaciones.

Esta vez nuestro objetivo es la columna Sessions. Si se observa, cada aplicación tiene un número. Si no existen conexiones a la aplicación pondrá un 0 y si existe alguna conexión o varias se visualizará el número de conexiones, que pueden ser activas o inactivas. Para las inactivas existe una opción que permite finalizar dicha conexión pasado un tiempo determinado.

Si se pulsa, por ejemplo, en la sesión de aplicación manager, que es la sesión activa nuestra, se muestra una imagen como la de la figura 5.18.

Active HttpSession informations									
<a href="#">Refresh Sessions list</a> 1 active Sessions									
Session Id	Type	Guessed Locale	Guessed User name	Creation Time	Last Accessed Time	Used Time	Inactive Time	TTL	
6C6B8A3FCF206BB669BACE4F41A4EEF3	Primary		tomcat	2020-08-11 12:39:18	2020-08-11 13:08:22	00:30:04	00:00:00	00:29	

Figura 5.18  
Sesiones

La información de las sesiones HTTP activa es la siguiente:

- *Session id*: es una cadena de 32 caracteres (números y letras) en hexadecimal que identifica plenamente a una sesión que es creada por el administrador de aplicaciones de Tomcat. La longitud se crea usando el atributo `sessionIdLength` que define el número de bytes aleatorio, que, por defecto, es 16.
- *Type*: es el tipo de sesión, dependiendo de si el acceso es primario o es con base en un acceso primario, que en ese caso sería secundario.
- *Guessed Locale*: equipo desde el cual se ha generado la petición, es configurable.
- *Guessed User name*: el nombre del usuario que ha generado la sesión de la aplicación.
- *Creation time*: es el momento de creación de la sesión.
- *Last Accessed time*: es el atributo que indica el último acceso a la sesión.
- *Used time*: el periodo de tiempo que está usando la sesión recursos del servidor de aplicaciones.
- *Inactive time*: es el tiempo de inactividad de la sesión.
- *TTL*: es el tiempo de vida que tiene la sesión, que si está inactiva va descontando de los 30 minutos que posee. Y si a partir de 30 minutos no tiene actividad, expira la sesión. Este parámetro es configurable en el administrador de aplicaciones.

Además, se puede pulsar sobre el ID de una sesión de una aplicación y se obtendrá una pantalla como la de la figura 5.19.

The screenshot shows a web interface titled 'Details for Session ABA3BB0609DF702044EF1633026AE57D'. It displays session details such as Session Id (ABA3BB0609DF702044EF1633026AE57D), Guessed Locale (Guessed User (tomcat), Creation Time (2020-08-12 13:29:16), Last Accessed Time (2020-08-12 13:30:42), Session Max Inactive Interval (00:30:00), Used Time (00:14:56), Inactive Time (00:00:00), and TTL (00:29:59). Below this is a 'Refresh' button. A '1 ATTRIBUTES' section follows, containing a table with one row: 'Remove Attribute' (button), 'Attribute name' (org.apache.catalina.filters.CSRF\_NONCE), and 'Attribute value' (org.apache.catalina.filters.CsrfPreventionFilter\$LRuCache@850186c). There is also a 'Remove' button in this row. At the bottom is a 'Return to session list' button.

**Figura 5.19**  
Sesión específica

Con relación a las sesiones persistentes, se pueden configurar y deshabilitar dependiendo de nuestro objetivo en la administración de aplicaciones Apache-Tomcat. Las sesiones de las aplicaciones alojadas en nuestro servidor de aplicaciones están configuradas por defecto como persistentes. Esto quiere decir que, en caso de fallo o pérdida de conexión, esas conexiones no se pierden. Esto hay que valorarlo en entornos de producción 24×7, donde el servicio debe ser la máxima prioridad.

Por otro lado, las sesiones inactivas pueden ser configuradas para que se puedan almacenar en disco en caso de fallo o pérdida de conexión. Esto hay que controlarlo porque puede provocar fallos en el sistema en caso de colapso del almacenamiento.

La configuración de las sesiones persistentes puede ser en general para todas las aplicaciones o para una en particular, dependiendo del nivel o el contrato de servicio suscrito con la empresa o usuario de la aplicación. Los tags o marcas que intervienen son <Context> y <Manager>. Manager tiene que estar dentro de Context.

Por una parte, si se quiere configurar la sesión persistente de forma global sería necesario modificar el fichero \$CATALINA\_HOME/conf/context.xml.

Un ejemplo de configuración con los parámetros puede ser el siguiente:

```
<Context>
    <Manager className="org.apache.catalina.session.PersistentManager"
        saveOnRestart="true"
        miniIdleSwap="0"
        maxIdleSwap="60"
        maxActiveSession="5"
        maxIdleBackup="5">

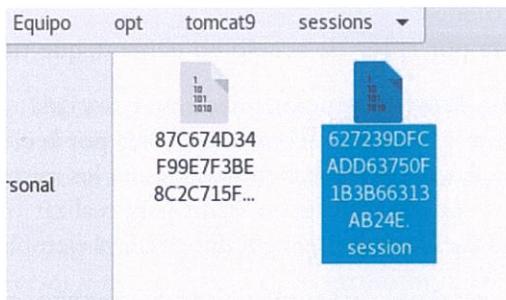
        <Store className="org.apache.catalina.session.FileStore"
            checkInterval="60"
            directory="/opt/tomcat9/sessions"
        />
    </Manager>
</Context>
```

Se van a comentar los parámetros implementados en el fichero context.xml de Tomcat:

- ✓ *saveOnRestart*: permite guardar todas las sesiones cuando el servidor reinicie.

- ✓ *maxActiveSession*: es el límite a partir del cual se guardan las sesiones en disco.
- ✓ *minIdleSwap*: establece el límite mínimo en segundos antes de que una sesión pueda almacenarse en disco.
- ✓ *maxIdleSwap*: establece el límite máximo en segundos antes de que una sesión pueda almacenarse en disco.
- ✓ *maxIdleBackup*: es el tiempo en segundos que pasa desde el último acceso de la sesión antes de que la sesión sea almacenada. La sesión no es eliminada de la memoria.
- ✓ *checkInterval*: es el intervalo en segundos que transcurre desde que las sesiones están inactivas hasta que se almacenan en disco. Por defecto son 60 segundos.
- ✓ *directory*: es la ruta que se especifica para que se almacenen las sesiones.

Si se configuran estas opciones en el fichero, se arranca Tomcat y se ejecuta alguna aplicación. Luego, se podrá comprobar en el directorio sessions las sesiones que se han almacenado, como en la pantalla de la figura 5.20.



**Figura 5.20**  
Sesiones almacenadas

Por otra parte, se pueden implementar estas opciones en una aplicación en particular y, más concretamente, en el fichero context.xml de la aplicación. Para tener controlada la configuración sería necesario generar un directorio por cada aplicación que englobe Tomcat.

Por último, se pueden deshabilitar las sesiones persistentes modificando el fichero \$CATALINA\_HOME/conf/context.xml y reiniciando el servidor TOMCAT de la siguiente forma:

```
<Manager pathname="" />
```



### Actividad propuesta 5.7

Configura una sesión persistente de cualquier aplicación y demuéstralos con los ficheros idóneos.

## 5.6. Archivos de registro de acceso y filtro de solicitudes

Como en cualquier servidor de aplicaciones o de cualquier ámbito es fundamental la configuración de los logs de administración del servidor. En el servidor de Tomcat es básico configurar los ficheros de acceso y de funcionamiento de las peticiones de los clientes, ya sea en el servidor en general o en cualquiera de las aplicaciones que estén desplegadas.

Los logs de acceso se generan por las peticiones que se realizan por parte de los clientes al servidor de aplicaciones. La información va desde la IP que realiza la petición, el tipo de petición, la fecha de la petición, etc. Todos estos campos son configurables.

La configuración de estos registros se puede configurar tanto en la aplicación como en el servidor de aplicaciones, dependiendo de nuestras necesidades.

Para llegar al objetivo de configurar los registros de acceso, Tomcat usa el concepto de Válvulas (Valve), que en la versión 4 de Tomcat se utilizó por primera vez y son propias del servidor. Estas válvulas se permiten asociar a una instancia de Java del contenedor global CATALINA.

Esta configuración permite el preprocesamiento de las peticiones que se realizan al servidor, por lo que no se puede configurar en otros servlets que no sean el principal. El componente Valve de la versión 9.0.37 de Tomcat tiene una ingente de opciones, se va a intentar ver las más interesantes.



#### TOMA NOTA

Antes de llevar a cabo cualquier modificación en los ficheros de configuración .xml es necesario realizar una copia de tales ficheros, por si hay que volver hacia atrás.

Se comienza por el log de acceso en general, que tiene las dos opciones siguientes:

1. *Access Log Valve*: esta opción puede estar asociada a cualquier contenedor de Catalina (Context, Host o Engine). Está implementada por la clase “org.apache.catalina.valves.AccessLogValve”. Además, crea ficheros de logs en los mismos formatos que los servidores web, por lo que se puede realizar un script para realizar un rastreo de los posibles problemas que existan en el servidor de aplicaciones. Un ejemplo de esta opción podría ser el siguiente:

```
<Valve className="org.apache.catalina.valves.AccessLogValve"
directory="access_log"
prefix="192.168.1.20_access_log."> suffix=".txt"
pattern="common"/>
```

En este caso, se han usado varios atributos:

- *Directory*: nombre del directorio de almacenamiento de logs.
- *Prefix*: nombre del fichero de logs.
- *Suffix*: extensión del fichero de logs.
- *Pattern*: el patrón que se va a seguir en el fichero de logs. Tiene dos opciones common o combined. La opción de common tiene como información la siguiente:
  - %h: IP del host remoto.
  - %l: nombre usuario que realiza la petición.
  - %u: nombre usuario remoto que fue autenticado.
  - %t: hora y fecha de la petición.
  - %r: identificación de la sesión de usuario.
  - %s: código de respuesta de HTTP.
  - %b: bytes enviados.

#### Actividad propuesta 5.8



Configura un fichero de logs lo más completo posible y muestra el log creado a partir de la configuración.

Un ejemplo de la configuración anterior se puede observar en el fichero creado en el directorio \$CATALINA\_HOME/logs/192.168.1.20\_access\_log.txt:

		192.168.1.20_access_log.2020-08-14.txt	Guardar
		/opt/tomcat9/apache-tomcat-9.0.37/logs	
192.168.1.8	- -	[14/Aug/2020:10:17:03 -0400] "GET / HTTP/1.1" 200 11216	
192.168.1.8	- -	[14/Aug/2020:10:17:04 -0400] "GET /tomcat.png HTTP/1.1" 200 5103	
192.168.1.8	- -	[14/Aug/2020:10:17:04 -0400] "GET /tomcat.css HTTP/1.1" 200 5581	
192.168.1.8	- -	[14/Aug/2020:10:17:04 -0400] "GET /bg-upper.png HTTP/1.1" 200 3103	
192.168.1.8	- -	[14/Aug/2020:10:17:04 -0400] "GET /bg-nav.png HTTP/1.1" 200 1401	
192.168.1.8	- -	[14/Aug/2020:10:17:04 -0400] "GET /bg-button.png HTTP/1.1" 200 713	
192.168.1.8	- -	[14/Aug/2020:10:17:04 -0400] "GET /asf-logo-wide.svg HTTP/1.1" 200 27235	
192.168.1.8	- -	[14/Aug/2020:10:17:04 -0400] "GET /bg-middle.png HTTP/1.1" 200 1918	
192.168.1.8	- -	[14/Aug/2020:10:17:05 -0400] "GET /manager/html HTTP/1.1" 401 2499	
192.168.1.8	- -	[14/Aug/2020:10:17:11 -0400] "GET /manager/html HTTP/1.1" 401 2499	
192.168.1.8	- -	[14/Aug/2020:10:17:15 -0400] "GET /manager/images/asf-logo.svg HTTP/1.1" 200 20486	
192.168.1.8	- -	[14/Aug/2020:10:17:15 -0400] "GET /manager/images/tomcat.gif HTTP/1.1" 200 2066	
192.168.1.8	- tomcat	[14/Aug/2020:10:17:15 -0400] "GET /manager/html HTTP/1.1" 200 23884	
192.168.1.8	- -	[14/Aug/2020:10:17:16 -0400] "GET /favicon.ico HTTP/1.1" 200 21630	

**Figura 5.21**  
Fichero de logs

En esta imagen se puede observar que el acceso se ha realizado desde la IP 192.168.1.8, la hora que se ha realizado el acceso, el método HTTP que ha sido GET, los recursos que ha accedido, el código de respuesta de HTTP y los bytes enviados.

2. *Extended Access Log Valve*: es una implementación extendida de la anterior opción. La principal diferencia entre la anterior y esta es que usa dos atributos diferentes, como son org.apache.catalina.valves.ExtendedAccessLogValve y pattern que posee otros valores diferentes. Además usa tokens diferentes en la plantilla, cumpliendo la normativa de Extended Log File Format definida por W3C. Un ejemplo sería el siguiente:

```
<Valve className="org.apache.catalina.valves.ExtendedAccessLogValve" directory="extended_access_log" pattern="date time c-ip s-ip cs-method cs-uri-stem cs-uri-query sc-status bytes time-taken cs(User-Agent) cs(Cookie) cs(Referer) cs(HOST)" prefix="${tomcat.instance.name}-" resolveHosts="false" suffix=".log"/>
```

**Recurso web**

Los tokens soportados del log de acceso ampliado se pueden observar en la web de Apache-Tomcat.



Además, se puede configurar el control de acceso remoto, que posee tres opciones, que son las siguientes:

- ✓ *Remote Address Valve*: permite comprobar la IP del cliente con una o más expresiones regulares y como resultado puede permitir la solicitud o denegar la petición de solici-

tud del cliente. Un ejemplo que denegaría las solicitudes de las IP cuyo primer dígito comience por 192, sería el siguiente:

```
<Valve className="org.apache.catalina.valves.RemoteAddrValve"
deny="192.*">
```

Otro ejemplo sería dar acceso si la solicitud intentara la conexión por el puerto 8009, sería el siguiente:

```
<Valve className="org.apache.catalina.valves.RemoteAddrValve"
       addConnectorPort="true"
       invalidAuthenticationWhenDeny="true"
       allow=". *;8009"/>
```

- ✓ *Remote Host Valve*: es similar al anterior pero, en lugar de IP, se compara el nombre del host. Un ejemplo en el que se denegaría los servidores que comiencen por uw sería el siguiente:

```
<Valve className="org.apache.catalina.valves.RemoteHostValve"
deny="uw*"/>
```



#### PARA SABER MÁS

Para esta opción es necesario tener habilitada la resolución DNS mediante el parámetro:

```
<Connector enableLookups="true" />
```

- ✓ *Remote CIDR Valve*: permite comparar la IP del cliente con una o más subredes y, dependiendo de los valores suministrados, dará acceso a la solicitud o denegará la misma. Un ejemplo sería permitir a los clientes conectarse solamente desde la máquina local:

```
<Valve className="org.apache.catalina.valves.RemoteCIDRValve"
       allow="127.0.0.1, ::1"/>
```

Por otro lado, se tiene la posibilidad de que los usuarios inicien sesión solamente una vez en todas las aplicaciones de un mismo virtual host:

- ✓ *Single Sign On Valve*: esta opción es muy válida si se tienen varias aplicaciones en un mismo virtual host y solo se quiere una validación. Esta configuración habría que incluirla en el fichero \$CATALINA\_HOME/conf/server.xml. Un ejemplo sería el siguiente:

```
<Host appBase="webapps" autoDeploy="true" name="localhost" unpackWARs="true">
```

```
<Valve className="org.apache.catalina.authenticator.SingleSignOn"
/>
</Host>
```

Por último, se explicará el error de acceso a un método HTTP, que se configura mediante la opción:

- ✓ *Error Report Valve*: es una opción que permite administrar los códigos de error para HTTP, y se puede configurar mediante páginas estáticas HTML o mediante excepciones. Un ejemplo sería el siguiente:

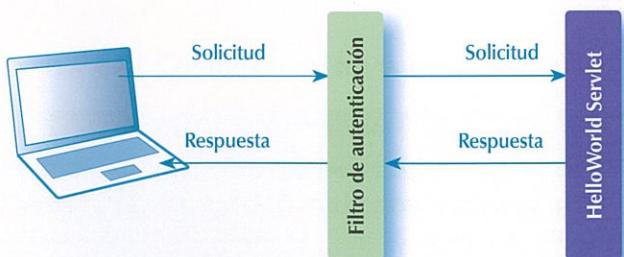
```
<Valve className="org.apache.catalina.valves.ErrorReportValve"
showReport="false"
showServerInfo="false" />
```

También se pueden definir los logs de depuración que permiten registrar eventos generados por el servidor y de las aplicaciones. Su principal objetivo es realizar una depuración exacta de dónde se ha producido un error, ya sea por fallo de configuración o por cualquier otra eventualidad del servidor. Un ejemplo sería el siguiente:

```
<Context path="/Bufete" docBase="Bufete" debug="0"
reloadable="true" crossContext="true">
<Logger className="org.apache.catalina.logger.FileLogger"
prefix="localhost_Bufete_log" suffix=".txt"
timestamp="true"/>
</Context>
```

Esta configuración se encontraría dentro del directorio \$CATALINA\_HOME/conf/server.xml y está ubicado dentro del tag host. Se aplicaría directamente a la aplicación Bufete. Como atributos tiene el nombre de la clase, tanto el prefijo del fichero como el sufijo, y también la fecha y hora que genera el ítem en el fichero de log.

Tomcat provee de una infraestructura de filtros que permite controlar el flujo de entrada y salida hacia una página web, normalmente del tipo JSP. Tales filtros pueden ser configurados para todas las aplicaciones usando \$CATALINA\_HOME/conf/web.xml o pueden ser configurados para cada una de las aplicaciones en el directorio de la aplicación WEB-INF/web.xml. La arquitectura puede ser algo parecida a la siguiente imagen:



**Figura 5.22**  
Arquitectura de filtros

Esta imagen muestra cómo funcionan los filtros y usan el flujo de información entre el servlet y el cliente mediante solicitud y respuesta. Considerando el flujo anterior, se pueden implementar una serie de filtros, como son los siguientes:

- a) Compresión de salidas y entradas: permite comprimir todos los ficheros en uno para que la descarga sea más rápida.
- b) Controles de seguridad: permite controlar los ataques del tipo CSRF, CORS, etc., en general filtrar caracteres no deseados y validar de forma segura a los usuarios.
- c) Registros de eventos o acciones: registrar todas las acciones que se produzcan en el servidor de aplicaciones mediante ficheros de logs.
- d) Modificación de entradas y salidas: limpiar los formularios, que es el principal foco de amenaza para las aplicaciones.
- e) Filtros de IP remotas: filtrar las direcciones IP no deseadas, de esta forma solo se permiten las solicitudes desde determinadas subredes o IP.

Se va a describir cómo es la implementación de los filtros. Primero hay que definir el nombre del filtro, posteriormente, el nombre de la clase que implementa el filtro con los parámetros que usa tal clase, y finaliza con la aplicación del filtro a un patrón de URL. Un ejemplo de cualquier aplicación de filtro sería de esta forma genérica:

```
<filter>
    <filter-name>NOMBRE_FILTRO</filter-name>
    <filter-class>CLASE_DEL FILTRO</filter-class>
    <init-param>
        <param-name>PARAM_1_FILTRO</param-name>
        <param-value>VALOR_1</param-value>
    </init-param>
    :
    <init-param>
        <param-name>PARAM_n_FILTRO</param-name>
        <param-value>VALOR_n</param-value>
    </init-param>
</filter>

<filter-mapping>
    <filter-name>NOMBRE_FILTRO</filter-name>
    <url-pattern>PATRON</url-pattern>
</filter-mapping>
```

Por ejemplo, si se quiere implementar un filtro como CORS sería de la siguiente forma:

```
<filter>
    <filter-name>Filtrado_seguridad</filter-name>
    <filter-class>org.apache.catalina.filters.CorsFilter</filter-class>
    <init-param>
        <param-name>cors.allowed.origins</param-name>
        <param-value>https://www.apache.org</param-value>
    </init-param>
```

```
<init-param>
    <param-name>cors.allowed.methods</param-name>
    <param-value>GET,POST,HEAD,OPTIONS,PUT</param-value>
</init-param>
<init-param>
    <param-name>cors.allowed.headers</param-name>
    <param-value>Content-Type,X-Requested-With,accept,Origin,Access-Control-Request-Method,Access-Control-Request-Headers</param-value>
</init-param>
<init-param>
    <param-name>cors.exposed.headers</param-name>
    <param-value>Access-Control-Allow-Origin,Access-Control-Allow-Credentials</param-value>
</init-param>
<init-param>
    <param-name>cors.support.credentials</param-name>
    <param-value>true</param-value>
</init-param>
<init-param>
    <param-name>cors.preflight.maxage</param-name>
    <param-value>10</param-value>
</init-param>
</filter>
<filter-mapping>
    <filter-name>Filtrado_seguridad </filter-name>
    <url-pattern>/*</url-pattern>
</filter-mapping>
```

Este filtro controla el origen cruzado de recursos, tales como la incrustación de imágenes, vídeos, etc., cuando se produce una solicitud HTTP a un dominio diferente del que está funcionando en ese momento. Y tiene los siguientes parámetros:

- *Cors.allowed.origins*: listado de URL que se le permite entrar al recurso, directorio, fichero .html, jsp, etc.
- *Cors.allowed.methods*: los métodos HTTP que son permitidos para acceder al recurso.
- *Cors.allowed.headers*: las cabeceras que son aceptadas para acceder al recurso.
- *Cors.exposed.headers*: las cabeceras de respuesta que son permitidas del tipo X-CUSTOM-HEADER.

### Recurso web

www

Accede al siguiente código QR para estudiar y comprobar todos los filtros que implementa Apache-Tomcat.



- *Cors.support.credentials*: indica el parámetro si el recurso soporta las credenciales del usuario.
- *Cors.preflight.maxage*: el tiempo en segundos que permite que el navegador tenga una solicitud en caché.

Y se aplica a todos los directorios debajo de la aplicación con el patrón /\*.

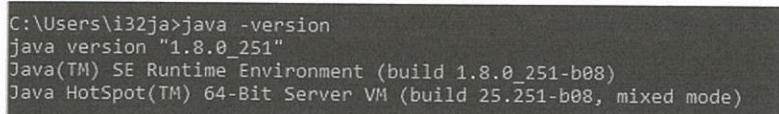
## 5.7. Instalación y configuración del servidor de aplicaciones en SO Windows

Después de haber instalado el servidor de aplicaciones en Linux, se va a proceder a la instalación de Apache-Tomcat en el sistema operativo Windows y configuración inicial para posteriormente desplegar aplicaciones. En la URL de Tomcat <https://tomcat.apache.org/download-90.cgi> se tienen dos opciones, la opción manual, esto es, descargar la opción Windows 64 bits o 32 bits, o la opción de un ejecutable por pantallas. La instalación que se va a realizar es manual para que se observe toda la configuración completa, y es la siguiente:

A lo largo del tiempo, Apache va sacando a la luz diferentes versiones de Tomcat; la versión última es la 10, pero está en fase de pruebas alpha. Por lo tanto, se va a instalar la versión más estable, que es Tomcat 9, con la versión de JDK 8, ya que es compatible. Primero se empieza con la versión de JDK. Para ello se observa si existe alguna versión instalada: se abre una ventana de MS-DOS y se ejecuta el siguiente comando:

```
C:/java -version
```

Y se visualizará una ventana como la de la figura 5.23.



```
C:\Users\i32ja>java -version
java version "1.8.0_251"
Java(TM) SE Runtime Environment (build 1.8.0_251-b08)
Java HotSpot(TM) 64-Bit Server VM (build 25.251-b08, mixed mode)
```

**Figura 5.23**  
Versión de Java

1. En nuestro caso, ya se tenía una versión, pero no es la última. La última es 1.8.0\_261, igual que se instaló en Linux. Se puede descargar de la página de Oracle <https://www.oracle.com/es/java/technologies/javase/javase-jdk8-downloads.html> o actualizar directamente desde el ícono que se visualiza en la parte inferior derecha de la barra de herramientas de Windows 10.
2. Posteriormente, se puede observar la versión última actualizada; cada año van saliendo versiones nuevas. Es importante comprobar la compatibilidad de Tomcat con JDK para no llevarse sorpresas. Ahora se descarga la versión de Tomcat 9 y se descomprime, por ejemplo, en C:\.
3. En el explorador, se debería ejecutar el archivo dentro de C:\apache-tomcat-9.0.34\bin\startup.bat y se iniciaría el servidor de aplicaciones Tomcat.
4. Se puede comprobar que Apache-Tomcat está funcionando correctamente abriendo el navegador y tecleando <http://localhost:8080> o [http://IP\\_equipo:8080](http://IP_equipo:8080), que mostrará una pantalla como la de la figura 5.6.

5. A continuación, hay que configurar algún usuario para poder administrar las aplicaciones mediante el acceso al botón *Manager App*. Para ello, es necesario modificar el fichero C:\apache-tomcat-9.0.34\conf\tomcat-users.xml. Y además hay que crear un usuario con los roles adecuados como manager-gui y admin-gui para poder administrar las aplicaciones. Por ejemplo, se habilita el usuario admin y password admin, y se observará la pantalla 5.24.



The screenshot shows the 'Gestor de Aplicaciones Web de Tomcat' interface. At the top, there is a message: 'OK - Arrancada aplicación en trayectoria de contexto [/host-manager]'. Below this, there are tabs for 'Listar Aplicaciones', 'Ayuda HTML de Gestor', and 'Ayuda do Gestor'. The main table is titled 'Aplicaciones' and has columns: Ruta, Versión, Nombre a Mostrar, Ejecutándose, Sesiones, and Comandos. It lists four applications:

Ruta	Versión	Nombre a Mostrar	Ejecutándose	Sesiones	Comandos
/	Ninguno especificado	Welcome to Tomcat	true	0	[Arrancar] [Parar] [Recargar] [Replegar] [Expirar sesiones sin trabajar > 30 minutos]
/docs	Ninguno especificado	Tomcat Documentation	true	0	[Arrancar] [Parar] [Recargar] [Replegar] [Expirar sesiones sin trabajar > 30 minutos]
/examples	Ninguno especificado	Servlet and JSP Examples	true	0	[Arrancar] [Parar] [Recargar] [Replegar] [Expirar sesiones sin trabajar > 30 minutos]
/host-manager	Ninguno especificado	Tomcat Host Manager Application	true	0	[Arrancar] [Parar] [Recargar] [Replegar] [Expirar sesiones sin trabajar > 30 minutos]

**Figura 5.24**  
Administrador de aplicaciones

## 5.8. Despliegue de aplicaciones en el servidor de aplicaciones

Una aplicación web es un software que permite cumplir con un objetivo y satisfacer unas necesidades de los clientes que se conectan a la misma. Otra definición más práctica sería un conjunto de ficheros que comprende desde servlets, páginas web estáticas y dinámicas, ficheros .jsp, ficheros .php, ficheros .css, clases de Java, ficheros de configuración .xml hasta ficheros necesarios para el buen funcionamiento de la aplicación.

Para desarrollarla se necesita de un entorno de desarrollo IDE, como Netbeans o Eclipse, donde se podrá probar la aplicación antes de pasarla a producción, aunque se podría desarrollar a mano directamente en el servidor, pero tiene más dificultad.

Una vez definidos estos conceptos, se va a detallar la estructura de ficheros de una aplicación web, aunque se pueden crear directorios adicionales. Tal estructura sería la siguiente:

- ✓ /: la carpeta raíz es la que contiene todos los directorios y además puede contener ficheros como .css, .html, .jsp, etc. Otra opción es crear un directorio o varios y estructurar bien la aplicación. Por ejemplo, un directorio para los ficheros .html y .css, otro para los ficheros .jsp, etc. Esto depende del programador.
- ✓ /META-INF: este directorio almacena el fichero context.xml, que consiste en el contexto de la aplicación.
- ✓ /WEB-INF: esta carpeta almacena el descriptor de despliegue (web.xml) en tal fichero y algunos directorios que se usarán en la aplicación.
- ✓ /WEB-INF/classes: este directorio almacenará las clases de Java que se usarán en la aplicación. Normalmente los servlets están en esta carpeta.
- ✓ /WEB-INF/lib: esta carpeta almacena las librerías .jar que se usarán por parte de las clases de Java.

- ✓ *web.xml*: es uno de los ficheros principales de la aplicación y de Tomcat, que permite definir los servlets con sus parámetros, definir las restricciones de seguridad con sus roles para acceso a los recursos, la definición de login mediante formulario o autenticación básica, y además los ficheros de inicio como index.jsp, index.html, etc.
- ✓ *context.xml*: este fichero contendrá las funciones o fuentes de datos que se quieren usar en la aplicación web.

Después de conocer la estructura de ficheros de la aplicación web, se van a explicar detalladamente los pasos para desplegar una aplicación web, ya sea manual o mediante el archivo .war.

Se ha desarrollado como ejemplo una aplicación web que calcula las operaciones aritméticas de matemáticas, como son la suma, resta, multiplicación y división. Básicamente, una calculadora con un servlet y un fichero index.jsp que llama al servlet. Se puede realizar esta aplicación en un IDE como Netbeans o Eclipse donde es necesario instalar Tomcat, y posteriormente crear el fichero .war o copiar el fichero completo de la aplicación de forma manual.

Para el despliegue manual de una aplicación web es necesario realizar lo siguiente:

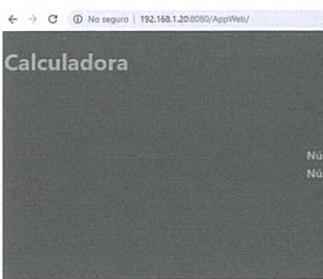
1. Crear una carpeta con el nombre del proyecto, en nuestro caso, AppWeb debajo de webapps de la carpeta raíz donde se ha instalado Tomcat.
2. Dentro de este directorio habrá dos directorios, META-INF y WEB-INF. Habría que crear y copiar la información que existe dentro del directorio WEB-INF de nuestra aplicación, así como copiar el fichero context.xml a META-INF.
3. En el directorio raíz se ubicará el fichero o ficheros iniciales de la aplicación, que en nuestro caso sería index.jsp y estilo.css.
4. Dentro de WEB-INF habrá dos directorios, classes y lib. Dentro de lib habría que copiar todos los ficheros .jar que se necesiten para la aplicación.
5. Es necesario crear el fichero web.xml, si no existiera, para incluir la información de servlet o cualquier otra configuración. En nuestro caso, se ha creado en Netbeans, por lo que se copia tal cual.
6. Por último, dentro del directorio classes existirá una carpeta creada llamada Logica y ahí estará la clase del Servlet Controlador.class ya compilada y lista para ejecutar.
7. Ahora ya se está en disposición de acceder a la aplicación mediante la URL <http://192.168.1.20:8080/AppWeb> o <http://localhost:8080/AppWeb> y se mostrará la página de la figura 5.25.
8. Si se teclean dos números en los dos cuadros de texto y se pulsa en sumar, se llamará al servlet llamado Controlador y mostrará el resultado, como se puede observar en la figura 5.26.



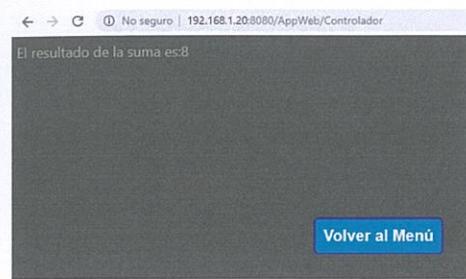
#### PARA SABER MÁS

También se puede desplegar la aplicación de forma directa copiando el archivo AppWeb.war generado en el directorio dist de la aplicación y posteriormente subiéndolo a la página de la administración de aplicaciones.

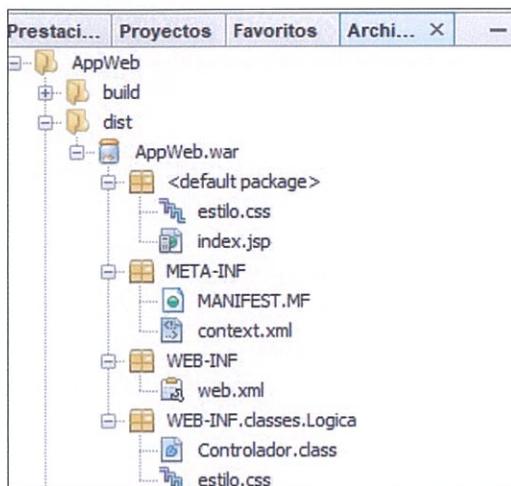
Ya está desplegada la aplicación. Se muestra la estructura de la aplicación en la figura 5.27.



**Figura 5.25**  
Aplicación AppWeb



**Figura 5.26**  
Servlet AppWeb



**Figura 5.27**  
Estructura AppWeb

A continuación se van a mostrar los ficheros más importantes de la aplicación, comenzando por el fichero web.xml:

```
<web-app version="3.1" xmlns="http://xmlns.jcp.org/xml/ns/javaee" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd">
    <servlet>
        <servlet-name>Controlador</servlet-name>
        <servlet-class>Logica.Controlador</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>Controlador</servlet-name>
        <url-pattern>/Controlador</url-pattern>
    </servlet-mapping>

    <welcome-file-list>
        <welcome-file>
            index.jsp
        </welcome-file>
    </welcome-file-list>
```

```
</welcome-file-list>
</web-app>
```

El siguiente fichero es el de contexto de la aplicación llamado context.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<Context path="/AppWeb"/>
```

Posteriormente, el fichero de inicio, llamado index.jsp:

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>

<link rel="stylesheet" type="text/css" href="estilo.css" />
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Calculadora Web</title>
    </head>
    <body>
        <h1>Calculadora</h1>
        <div class="centrar">
            <form action="Controlador" method="POST">
                <table>

                    <tr>
                        <td width="150px"><b>Número 1:</b></td>
                        <td width="10px"><input type="text" name="num1" size="20" Value="Número 1" onclick="this.value=''" />
                    </td>
                </tr>
                    <tr>
                        <td width="150px"><b>Número 2:</b></td>
                        <td width="10px"><input type="text" name="num2" size="20" Value="Número 2" onclick="this.value=''" />
                    </td>
                </tr>

                    <tr>
                        <td height="50px" width="50px">
                            </td>
                    </tr>
                </table>
            <div class="centrarboton">
                <input type="submit" name="btnsuma" value="SUMAR" class="boton">
                <input type="submit" name="btnresta" value="RESTAR" class="boton">
            </div>
        </div>
    </body>
</html>
```

```
<input type="submit" name="btncalcular" value="CALCULAR" class="boton">
<input type="submit" name="btnclear" value="Borrar" class="boton">
</div>
</form>
</body>
</html>
```

Por último, el controlador, que es el servlet de la aplicación:

```
public class Controlador extends HttpServlet {

    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        try (PrintWriter out = response.getWriter()) {

            /* TODO output your page here. You may use following sample code. */
            out.println("<!DOCTYPE html>");
            out.println("<link rel=\"stylesheet\" type=\"text/css\" href=\"estilo.css\" />");
            out.println("<html>");
            out.println("<head>");
            out.println("<title>Calculo</title>");
            out.println("</head>");
            out.println("<body>");

            String num1=request.getParameter("num1");

            String num2=request.getParameter("num2");

            String btnSuma=request.getParameter("btncalcular");
            String btnResta=request.getParameter("btnclear");
            String btnMulti=request.getParameter("btnclear");
            String btnDividir=request.getParameter("btnclear");

            if(btnSuma!=null){
                int resul=Integer.parseInt(num1) + Integer.parseInt(num2);
                out.println("El resultado de la suma es:" + resul);

            }
            if(btnResta!=null){
                int resul=Integer.parseInt(num1) - Integer.parseInt(num2);
                out.println("El resultado de la resta es:" + resul);

            }
        }
    }

}
```

```
if(btnMulti!=null){
    int resul=Integer.parseInt(num1) * Integer.parseInt(num2);
    out.println("El resultado de la multiplicación es:" + resul);

}
if(btnDividir!=null){
    int resul=Integer.parseInt(num1) / Integer.parseInt(num2);
    out.println("El resultado de la división es:" + resul);

}

out.println("<div class=\"centrarbotonvolver>");
out.println("<input type=\"button\" name=\"volver\" value=\"Volver al Menú\" class=\"boton\" onclick=\"location.href='index.jsp'\">");
out.println("</div>");
out.println("</body>");
out.println("</html>");

}

}
```

## **5.9. Seguridad en el servidor de aplicaciones.**

### **Configurar el servidor de aplicaciones con soporte SSL/T**

Una vez que se ha desplegado una aplicación web, no se puede olvidar proteger a la aplicación web y al servidor de aplicaciones de accesos malintencionados. Por otro lado, controlar que personas ajenas a la aplicación no intercepten información susceptible a través de Internet.

### 5.9.1. Seguridad y autenticación

Para llevar a cabo las soluciones ante las anteriores amenazas, un sistema de seguridad se basa en tres conceptos clave, que son los siguientes:

- a) *Autenticación*: proceso para identificar quién entra a la aplicación es quién dice ser, y que pueda acceder a los recursos.
  - b) *Confidencialidad*: solamente los extremos de la comunicación conocen la información que se intercambia.
  - c) *Integridad*: la información que se transmite de extremo a extremo no es modificada por agentes externos.

Con relación a la seguridad, lo importante es que el servidor de aplicaciones controle las comunicaciones entre los distintos elementos que intercambian información a lo largo del flujo de la aplicación. Esto se realiza de forma transparente al usuario. El fichero web.xml es el que se encarga de esta función con las marcas que se han explicado anteriormente.

Por otro lado, se puede controlar la seguridad mediante la programación de servlet y ficheros .jsp que permitan la seguridad de la aplicación.

Con relación a la autenticación, se tienen distintos tipos que se pueden implementar en una aplicación web. Son los siguientes:

- Autenticación digest: es una variante de basic. En lugar de viajar la password por la red, viaja encriptada mediante una función hash. Todos los servidores no soportan este tipo de autenticación.
- Autenticación basic: se basa en solicitar datos al usuario, como un nombre y una contraseña. Esta información no va codificada, por lo que es peligroso usar este tipo de autenticación.
- Autenticación basada en formularios: se basa en solicitar datos al usuario mediante un formulario que introduzca el usuario y una password. Este mecanismo es débil de cara a los hackers, ya que se puede obtener esta información de forma fácil.
- Certificados digitales y SSL: lo ideal es usar el protocolo HTTPS que funciona con el puerto 443 que permite la confidencialidad y la integridad de la información y, por otro lado, se tiene asegurada la autenticación. Todo ello se basa en la criptografía de clave pública. Este método es el que se implementará a continuación.

### 5.9.2. Configuración SSL sobre Tomcat

Con relación a la configuración SSL, se van a explicar detalladamente los pasos que se van a realizar:

1. Para usar transacciones SSL seguras es necesario crear un almacén de claves. Para ello es necesario crear un fichero con el programa de generación de claves de Java llamado keytool, que se incluye normalmente en el JDK instalado en el sistema operativo. Para crear un almacén de claves nuevo es necesario ejecutar el siguiente comando:

```
#keytool -genkey -alias tomcat -keyalg RSA -keystore tomcat.jks  
-validity 365 -keysize 2048
```

La opción alias permite colocar un nombre para identificar al fichero, el tipo de algoritmo es RSA, el almacén de claves es tomcat.jks, la validez es 365 días y, por último, la clave es de 2048 bits. Al ejecutar este comando, te solicitará una serie de datos, primero de todo una clave, que será necesaria para usarla posteriormente. Una vez introducidos los datos, se creará el fichero tomcat.jks, como se puede observar en la pantalla de la figura 5.28.

2. Posteriormente, sería necesario configurar el fichero \$CATALINA\_HOME/conf/server.xml de Tomcat para habilitar SSL (figura 5.29).

```
root@osboxes:/opt/tomcat9/certificados# keytool -genkey -alias tomcat -keyalg RSA -keystore tomcat.jks -validity 365 -keysize 2048
Introduzca la contraseña del almacén de claves: default
Volver a escribir la contraseña nueva: default
¿Cuáles son su nombre y su apellido? [Unknown]: localhost
¿Cuál es el nombre de su unidad de organización? [Unknown]: APP
Introduzca la contraseña de clave para <tomcat> [Unknown]: APP
¿Cuál es el nombre de su ciudad o localidad? [Unknown]: Córdoba
¿Cuál es el nombre de su estado o provincia? [Unknown]: Córdoba
¿Cuál es el código de país de dos letras de la unidad? [Unknown]: ES
¿Es correcto CN=localhost, OU=APP, L=Córdoba, ST=Córdoba, C=ES? [no]: si
[Unknown]: TLSv1.2
Either JSSE or OpenSSL style configuration may be used. OpenSSL style configuration is used below.
Introduzca la contraseña de clave para <tomcat>
(INTRO si es la misma contraseña que la del almacén de claves):
```

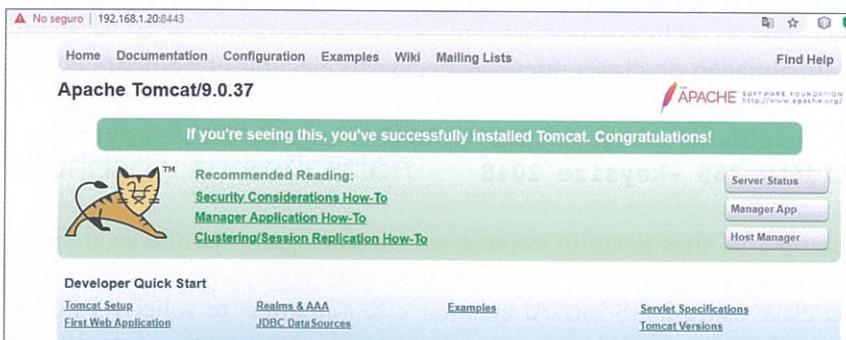
**Figura 5.28**  
Almacén de claves

```
<!-- Define an SSL/TLS HTTP/1.1 Connector on port 8443
This connector uses the NIO implementation. The default
SSLImplementation will depend on the presence of the APR/native
library and the useOpenSSL attribute of the
AprLifecycleListener.
Either JSSE or OpenSSL style configuration may be used regardless of
the SSLImplementation selected. JSSE style configuration is used below.
-->

<Connector port="8443" protocol="org.apache.coyote.http11.Http11NioProtocol"
maxThreads="150" scheme="https" secure="true" SSLEnabled="true"
KeystoreFile="conf/tomcat.jks" keystorePass="cat.12345" clientAuth="false"
sslProtocol="TLSv1.2" SSLVerifyClient="none">
</Connector>
<!-- Define an SSL/TLS HTTP/1.1 Connector on port 8443 with HTTP/2
This connector uses the APR/native implementation which always uses
OpenSSL for TLS.
```

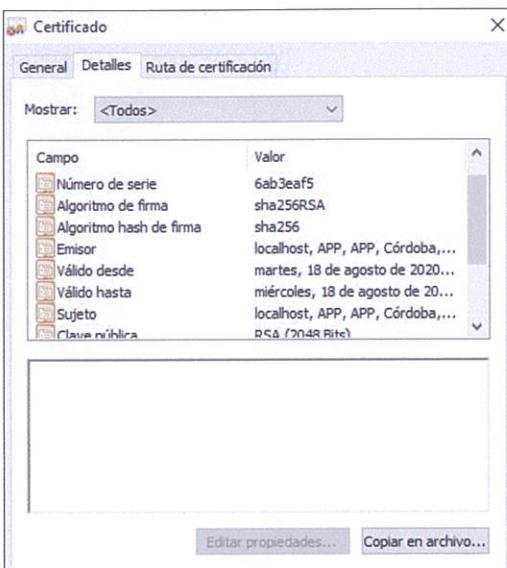
**Figura 5.29**  
Fichero server.xml

- Después, sería necesario teclear la URL `https://localhost:8443` o `https://IP_host:8443` (figura 5.30).



**Figura 5.30**  
Página inicial Tomcat

Si se observa la pantalla anterior, nos advierte que el sitio no es seguro, pero esto no quiere decir que no haya certificado, sino que no existe una entidad certificadora que permita validar el certificado que se ha creado (figura 5.31).



**Figura 5.31**  
Certificado

Para solucionar el problema anterior, se puede pagar a una autoridad certificadora y descargar el certificado correspondiente para agregarlo al navegador y que permita confiar en el emisor del certificado.

### Recurso web

La lista de entidades certificadoras de confianza se pueden localizar en la página del Ministerio de Asuntos Económicos y la Transformación Digital.



### Resumen

- En este capítulo se ha explicado detalladamente todo el entorno que engloba el despliegue de una aplicación.
- La arquitectura de modelo-vista-controlador es básica en cualquier aplicación del mercado, ya que permite separar al programador las partes bien diferenciadas de las que consta una aplicación. El modelo hace de puente entre el controlador y la vista, se encarga de realizar las peticiones y modificaciones en base a los privilegios. La vista es la parte que observa el usuario y que posee una capa de abstracción que permite que se oculte toda la lógica que existe detrás. Por último, se tiene el controlador, que es el cerebro de la aplicación y permite que todo funcione correctamente.

- En el mercado actual existen bastantes servidores de aplicaciones, en este caso, se ha elegido Apache-Tomcat por ser uno de los más usados por las empresas y consultorías de desarrollo. La instalación de Tomcat se basa en la instalación de JDK para compilar las clases de Java, en la instalación propiamente dicha del software, y la configuración de las variables de entorno.
- A partir de aquí es necesario configurar usuarios, contraseñas y roles en el fichero de configuración tomcat-users.xml para poder administrar las aplicaciones en el servidor de aplicaciones.
- En la administración de aplicaciones es fundamental conocer cómo se despliega una aplicación a partir de un fichero .war o de forma manual. Y, sobre todo, la estructura fija de directorios que debe poseer una aplicación para poder desplegarse en Tomcat.
- La autenticación de usuarios en Tomcat se realiza a partir de los dominios de seguridad (Realm) que engloba el conjunto de usuarios, contraseñas y roles que se han de definir en los ficheros .xml correspondientes.
- Por otro lado, se tienen la administración y configuración de las sesiones de las aplicaciones, que es necesario tener controladas para no permitir un fallo en el servidor. Y conocer cómo se configura una sesión persistente. A ello nos puede ayudar la configuración de los registros de acceso y los filtros de solicitudes.
- La instalación y configuración en sistemas Windows es parecida a la de Linux, las diferencias estriban en la estructura de directorios y algunos ficheros como los de ejecución. Pero es fundamental conocer cómo se instala el servidor de aplicaciones en varios sistemas operativos.
- Una vez que se programa una aplicación web, ya sea manual o mediante la ayuda de un IDE de programación, es clave saber cómo se despliega y que el programador conozca su estructura y ficheros clave en el despliegue. Ellos son web.xml, context.xml, el fichero de inicio index.jsp, index.html, o cualquier otro que sea compatible con el servidor de aplicaciones.
- Por último, securizar el servidor de aplicaciones mediante SSL es fundamental para impedir el acceso malintencionado de personal ajeno a la aplicación. Es necesario modificar el fichero server.xml del servidor.

### Supuestos prácticos

1. Aprovechando la instalación de Tomcat, se van a configurar distintos parámetros para comprobar su funcionamiento:
  - Cambia el puerto de conexión al 8090, por defecto 8080.
  - Crea un usuario llamado admin y password sol.1234 para poder administrar las aplicaciones.
2. Comprueba la versión de JDK con los comandos oportunos y, posteriormente, actualízala con la última versión descargada en la URL de Oracle.

3. Configura el fichero correspondiente para que Apache-Tomcat se pueda administrar desde otro dispositivo que no sea el local.
4. Genera un error en el servidor Apache-Tomcat mediante la configuración de un fichero y asegúrate que detectáis el error comprobando el fichero de log.
5. Configura una sesión persistente que posea los siguientes parámetros:
  - El directorio donde se almacenarán las sesiones será /opt/sesiones.
  - El intervalo será de 180 segundos.
  - Investiga un poco los demás parámetros y configúralos de forma óptima.
  - Comprueba que las sesiones se recuperan una vez el servidor está caído.
6. Crea una comunicación SSL mediante el puerto correspondiente, y la siguiente URL debería funcionar <https://localhost:8443>. Intenta configurarla por otro puerto.

## Ejercicios propuestos



1. Una vez instalado Apache-Tomcat, desarrolla una pequeña aplicación web que permita introducir tu nombre y tu año de nacimiento por un formulario, cuyo resultado debería ser la edad que tienes tomando como base el año actual:
  - Crea la aplicación en Netbeans.
  - Crea un fichero .jsp de entrada:
  - Crea un servlet que calcule la edad, es un fichero .java.
  - Fichero web.xml, context.xml.
  - Por último, despliega la aplicación en el servidor Apache-Tomcat.
2. A partir de la aplicación anterior, crea lo siguiente:
  - Un log específico para ella.
  - Una conexión persistente y demuestra que funciona.
3. Partiendo de la aplicación anterior, se desea que la aplicación sea de inicio en lugar de la página de inicio de Tomcat. Esto es, si se teclea la URL <http://localhost:8080> debería visualizarse nuestra aplicación. Se recomienda realizar una copia de seguridad antes de realizar este ejercicio, por si es necesario volver hacia atrás.
4. Configura el servidor para que acepte conexiones seguras mediante SSL, realizando el almacén de claves que se llame servertomcat.jks y colócalo en /opt/certificados.
5. Toma como ejemplo cualquier aplicación y permite solo las peticiones que procedan de la red 192.168.\*.\* mediante el uso de Remote Address Valve.

## ACTIVIDADES DE AUTOEVALUACIÓN

1. ¿Qué significan las siglas MVC en la arquitectura de Tomcat?:

- a) Modelo-Vista-Controlador.
- b) Modelado-Vista-Conexión.
- c) Modelado-Válvula-Controlador.
- d) Ninguna de las respuestas anteriores es correcta.

2. ¿Cuál de los siguientes son servidores de aplicaciones?:

- a) Apache.
- b) Tomcat.
- c) GlassFish.
- d) Las respuestas b) y c) son correctas.

3. ¿Qué fichero configura los usuarios que permiten la administración en Apache-Tomcat?:

- a) tomcat.xml
- b) users-xml
- c) tomcat-users.xml
- d) users-tomcat.xml

4. De los siguientes comandos, ¿cuál es el que permite crear el fichero AppWeb.war?:

- a) jar -cxvf AppWeb.war \*
- b) jav -tvf AppWeb.war \*
- c) jar -cvf AppWeb.war \*
- d) jas -cvf AppWeb.war \*

5. ¿Qué fichero nos permite configurar el puerto de Tomcat para administrar las aplicaciones?:

- a) context.xml
- b) server.xml.
- c) web.xml.
- d) Ninguna de las respuestas anteriores es correcta.

6. ¿Qué comando nos permite generar un almacén de claves para Tomcat y en qué fichero se puede configurar el protocolo SSL?:

- a) keytool y web.xml.
- b) openssl y web.xml.
- c) Toolkey y server.xml.
- d) keytool y server.xml.

7. ¿Qué puerto se configura por defecto en Tomcat para aceptar conexiones SSL?:

- a) 8080.
- b) 8443.
- c) 8444.
- d) 8445.

8. ¿En qué fichero de la aplicación se definen los servlets que se van a usar?:

- a) server.xml.
- b) context.xml.
- c) web.xml.
- d) tomcat.xml.

9. ¿Cuál de las siguientes clases permite implementar Access Log Valve (log de acceso)?:

- a) org.apache.catalina.valves.AccessLogValve.
- b) org.apache.catalina.AccessLogValve.
- c) org.catalina.valves.AccessLogValve.
- d) catalina.valves.AccessLogValve.

10. ¿En qué conceptos clave se basa un sistema de seguridad?:

- a) Integridad, modificación y autenticación.
- b) Coherencia, modificación y autenticación.
- c) Integridad, confidencialidad y autenticación.
- d) Integridad, phising y autenticación.

#### SOLUCIONES:

1. **a**  **b**  **c**  **d**

2.  **a** **b**  **c**  **d**

3.  **a**  **b** **c**  **d**

4.  **a**  **b** **c**  **d**

5.  **a** **b**  **c**  **d**

6.  **a**  **b**  **c** **d**

7.  **a**  **b**  **c**  **d**

8.  **a**  **b** **c**  **d**

9.  **a**  **b**  **c**  **d**

10.  **a**  **b** **c**  **d**

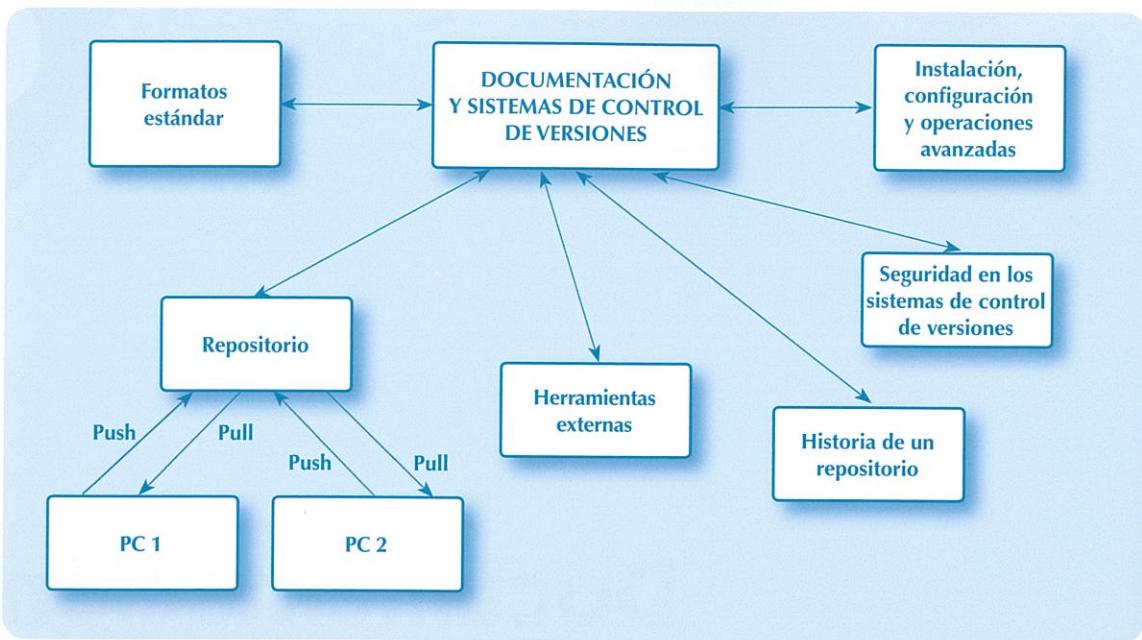


# Documentación y sistemas de control de versiones

## Objetivos

- ✓ Identificar las diferentes herramientas de generación de documentación.
- ✓ Documentar los componentes software utilizando los generadores específicos de las plataformas.
- ✓ Usar diferentes formatos para la documentación.
- ✓ Utilizar herramientas colaborativas para la elaboración y mantenimiento de la documentación.
- ✓ Instalar, configurar y usar un sistema de control de versiones.
- ✓ Garantizar la accesibilidad y seguridad de la documentación almacenada por el sistema de control de versiones.

## Mapa conceptual



## Glosario

**Array.** Se le llama también vector. Es una colección de elementos del mismo tipo, y existen varias clases como unidimensionales, bidimensionales, etc.

**Bug.** Es un error de software que provoca un resultado no deseado. También puede ser un problema de seguridad que permite un ataque al software por parte de personal externo.

**Historia de un repositorio.** Es un listado de cualquier repositorio que permite observar los diferentes commits, push, creación de ramas del ciclo de vida del desarrollo del proyecto. Cada cambio en las diferentes ramas permite generar una versión del software. Todos los sistemas de control de versiones deben contenerlo.

**IDE (Integrated Development Environment).** Es un programa informático que permite al programador desarrollar todo el software relacionado con un proyecto o aplicación web.

**Plantilla.** Es un texto o código que puede ser usado varias veces y permite ahorrar tiempo a la hora de implementar cualquier elemento en programación.

**Tags.** Son unas etiquetas que se aplican en el entorno de programación, por ejemplo, cuando se documenta un proyecto software.

**Template.** También se le llama *plantilla*. Es un documento base, a partir del cual se puede programar o generar documentación.

**Versión.** Indica numéricamente cuál es el avance en el desarrollo de una aplicación software.

## 6.1. Introducción

En este último capítulo se va a tratar la documentación de una aplicación web, tan fundamental como el código de la misma. Por otro lado, también se tratará el concepto de control de versiones de una aplicación cuando se está desarrollando por parte de un equipo de programadores. Su objetivo final es poner en producción la aplicación web para que se use con un objetivo específico. Y es esencial que el equipo de trabajo suba las distintas versiones probadas de la aplicación; lo ideal sería modular, y que cada programador se encargue de un módulo.

## 6.2. Herramientas externas para la generación de documentación. Instalación, configuración y uso

Antes de comenzar a explicar o entrar en detalle en herramientas para generar documentación, sería necesario tener en cuenta qué componentes o módulos serían necesarios documentar. Normalmente es un equipo de trabajo el que lleva a cabo la programación de una aplicación web, dependiendo de la dimensión del proyecto. Por ello, es preciso documentar toda aquella parte del código que será reutilizable o que pueda ser modificada por una versión posterior o por algún error de flujo de datos.

En el mercado actual existen bastantes herramientas para la generación de documentación, pero sería imposible tratar todas ellas en este libro; se va a realizar un resumen de algunas de ellas y se instalarán algunas para observar cómo se comportan. La lista de las herramientas que generan automáticamente documentación son las siguientes:

- *Javadoc*: es la herramienta de Oracle por excelencia para generar documentación en formato HTML a partir del código desarrollado en Java. Hay que explicarla, ya que se han puesto bastantes ejemplos a lo largo del libro. La mayoría de los IDE de Java generan automáticamente la documentación, y uno de ellos es Netbeans 8.2, pero todas las versiones tienen esta posibilidad.
- *phpDocumentor*: es otra herramienta escrita en PHP para generar documentación automáticamente vía código fuente PHP y como salida tiene el estándar PHPDoc.
- *Doxygen*: esta herramienta es más versátil ya que permite generar documentación en bastantes lenguajes de programación, como por ejemplo C++, Java, C, Fortran, VDHL, CLI, Python, IDL y también con ciertos matices PHP y D. Funciona en la mayoría de los sistemas Unix, Windows y MAC OS X.



### Actividad propuesta 6.1

Investiga en Internet qué herramientas existen en el mercado, además de las nombradas anteriormente, e intenta instalar y configurar alguna de ellas.

### 6.2.1. Instalación, configuración y uso de Jayadoc

La instalación de Javadoc no es necesaria en Windows, puesto que viene incorporado el componente en el IDE Netbeans, por lo que se puede configurar y usar directamente sobre el código Java desarrollado en cualquier proyecto.

A partir de un proyecto generado de cualquier aplicación se pueden documentar las clases, los métodos, las funciones, etc. Para escenificar este uso, se pondrá el siguiente ejemplo de un proyecto de Java relacionado con XML:

## Figura 6.1 Proyecto Java

Si se observa la imagen, es un código de una clase de Java con sus funciones. Lo normal en una clase es documentar el nombre, una descripción general, la versión y el nombre de autor o autores. Si se documenta la clase, se podrán documentar los siguientes parámetros:

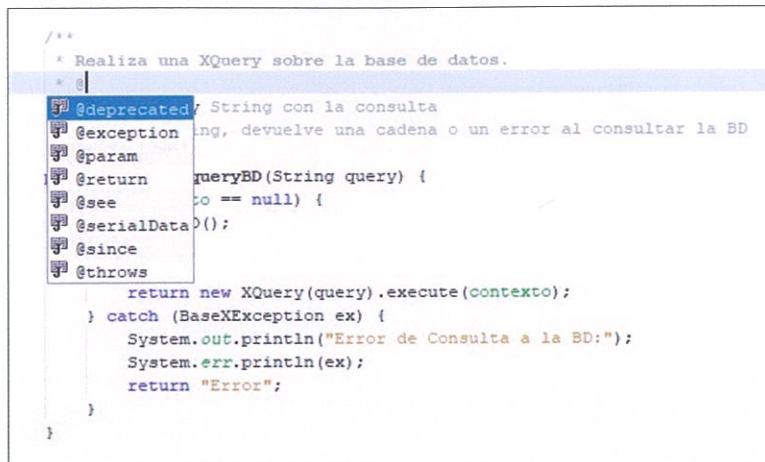
- ✓ **@author:** este atributo permite declarar el nombre del autor de la clase.
  - ✓ **@deprecated:** para declarar si algún elemento (por ejemplo, la clase) está obsoleto y no se recomienda su uso. Puede ser por varias razones, una de ellas es porque existe un problema de seguridad o bug, otra sería porque la forma de programar esa parte son malas prácticas, y la última porque existe una versión más actual.
  - ✓ **@param:** definición de un parámetro de un método.
  - ✓ **@see:** se asocia con otro método o clase.
  - ✓ **@serial:** describe el motivo del campo y sus posibles valores.
  - ✓ **@since:** la versión del producto que se está desarrollando.
  - ✓ **@version:** es la versión numérica de una clase o un método. Se puede comenzar por 0 o por 1, y, posteriormente, ir incrementando a medida que se implementen más versiones, por ejemplo 1.1. La secuencia la determina el programador o el equipo de programación.



#### TOMA NOTA

Es importante introducir la @ en la documentación con el parámetro correspondiente, ya que, en caso contrario, Javadoc dará un error por no poder generar la documentación adecuada.

Para documentar un método se visualizan algunos parámetros similares y otros distintos, como se observa en la figura 6.2.



```


    /**
     * Realiza una XQuery sobre la base de datos.
     */
    @deprecated String con la consulta
    @exception Ing, devuelve una cadena o un error al consultar la BD
    @param
    @return queryBD(String query) {
        @see
        @serialData()
        @since
        @throws
            return new XQuery(query).execute(contexto);
        } catch (BaseXException ex) {
            System.out.println("Error de Consulta a la BD:");
            System.err.println(ex);
            return "Error";
        }
    }
}


```

**Figura 6.2**  
Documentar método

Los parámetros del método son los siguientes:

- ✓ **@deprecated:** para declarar si algún elemento del método está obsoleto y no se recomienda su uso. Puede ser por varias razones, una de ellas es porque existe un problema de seguridad o bug, otra sería porque la forma de programar esa parte son malas praxis, y la última porque existe una versión más actual.
- ✓ **@exception:** es sinónimo de **@throws**, se inventó primero. Con el paso del tiempo los expertos concluyeron que era más exacto **@throws**.
- ✓ **@param:** describe el parámetro o parámetros que recibe el método.
- ✓ **@return:** describe el valor de salida del método; si el método es void no devuelve nada.
- ✓ **@see:** se asocia con otro método o clase.
- ✓ **@serialData:** describe el formato de serialización usado en el método.
- ✓ **@since:** la versión del producto que se está desarrollando.
- ✓ **@throws:** describe una excepción lanzada por el método que debe ser tenida en cuenta.

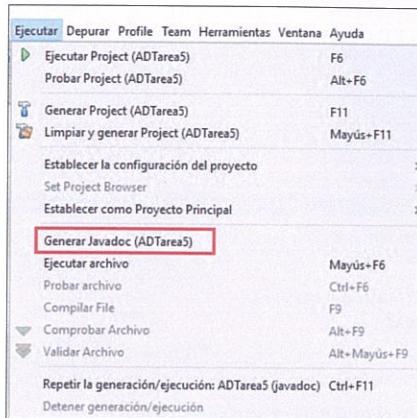
Una vez documentada la clase y los métodos de la misma se está en disposición de generar la documentación. Para ello, se ejecuta Javadoc dentro de la opción *Ejecutar* y pulsando en *Generar Javadoc* en el IDE Netbeans, como se observa en la figura 6.3.

Si se pulsa la opción anterior se visualizará una página como la siguiente, donde aparecerá toda la documentación generada por Javadoc de tu aplicación (figura 6.4)

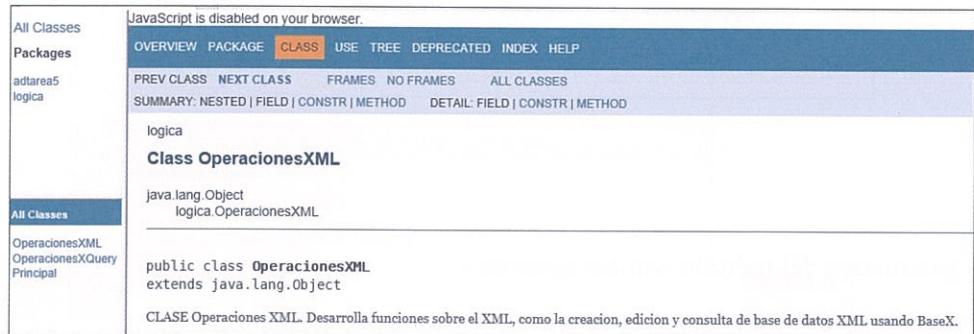


### Actividad propuesta 6.2

Toma una aplicación como ejemplo y genera documentación con Javadoc, previamente se deben documentar las clases y sus métodos.



**Figura 6.3**  
Javadoc



**Figura 6.4**  
Salida Javadoc

En esta salida se pueden observar las clases realizadas, como son Principal, OperacionesXML y OperacionesXQuery, los métodos con sus parámetros, etc. En conclusión, una perspectiva bastante acertada de la aplicación para observar cuál es su objetivo, cómo está implementada, la relación entre las clases. Por ello es tan importante realizar una documentación de una aplicación desarrollada.

### 6.2.2. Instalación, configuración y uso de phpDocumentor

Como se ha comentado anteriormente, existen varias herramientas para generar documentación de forma automática y una de ellas es phpDocumentor. Esta herramienta genera documentación para el lenguaje de programación PHP.

Los elementos que pueden ser documentados son los siguientes:

- Variables globales.
- Clases.
- Funciones.
- Métodos y atributos.
- Sentencias.

Por otro lado, se puede documentar también un bloque de código y puede hacer referencia a un archivo en particular, ya que normalmente en PHP la aplicación puede estar compuesta por varios archivos. La etiqueta que permite esta documentación es aquella que se denomina `@package`. Existen tres tipos de documentaciones relacionadas con el Modelo-Vista-Controlador:

- a) *Vista o Interfaz*: qué permite realizar, cómo lo hace, qué retorna, etc.
- b) *Modelo*: qué algoritmos usa, qué estructura tiene, qué flujo usa, etc.
- c) *Controlador*: qué métodos usa para gestionar el flujo entre la vista y el modelo, cómo optimizar tales métodos, etc.

**RECUERDA**

- ✓ La documentación de cualquier aplicación es básica, la cual reside dentro del código para que el desarrollador, con un simple vistazo, conozca de primera mano cada función, clase, etc. Luego un documento externo servirá para que cualquier consultor o jefe de proyecto pueda observar claramente la estructura y el funcionamiento de la aplicación a grandes rasgos.

Al igual que en Javadoc, en phpDocumentor existen etiquetas para documentar los bloques de documentación, todas precedidas por la `@`. Son las siguientes:

- ✓ `@author`: autor que implementa el código.
- ✓ `@copyright`: derechos de autor.
- ✓ `@access`: esta marca puede ser privada o pública. Si es privada no genera documentación y si es pública sí. El valor por defecto es pública.
- ✓ `@deprecated`: indica que un elemento está obsoleto y en futuras versiones del código no debería usarse.
- ✓ `@internal`: permite indicar la documentación para los programadores, pero no es pública.
- ✓ `@version`: la versión actual del código.

Y, para las funciones, serían las siguientes:

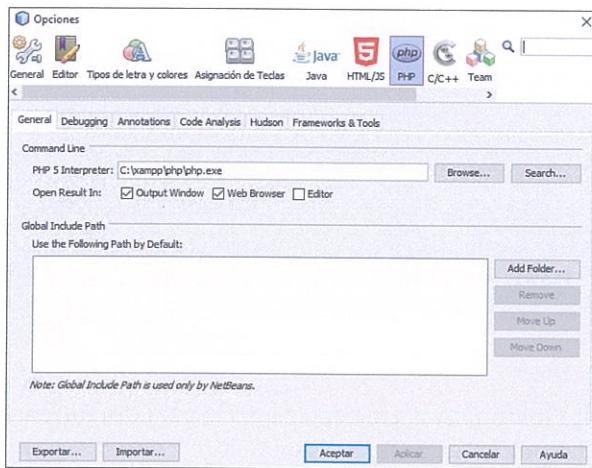
- ✓ `@global`: indica el uso global de una variable.
- ✓ `@return`: valor devuelto por la función.
- ✓ `@param`: parámetro o parámetros que recibe una función.
- ✓ `@var`: se usa para documentar los atributos de la clase.

**PARA SABER MÁS**

Si quieres profundizar más en phpDocumentor, puedes consultar su página web: <https://www.phpdoc.org/>

Si se toma una aplicación con una clase o una jerarquía de clases en PHP con sus métodos correspondientes, se puede documentar mediante los tags anteriores y una configuración que es necesario realizar en Netbeans 8.2. La configuración es la siguiente:

1. Se crea un proyecto en PHP mediante el interfaz de Netbeans y se crean los respectivos ficheros que va a contener la aplicación. Es necesario tener instalado XAMPP y Netbeans en el equipo.
2. En la interfaz de Netbeans 8.2 se selecciona la opción *Herramientas* y, dentro de esta, la opción *Opciones*. Se visualizará una pantalla como la 6.5.



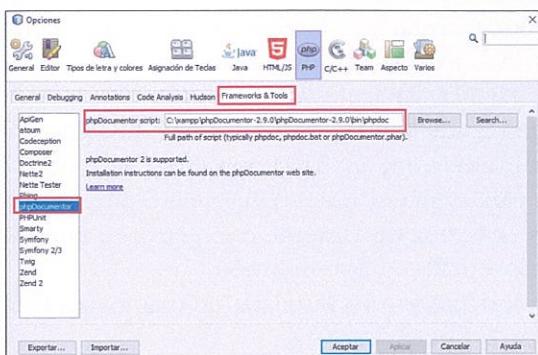
**Figura 6.5**  
Configuración PHP



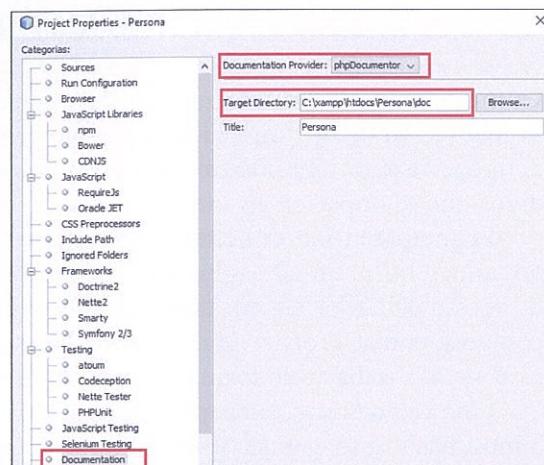
#### TOMA NOTA

Puede ser necesario activar la opción PHP en el menú de la figura 6.5, y se visualizarán todas las opciones.

3. Como se observa en la imagen anterior, es necesario colocar dónde está la ruta del intérprete de PHP, que en este caso coincide con la instalación de XAMPP.  
En un paso posterior, es necesario descargar la versión de phpDocumentor 2.9.0 que es estable y funciona perfectamente. Las siguientes versiones tienen algunos bugs que está solucionando la comunidad de programadores. Para descargar la versión se encuentra disponible en la URL <https://www.codingfix.com/es/installing-phpdocumentor-issues-solved/>.
4. Una vez descargada, se descomprime en el directorio de XAMPP y se configura en Netbeans en la misma ventana anterior, pero esta vez en la opción Framework & Tools (figura 6.6).
5. El siguiente paso sería configurar las propiedades del proyecto. Para ello es necesario seleccionar el proyecto con el botón derecho del ratón y después seleccionar *Propiedades*. Se visualizará una pantalla: en la parte izquierda es necesario seleccionar *Documentación* y en la parte derecha *Documentation Provider phpDocumentor*, y configurar en directorio de salida de la documentación. Normalmente se crea un directorio llamado doc que cuelga del directorio raíz (figura 6.7).

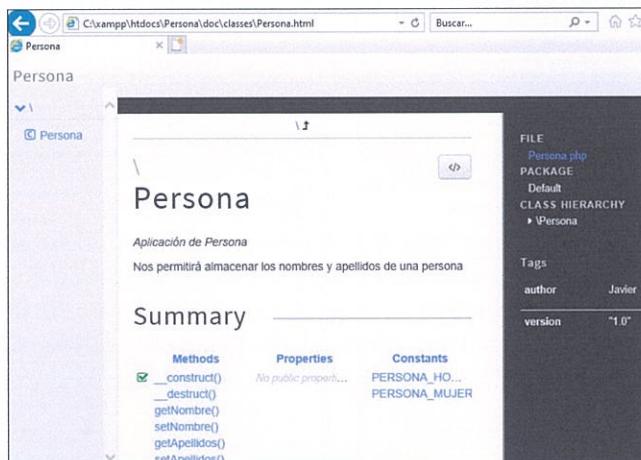


**Figura 6.6**  
Configuración phpDocumentor



**Figura 6.7**  
Documentación de la aplicación

- Con este último paso ya se está en disposición de generar la documentación de la aplicación. Para ello se pulsará con el botón derecho del ratón en la opción *Generar documentación* o *Generate Documentation*. Se generará una página HTML como la de la figura 6.8.



**Figura 6.8**  
Documentación final



### Actividad propuesta 6.3

Toma una aplicación como ejemplo y genera documentación con phpDocumentor, previamente se deben documentar las clases y sus métodos. Configurar también phpDocumentor en Netbeans 8.2.

### 6.2.3. Instalación, configuración y uso de Doxygen

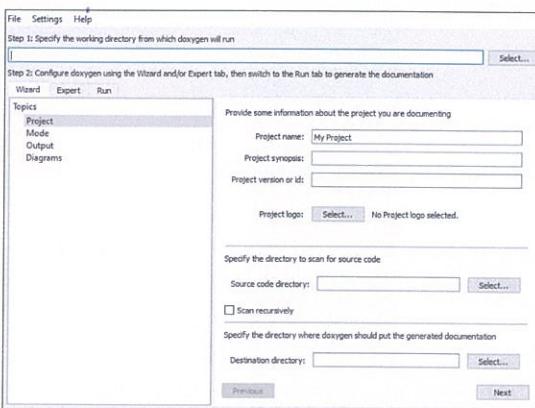
Como se comentó anteriormente, Doxygen es una de las herramientas más versátiles dentro de la generación de documentación, porque permite documentar un abanico amplio de lenguajes de programación. En nuestro caso, se va a tomar como ejemplo una de las prácticas de los módulos que se imparten en informática.

Primero de todo, es necesario descargar Doxygen desde la URL <https://www.doxygen.nl/download.html>. En tal página está disponible la última versión, que es la número 1.8.20 del 24 de agosto del 2020. Y están disponibles los binarios para Linux, Windows y Mac OS, además del código fuente. La forma de trabajar de Doxygen es en forma de asistente o experto, en nuestro caso, se va a trabajar de forma wizard para hacer más sencilla su comprensión.

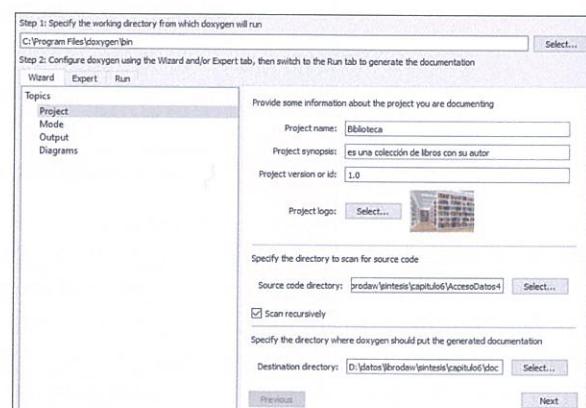
Una vez descargada la aplicación en Windows, sería necesario instalarla de una forma fácil e intuitiva. Y, si se ejecuta, se visualizará la interfaz de la figura 6.9.

La aplicación web es una colección de libros con su autor realizada en Java con sus clases y métodos. La configuración de la primera pantalla de Doxygen consistiría en los siguientes ítems. Se observará en la pantalla de la figura 6.10.

- *Project name*: nombre del proyecto.
- *Project synopsis*: breve descripción del proyecto.
- *Project version*: la versión de la aplicación.
- *Project Logo*: una imagen que represente el proyecto.
- *Source Code directory*: el directorio raíz de la aplicación.
- *Destination directory*: el directorio donde se almacenará la documentación generada.



**Figura 6.9**  
Doxygen



**Figura 6.10**  
Wizard Doxygen

Una vez completada la ventana anterior, se pulsa en *Next* y se visualizarán las siguientes opciones para configurar que corresponden al *Mode*:

- ✓ *Extraction mode*: se va a seleccionar *All Entities*.
- ✓ *Select programming language*: en nuestro caso es *Java*.

Si se pulsa *Next* se llegarán a las siguientes opciones, como se puede observar en la figura 6.11:

- *Output format to generate*: en nuestro caso, se selecciona HTML, y posee la opción de texto plano o navegación mediante panel. También permite la opción de cambiar el color de la documentación HTML.
- *Latex*: posee distintas opciones, se puede seleccionar como formato intermedio para enlazarlo con pdf.
- *Opciones de ficheros*: man pages, RTF o XML. Se seleccionan páginas de ayuda.

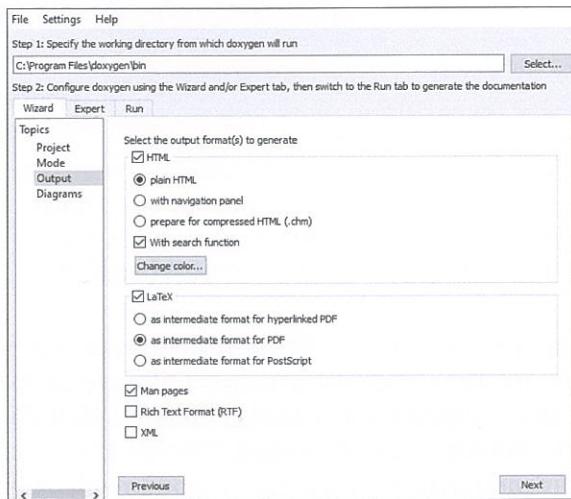


Figura 6.11  
Output Doxygen

La última ventana sería la de *Diagrams*. Lo útil sería seleccionar *Use built-in class diagram generator*, y se pulsa la opción *Next*. Ahora llega el momento de ejecutar Doxygen para permitir generar la documentación en HTML, por lo que se pulsará el botón *Run Doxygen*. En esta ventana se tiene la opción de mostrar toda la configuración mediante parámetros y también almacenar el log de salida (figura 6.12).

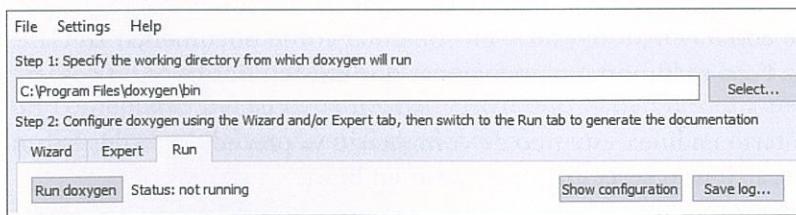
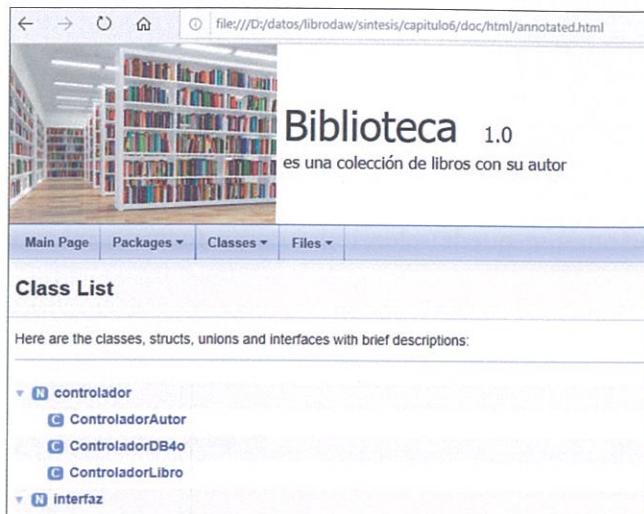


Figura 6.12  
Run Doxygen

Por último, se obtendrá toda la documentación que se ha configurado en la aplicación y en Doxygen, y se visualizará una página web como la de la figura 6.13.



**Figura 6.13**  
Documentación final

### 6.3. Formatos estándar para la documentación

En el mercado actual existe un conjunto amplio de formatos estándar para documentar, pero se establecen una serie de protocolos o medidas para que todo el mundo las lleve a cabo. De esta forma se pretende que cuando alguien lea el código entienda fácil e intuitivamente qué flujo lleva el código y cuál es su función a grandes rasgos.

Con relación a la nomenclatura, es crucial dar un nombre específico a las clases, métodos, variables, etc., para poder posteriormente identificar cualquier variable o clase dentro del código. Por ejemplo, a la hora de nombrar las variables sería interesante poner un identificador inicial que haga referencia al tipo de variable. Si se tiene un integer y la variable almacena una posición dentro de un array, lo ideal sería nombrarla *iposition* o *iPosition*, dependiendo de si se está codificando el código en inglés o en español.

Toda la documentación y nomenclatura se realiza para que el código pueda ser reutilizado en un futuro, sin necesidad de estudiar el código en profundidad. Por lo que sería necesario documentar de una forma correcta la aplicación y cada una de las clases que la componen.

Por ejemplo, en Java o en cualquier lenguaje de programación es necesario comentar las líneas de código. Los comentarios no cambian nada el código implementado, sino que simplemente ayudan al programador a comprender mejor las líneas de código del proyecto software. Más adelante, se entrará en profundidad en comentar cómo documentar las clases y los métodos implementados. Pero es importante mencionar que existen dos tipos de comentarios:

- Comentario en línea: este tipo de comentario va precedido de dos barras slash “//”. Por ejemplo, atributos sería un comentario en línea:

```
//Atributos
/**
 * Nombre del autor
 */
private String nombre;
```

- b) Comentario en multilínea o en bloque: estos comentarios van precedidos de “`/*`” y luego se escribiría texto y se finalizaría con “`*/`”. Todo el texto que va incluido entre estos símbolos se considera comentario. Por otro lado, si se quiere que lo interprete Javadoc para que se procese como documentación final de la aplicación web, es diferente. Habría que englobar los comentarios entre los símbolos “`/***`” y “`**/`”. Un ejemplo para que se observe sería de la siguiente forma:

```
/**
 * Clase Autor
 * Contiene información sobre el autor de los libros
 * @author javier
 * @version 1.0
 *
 */
```

Se han explicado varias herramientas de generación automática de documentación, pero todas tienen que tener un objetivo común, que es documentar de forma clara y concisa las clases. En primer lugar, se explicará cómo se documenta una clase, por ejemplo, de Java la lista de atributos serían los siguientes:

- ✓ *Nombre de la clase*: sería teclear el nombre de la clase de forma identificativa.
- ✓ *Descripción*: una descripción corta de qué contiene la clase.
- ✓ *Autor*: quién es el autor o autores de la implementación de la misma.
- ✓ *Versión*: es fundamental para comprobar cuántas versiones se han realizado de la misma, en caso de sufrir modificaciones.

Un ejemplo se puede observar en el siguiente código:

```
/**
 * Clase Autor
 * Contiene información sobre el autor de los libros
 * @author javier
 * @version 1.0
 *
 */
public class Autor {
```

A continuación, en la implementación del código vienen los atributos de la clase, que se teclearán con un comentario inicial de la forma `//Atributos`, y después una descripción corta de cada uno de ellos. Como se observa en el siguiente código:

```
//Atributos
 /**
 * Nombre del autor
 */
private String nombre;
 /**
 * Ciudad del autor
```

```

    */
private String ciudad;
/**
 * Lista de libros
 */
private List libros;

```

Si se sigue avanzando en la clase, es importante documentar el constructor o constructores, métodos y destructores de la clase. En este elemento sí habría que poner algunos campos más, y serían los siguientes:

- *Tipo de método*: público o privado.
- *Nombre del método o constructor*: descripción breve del método o constructor.
- *Parámetros*: las variables de entrada al método o constructor.
- *Variable de salida*: si el método devuelve algo.

Se puede observar en el siguiente código la documentación de constructor:

```

//Constructor
/**
 * Autor
 * @param nombre nombre del autor
 * @param ciudad ciudad del autor
 *
 */
public Autor (String nombre, String ciudad)
{
    this.nombre=nombre;
    this.ciudad=ciudad;
    this.libros=new ArrayList();
}

```

Se puede observar en el siguiente código la documentación de un método público:

```

//Métodos públicos
/**
 * setNombre
 * @param n nombre del autor
 */
public void setNombre(String n){
    this.nombre=n;
}
/**
 * getNombre
 * @return nombre nombre del autor
 */
public String getNombre(){
    return this.nombre;
}

```



### Actividad propuesta 6.4

Toma una aplicación como ejemplo y documenta la clase y sus métodos, como se ha explicado anteriormente. De manera opcional, se puede generar la documentación mediante Javadoc.

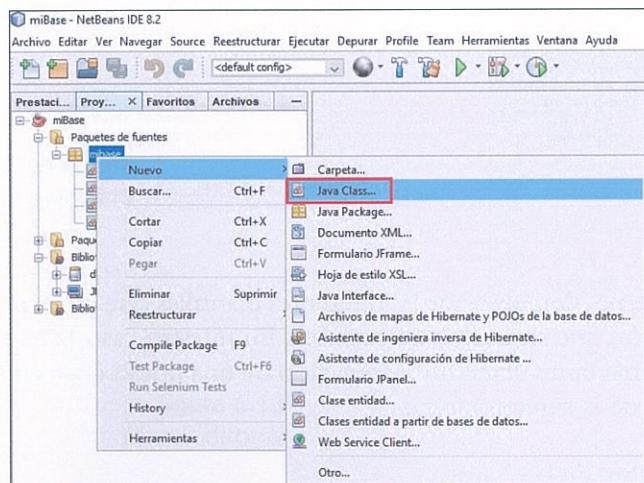
## 6.4. Creación y utilización de plantillas

Se ha explicado cómo realizar documentación de forma manual. También se puede realizar una documentación estructurada, y que se observe de forma intuitiva y amigable, para lo que se usa Javadoc, phpDocumentor o cualquier otra herramienta de generación de documentación. Pero existe otra posibilidad que permite programar las clases en Java con una plantilla o template por defecto.

El objetivo de estas plantillas es ahorrar tiempo sin tener que escribir un código que es literal y repetitivo, además de comentarios, como, por ejemplo, el autor de la clase, la versión, etc.

En cualquier IDE, como Netbeans, se usan las plantillas de archivo, de proyecto, de clases, etc. También existe la posibilidad de las sugerencias de código. En nuestro caso, se van a explicar las plantillas de archivo que permitirán crear la clase con unos comentarios y código por defecto, muy útiles a la hora de programar una aplicación con una serie de clases.

A partir de un proyecto creado, se va a crear una clase nueva, y se observará cómo se crean elementos y códigos por defecto. Si se pulsa en un paquete del proyecto en el botón derecho y, a continuación, en Nuevo y por último en Java Class, se observa la ventana 6.14.



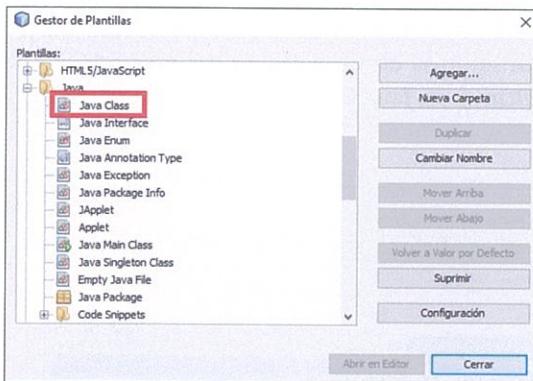
**Figura 6.14**  
Java Class

Una vez seleccionada esa opción, será necesario darle un nombre a la clase (nuestro caso, Editorial). Después de terminar el asistente aparecerá un archivo creado a partir de una plantilla. El archivo creado es algo parecido al de la figura 6.15.

## **Figura 6.15**

### Clase Editorial

En la imagen anterior hay que distinguir dos partes, una de ellas es la cabecera o header, en la que se pueden cambiar las propiedades del proyecto. Y otra sería la plantilla que genera el archivo (nuestro caso). Para editar, modificar, duplicar la plantilla se tiene que seleccionar la opción Herramientas y dentro de esta la subopción Plantillas o Templates. Aparecerá una ventana como la 6.16.



## Figura 6.16 Plantillas

En esta ventana se pueden observar las diferentes plantillas que existen de los diferentes lenguajes de programación que soporta Netbeans 8.2. En nuestro caso, la explicación se focalizará en el lenguaje Java y, más concretamente, la creación de una nueva clase en Java (Java Class). Una plantilla se compone de la siguiente lista:

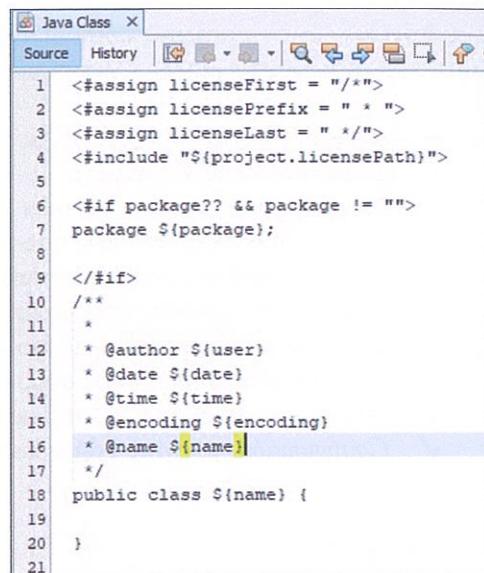
- *Nombre*: nombre de la plantilla.
  - *Descripción*: breve descripción de la plantilla.
  - *Cuerpo de la plantilla*: es un conjunto de texto fijo y variables que se pueden usar para que, cuando se cree un elemento del tipo de plantilla, se generen automáticamente. Algunas de las variables son las siguientes:
    - \${user}: inserta el nombre del usuario.
    - \${date}: inserta la fecha actual, con el formato 04-sep-2020.

- \${time}: inserta la hora actual en la cual se crea el fichero de la clase.
- \${encoding}: coloca la codificación de caracteres UTF-8.
- \${name}: visualiza el nombre del fichero sin extensión.

Para colocar las variables anteriores en la plantilla, se selecciona *Java Class* y luego *Abrir en Editor*. Con ello se visualizará una pantalla como la 6.17.

Si ahora se crea una nueva clase llamada Editorial, tomará como base la plantilla creada anteriormente, y el resultado será el de la figura 6.18.

Por otro lado, si se quieren modificar las propiedades del usuario, por ejemplo, su nombre, sería necesario pulsar en la opción Configuración de la figura 6.16, y se visualizará una pantalla como la 6.19.

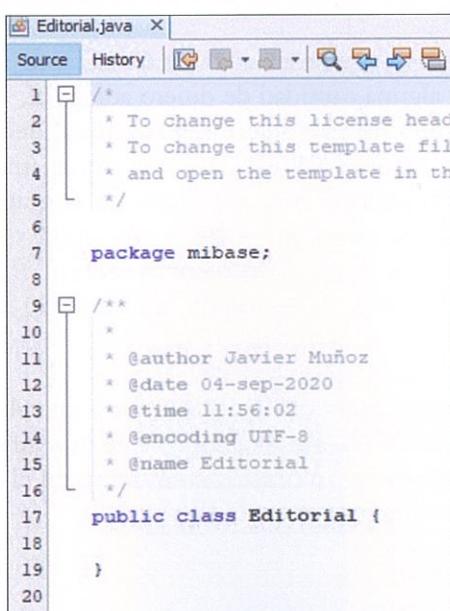


```

1 <#assign licenseFirst = "/*">
2 <#assign licensePrefix = " * ">
3 <#assign licenseLast = " */>
4 <#include "${project.licensePath}">
5
6 </if package?? && package != "">
7 package ${package};
8
9 </if>
10 /**
11 *
12 * @author ${user}
13 * @date ${date}
14 * @time ${time}
15 * @encoding ${encoding}
16 * @name ${name}
17 */
18 public class ${name} {
19 }
20

```

**Figura 6.17**  
Plantilla Java Class

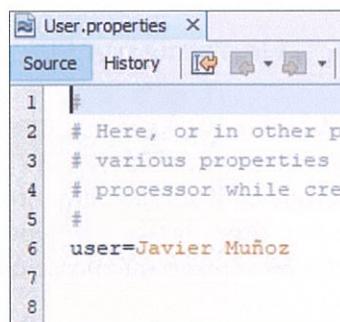


```

1 /*
2  * To change this license head
3  * To change this template fil
4  * and open the template in th
5  */
6
7 package mibase;
8
9 /**
10 *
11 * @author Javier Muñoz
12 * @date 04-sep-2020
13 * @time 11:56:02
14 * @encoding UTF-8
15 * @name Editorial
16 */
17 public class Editorial {
18
19 }

```

**Figura 6.18**  
Clase Editorial



```

1 #
2 # Here, or in other p
3 # various properties t
4 # processor while cre
5 #
6 user=Javier Muñoz
7
8

```

**Figura 6.19**  
User.properties

## Recurso web

Accede a la Wiki de Netbeans para profundizar sobre las variables de las plantillas.



Además de las opciones comentadas, la ventana de las plantillas (figura 6.16) permite las siguientes opciones:

- ✓ *Agregar*: permite agregar un fichero donde esté definida una plantilla en la que el tipo de fichero debe ser .java.
- ✓ *Nueva carpeta*: crea una carpeta para generar plantillas.
- ✓ *Duplicar*: permite duplicar una plantilla ya creada.
- ✓ *Cambiar nombre*: cambia el nombre a una plantilla.
- ✓ *Mover arriba*: mueve hacia arriba una plantilla.
- ✓ *Mover abajo*: mueve hacia abajo una plantilla.
- ✓ *Volver a valor por defecto*: regresa a los valores por defecto antes de modificar nada.
- ✓ *Suprimir*: elimina una plantilla creada anteriormente.
- ✓ *Configuración*: permite editar las propiedades del usuario.

## 6.5. Herramientas colaborativas para la elaboración y mantenimiento de la documentación

Actualmente, en el mercado existe un abanico amplio para usar herramientas colaborativas para elaborar documentación, tanto open source como de pago. Hay varias herramientas que son bastante interesantes, pero creo que una de las mejores es Slack, que posee una versión bastante potente gratuita, aunque si se quiere ampliar habría que pagar alguna cantidad de dinero adicional.

Y, por otro lado, la mayoría de las empresas de ámbito nacional e internacional dedicadas a la implementación de software usan el paquete Microsoft Office 365, que posee una gran variedad de opciones, desde el paquete básico hasta el paquete Premium (esta no es gratuita).

### 6.5.1. Herramienta Slack

Con relación a Slack, es una de las herramientas que más se usa en el mercado actual, sobre todo en equipos de trabajo para compartir información y documentación. Posee una dificultad añadida, que es el amplio conjunto de posibilidades que tiene, pero en el momento en el que el equipo se acostumbra es realmente útil y práctico su funcionamiento.

Entre las ventajas que posee están las siguientes:

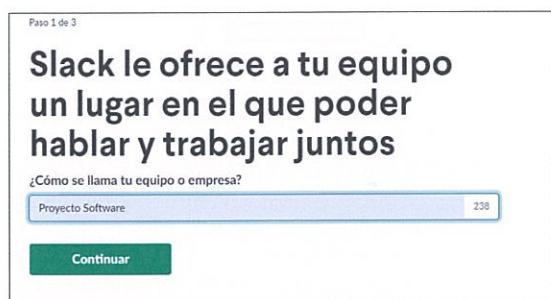
- Se puede compartir documentación, vídeos e imágenes desde tu equipo o desde Google Drive.
- Posee integración con otras aplicaciones del entorno de trabajo remoto como Trello, herramientas diarias como Gmail y Google Calendar, aplicaciones esenciales GoToMeeting y Salesforce, incluso Robots como Troops.
- Versiones de escritorio, web y móvil.
- Permite crear canales para controlar un proyecto determinado y ver los detalles de tal canal.
- Personalización de tus notificaciones, tiempo, sonido, etc.
- Contiene un generador de workflow, fundamental en la realización de un proyecto software o aplicación web.

Por todo ello, es una herramienta ideal colaborativa y lo suficiente potente para comenzar a trabajar. Si bien es cierto, que si necesitas más almacenamiento o mayores medidas de seguridad existen dos opciones de pago, que son las versiones Standard y Plus.

Para observar la herramienta se comienza con ingresar en la URL <https://slack.com/signin> y se visualizará una pantalla donde se tendría que ingresar tu email.

Una vez que se introduce el correo electrónico y se pulsa Continuar te has dado de alta en Slack. Posteriormente para evitar spam y ataques innecesarios te envían un código de seis números al correo electrónico que se ha tecleado. Al introducirlo en la URL que se visualiza aparecerá una página donde se tecleará el nombre del equipo software (figura 6.20).

En la siguiente página, te pedirá introducir alguien más del equipo de trabajo: simplemente se puede omitir o poner alguien que forme parte del equipo de trabajo, tecleando el email del mismo. También se puede enviar un enlace de invitación.

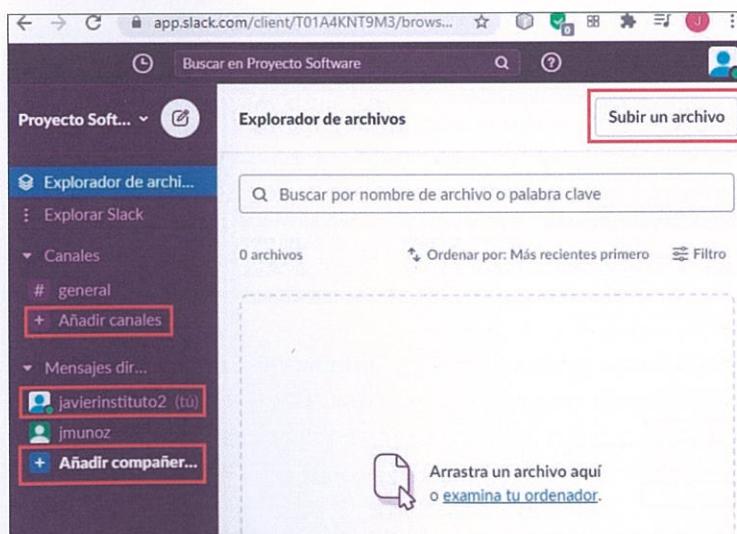


**Figura 6.20**  
Equipo de trabajo



**Figura 6.21**  
Mensaje al equipo

Por último, se envía un mensaje al equipo y se está listo para comenzar a trabajar (figura 6.23). En este momento, ya está el equipo preparado para comenzar el proyecto software. La página principal del proyecto es la que se muestra en la figura 6.22.



**Figura 6.22**  
Proyecto Software

En la pantalla de la figura anterior se tiene una serie de opciones que se van a explicar a continuación:

- ✓ *Canales*: permite abrir un canal para un tema determinado o por un problema del proyecto o cualquier otra anomalía.
- ✓ *Equipo*: se puede añadir a compañeros de equipo en cualquier momento.
- ✓ *Archivos*: se pueden subir archivos de importancia para el proyecto.
- ✓ *Aplicaciones*: nos permite acceder a cualquier tipo de aplicación de las que se han visto anteriormente.

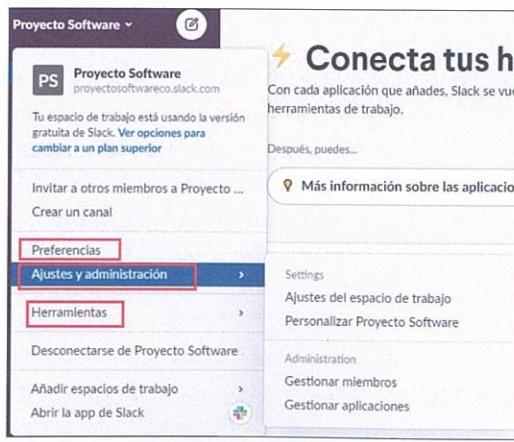
Este último apartado es interesante, ya que permite llegar a las aplicaciones a las que se puede acceder desde Slack. Si se pulsa en la opción *Explorar Slack* de la figura 6.22 se visualizará la pantalla 6.23.



**Figura 6.23**  
Aplicaciones Slack

Además, se puede navegar al directorio de aplicaciones y se puede observar el listado de aplicaciones disponibles.

Por otro lado, dentro del *Proyecto Software* de la figura 6.22 se puede observar la imagen 6.24.



**Figura 6.24**  
Configuración Slack

Dentro de esta lista de opciones, algunas de ellas ya se han comentado. Las principales son las siguientes:

- a) Preferencias: permite configurar las notificaciones, la barra lateral de iconos rápidos, los temas que se pueden configurar, los mensajes y los medios, el idioma y la región, la accesibilidad, marcar como leído y avanzados que está relacionado con las opciones que existen para escribir código, para buscar alguna información en los canales, etc.
- b) Ajustes y administración: está relacionado con los ajustes y autorizaciones del espacio de trabajo. Existen algunas opciones de esta ventana que son de pago como la autenticación o algunos permisos específicos, por ejemplo, las solicitudes en canales compartidos.
- c) Herramientas: es una opción bastante interesante, dado que permite conocer los mensajes enviados, almacenamiento usado (límite 5 Gb) y aplicaciones e integraciones instaladas.



### Actividad propuesta 6.5

Crea un espacio de trabajo en Slack que se llame Aplicación Mutante, y mediante dos correos electrónicos forma un equipo de trabajo para intercambiar mensajes, e instala alguna de las aplicaciones. Todo de forma gratuita para observar el manejo de esta herramienta tan útil.

#### 6.5.2. Herramienta Microsoft Office 365

En el tejido productivo empresarial y sobre todo en el mundo de la informática, Office 365 es una de las herramientas punteras, pero tiene el inconveniente que es de pago totalmente, por lo que se sugiere usar la anterior, pero en caso de que exista presupuesto, se puede utilizar perfectamente.

Office 365 es un entorno de colaboración que aglutina una serie de aplicaciones cruciales a la hora de acceder, compartir documentos. Además de los ya conocidos, como Word, OneNote, Excel o Powerpoint. Algunas de las ventajas de este producto son las siguientes:

- *OneDrive*: es un lugar de almacenamiento seguro en la nube, donde se podrá acceder a la información en cualquier momento, siempre que se disponga de una conexión a Internet. Además permite realizar un backup, en caso de pérdida o deterioro de información.
- *Seguridad y compatibilidad*: la seguridad se basa en que la información está cifrada, por lo que impide la lectura de la información en cualquiera de las aplicaciones que la componen. En cuanto a la compatibilidad, se puede acceder desde cualquier navegador y sistema operativo.
- *Acceso*: se puede acceder desde cualquier dispositivo, desde un Smartphone, una Tablet, portátil, etc. Por lo que su uso se puede dar en cualquier momento.
- *Colaboración*: se puede trabajar con varias personas del equipo de trabajo al mismo tiempo sobre un documento de trabajo.

Para contratar esta herramienta sería tecleando la URL <https://www.microsoft.com/es-es/microsoft-365?rtc=1>, y partir de aquí sería contratar un plan y empezar a funcionar con esta gran herramienta.

Para concluir, se van a listar las aplicaciones que posee tal entorno de trabajo, además de las propias de Microsoft Office:

**CUADRO 6.1**  
Otras aplicaciones

Aplicación	Descripción
Delve	Colaborar con otros usuarios para organizar la información.
Forms	Crear formularios y analizar los resultados obtenidos.
Kaizala	Tener una agenda de trabajo para el día a día, aplicación móvil.
Microsoft Teams	Colaborar en equipos de trabajo, crear canales, iniciar chats, llamadas y reuniones. Todo ello securizado.
OneDrive	Compartir, sincronizar y almacenar información con otros usuarios del equipo de trabajo.
OneNote	Realizar notas, dibujos y poder compartir tal información.
Planner	Obtener consejos de expertos para realizar un evento.
Publisher	Diseño de calendarios, boletines e información de marketing.
SharePoint	Punto de compartición de información sobre noticias, publicaciones, etc.
Skype Empresarial	Organizar reuniones online y llamadas para el equipo de trabajo.
Stream	Compartir vídeos y clasificarlos.
Sway	Crear todo tipo de documentos como textos, imágenes, etc.
To-Do	Agenda con eventos de vencimiento y sincronizada con Outlook.
Whiteboard	Lienzo digital para dibujar de forma libre.
Yammer	Conectarse con el equipo de trabajo y aprovechar al máximo su rendimiento.

## 6.6. Instalación, configuración y uso de sistemas de control de versiones en SO Windows

Antes de comenzar a instalar y configurar un sistema de control de versiones es necesario comentar conceptos básicos que permiten controlar las diferentes versiones del ciclo de vida de una aplicación web o un software implementado mediante cualquier lenguaje de programación.

Cuando no existían sistemas de control de versiones, se programaba en un directorio y, si se modificaba algo, era necesario realizar un backup del directorio. Esto se hacía para volver hacia atrás en caso de que la modificación no fuera la correcta. Lo que significa que la forma de trabajar en unos veinte años ha cambiado considerablemente y, en la actualidad, es más fácil programar y reutilizar un código anteriormente programado.

### 6.6.1. Conceptos básicos

El sistema de control de versiones permite la modificación de las distintas partes o componentes de una implementación de un software específico. La versión es el estado en el cual se encuentra el producto que se está desarrollando en un punto del ciclo de vida del proyecto software.

Es interesante conocer algunos conceptos vitales para comprender el funcionamiento del control de versiones y para que el equipo de trabajo conozca la nomenclatura de cómo se trabaja en control de versiones software. Para ello se van a definir los siguientes conceptos:

- a) *Repositorio (repository)*: el lugar de almacenamiento de las distintas versiones del software que está desarrollando el equipo de trabajo. Puede ser la parte de la interfaz, una base de datos, una conexión, todo lo relacionado con la implementación software.
- b) *Rama (branch)*: cada programador tiene su copia de trabajo en local y realiza modificaciones sobre una rama en concreto del proyecto software. Incluso si es necesario puedes descargar una copia de todo el proyecto y modificar la rama del proyecto asignada sin perjudicar a nadie. En caso de que los cambios sean correctos, se pueden subir al repositorio notificando el cambio.
- c) *Conflicto*: se produce cuando varios programadores han realizado varios cambios en la misma rama del software y son contradictorios. Por lo que el control de versiones te avisa del conflicto, y es necesario solucionarlo hablando con tus compañeros de equipo. En caso de duda, el jefe de proyecto tendrá la última palabra.
- d) *Cambio (change)*: se realiza cuando se ha modificado la parte asignada a un programador y es subida al repositorio controlando este cambio el control de versiones.
- e) *Revisión (revision)*: es una versión del software y permite al sistema control de versiones controlar las distintas revisiones del software realizadas. Normalmente, se identifica con un contador o también con alguna etiqueta identificativa.
- f) *Confirmar (commit)*: confirmación de que se han realizado varios cambios en el mismo documento.

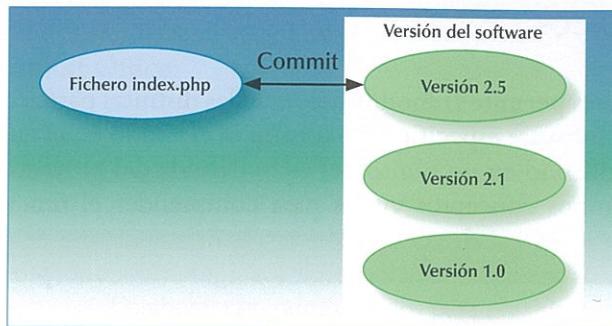
### 6.6.2. Funcionamiento del control de versiones

El sistema de control de versiones funciona con un repositorio local o remoto, que es donde se encuentra todo el código del proyecto, y con un cliente que se conecta a tal repositorio. En esta conexión se producen los diferentes cambios, commit, y las distintas operaciones que ofrece el cliente sobre el repositorio. Algunas de las herramientas que existen en el mercado son Subversion, Mercury, Git, Perforce, etc.

Existen varias formas de trabajar, dependiendo de la localización del repositorio y de la infraestructura que posea la empresa o el equipo de trabajo que use este tipo de sistemas. Se pueden clasificar los sistemas de control de versiones atendiendo a la arquitectura usada en tres tipos:

#### A) Locales

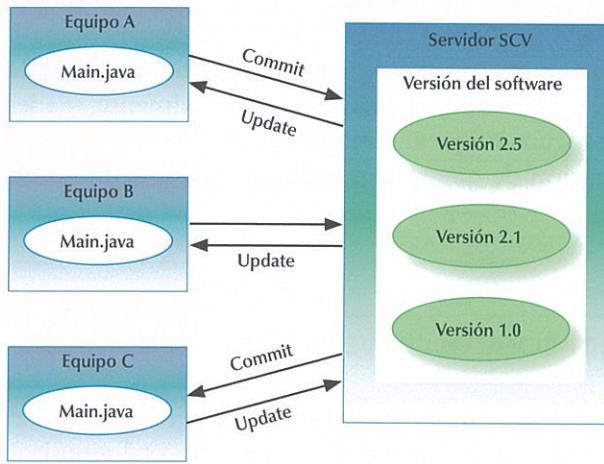
Es la versión más simple de la arquitectura de los sistemas, ya que la copia del proyecto software no se comparte con nadie. Es como si se hiciera un backup en local. Pero es la base de los siguientes tipos. Es posible hacerse una idea de esta arquitectura al observar la figura 6.25:



**Figura 6.25**  
Sistema local

### B) Centralizados

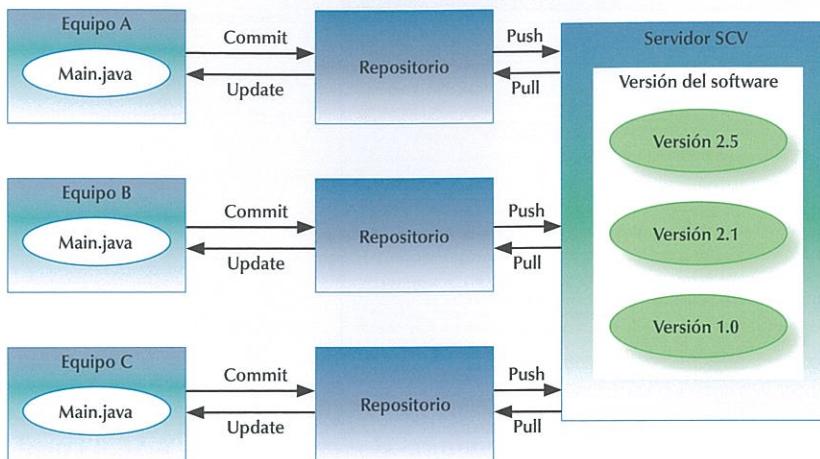
Esta arquitectura consiste en un servidor central que contiene el repositorio con todo el código, y este está asignado a un usuario, que normalmente es el jefe del proyecto software o la persona encargada de dar el visto bueno. Todas las operaciones que se hagan, como ramificaciones, o modificaciones de gran calado, pasan por su aprobación. Ejemplos de ellos son Github, Subversion, etc. Se puede ver la forma de trabajar observando la siguiente figura:



**Figura 6.26**  
Sistema centralizado

### C) Distribuidos

En este caso, la noción de repositorio central no existe, sino que cada equipo de trabajo tiene su propio repositorio. Los repositorios que existen en cada equipo se pueden intercambiar y fusionar revisiones entre ellos. Debería existir un repositorio disponible, a partir del cual se pueden sincronizar los demás. En esta opción es necesario que el equipo de trabajo controle perfectamente esta forma de trabajar, de lo contrario, pueden ocurrir interferencias. Tiene la ventaja de que no es necesario conectarse al servidor central. Por ejemplo, Git puede trabajar de esta forma. Para que el programador se haga una idea, la arquitectura sería la siguiente:



**Figura 6.27**  
Sistema distribuido



Se puede consultar este artículo de la página web de Git para profundizar sobre los sistemas de control de versiones centralizados.



### 6.6.3. Instalación y configuración de Git en Netbeans

Antes de comenzar con la instalación, se necesita recordar que se va a configurar una arquitectura del tipo distribuida, en la que el cliente va a ser Git en Netbeans y el repositorio central Github.com, que se puede acceder mediante la URL <https://github.com/>.

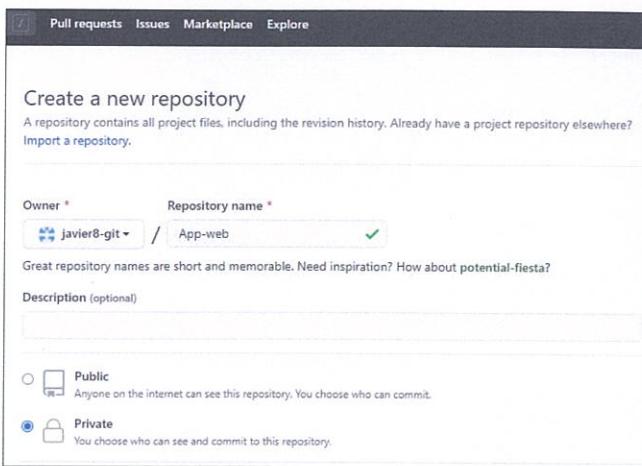
A partir de estas nociones, se va a configurar e instalar el sistema de control de versiones en Netbeans 8.2 y Github de la siguiente forma:

1. Lo primero sería dar de alta un usuario en Github mediante un login que no exista, un correo electrónico y una password. Posteriormente, Github te hace algunas preguntas y verifica que eres quien dices ser mediante el envío de un correo al email que se ha configurado en la página (figura 6.28).



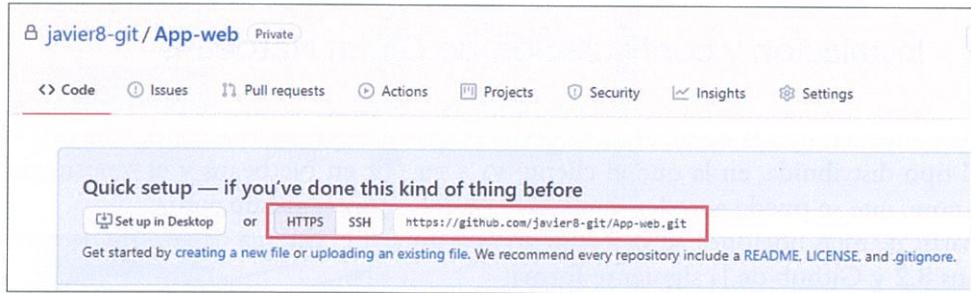
**Figura 6.28**  
Github

2. Se pulsa *Start a new Project* y se visualizará una página como la de la figura 6.29.



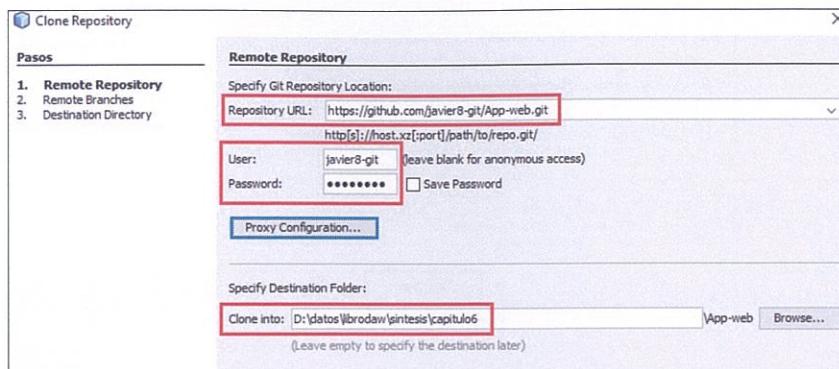
**Figura 6.29**  
Creación de repositorio

3. En la figura 6.29 se visualizan varias opciones. La URL da la opción de importar un repositorio ya existente, el nombre de nuestro repositorio (App-web); muestra dos opciones, Private o Public. Si nuestro repositorio es público: cualquier persona de Internet puede ver nuestro repositorio, pero el propietario del repositorio acepta o no las modificaciones. Y, por otra parte, si es privado, el propietario elige quién observa el repositorio. En nuestro caso, se va a elegir Private. También Github da la opción de inicializar el repositorio con un Readme, de ignorar una lista de ficheros e incluso elegir una licencia. Con las opciones anteriores seleccionadas, se pulsa en Create repository, y se visualizará una pantalla como la 6.30.



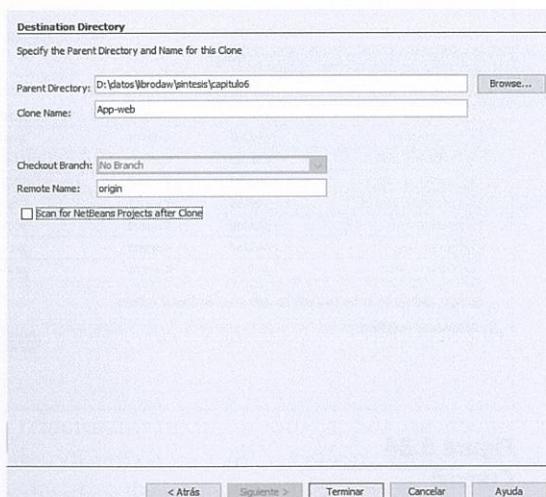
**Figura 6.30**  
Repositorio App-web

4. Dos conceptos se van a tratar, el primero de ellos es que para sincronizar nuestro proyecto con el repositorio se configura en el escritorio o se copia la URL que está en el cuadro rojo y se configura posteriormente en Netbeans, como se explicará más adelante. Es necesario o recomendable crear un Readme, License y una lista de ficheros para ignorar en caso de que existiese. En nuestro caso, se copiará la URL para configurarla posteriormente en Netbeans.
5. El siguiente paso sería abrir un proyecto (en nuestro caso, Java) y, a continuación, sería abrir la opción Team, después la subopción Git y, por último, la elección Clone... Una vez seleccionada esta opción, se visualizará una pantalla como la 6.31.



**Figura 6.31**  
Configuración Netbeans

6. En la figura 6.32 se puede observar la configuración que se ha realizado. Lo primero de todo sería teclear la URL del repositorio que se ha copiado de Github. Por otro lado, hay que teclear el usuario y la password del repositorio. Se tiene la opción de configurar un proxy y de probar si la conexión funciona correctamente. Y, por último, se puede configurar un directorio de clonado. Ya se está en disposición de pulsar en Next, y se visualizará una pantalla para seleccionar las ramas del software. Como es la primera vez que se sincroniza, no es necesario seleccionar nada, por lo tanto, se pulsa de nuevo Next. Se visualizará la pantalla 6.32.



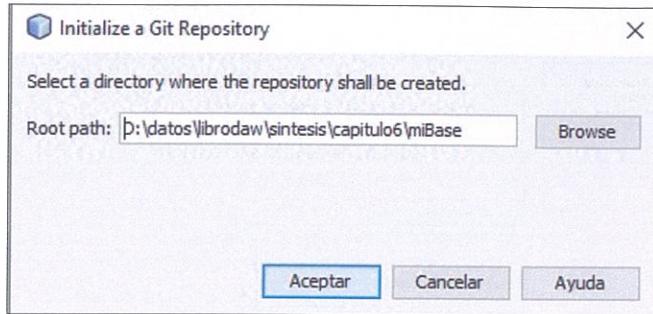
**Figura 6.32**  
Configuración Netbeans II



### Actividad propuesta 6.6

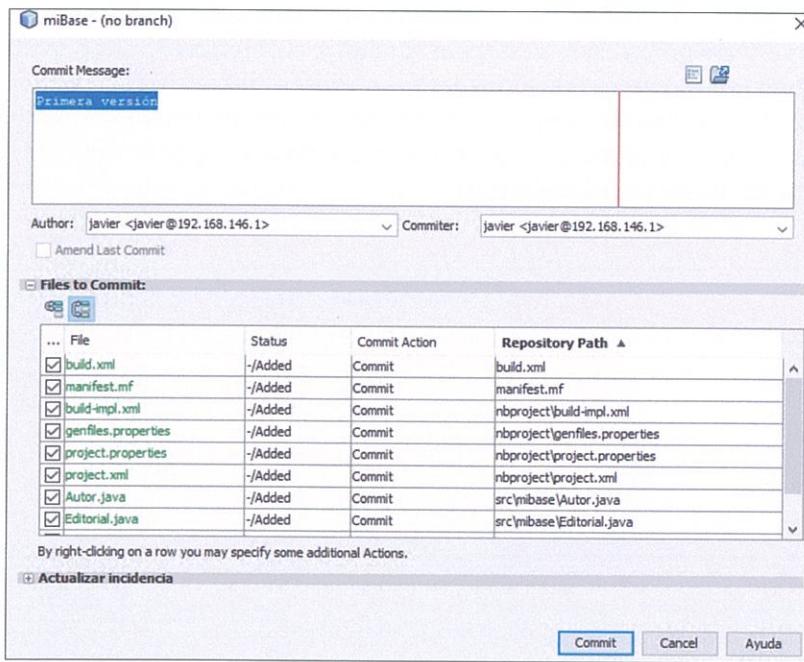
Configura Git con el repositorio Github o con cualquier otro y prueba que funciona correctamente.

7. Una vez configurado Git, se inicia el repositorio en la opción Initialize Repository, y se visualizará la pantalla 6.33.



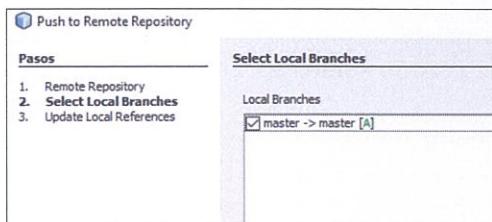
**Figura 6.33**  
Inicializar repositorio

8. El siguiente paso es pulsar en el nombre del proyecto con el botón derecho del ratón y seleccionar Git y, posteriormente, Commit. Se visualizará la ventana 6.34.

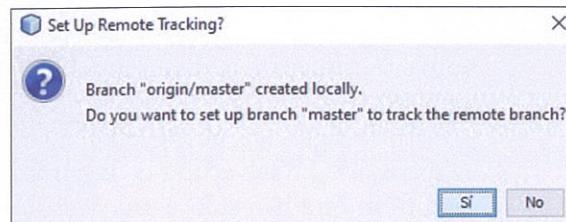


**Figura 6.34**  
Commit

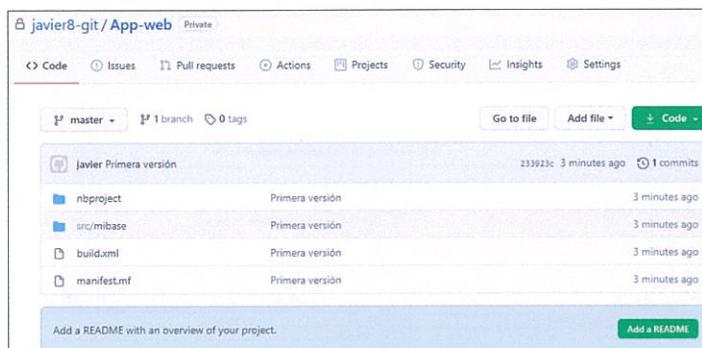
9. Será necesario realizar un push para subir el código al repositorio central. Esta operación se puede realizar de varias formas, una de ellas sería botón derecho del ratón pulsando en el proyecto y Git → Remote- → Push. Si se pulsa la opción anterior, se observará la ventana 6.35.
10. En la figura 6.35 se selecciona la rama master para poder enviarla al repositorio central de Github. Se observará otra pantalla parecida en la que se seleccionará también la opción de master. Si se pulsa Siguiente, se observará la pantalla 6.36.
11. En la figura anterior está preguntando si se creará una rama master o principal localmente, y si se quiere configurar esta rama para controlar la rama remota. Por lo que la respuesta es Sí. Con la operación anterior, ya está el código subido del proyecto a Github. Y se tiene la primera versión del software (figura 6.37).



**Figura 6.35**  
Operación Push



**Figura 6.36**  
Operación Push



**Figura 6.37**  
Github



### Actividad propuesta 6.7

Una vez configurado Git en Netbeans, toma como ejemplo un proyecto y súbelo por primera vez a tu repositorio en Github u otro sistema de control de versiones.

## 6.7 Operaciones avanzadas

Git permite todo tipo de operaciones relacionadas con el código, por lo que se van a explicar las más importantes y avanzadas. Antes de comenzar, se van a explicar algunos de los colores que se asignan a ficheros según la acción realizada sobre ellos:

- **Mibase.java:** no han existido cambios.
- **Mibase.java:** el fichero ha sido modificado localmente.
- **Mibase.java:** se han añadido más líneas de código en el fichero.
- **Mibase.java:** existe un conflicto en las versiones del fichero.
- **Mibase.java:** el fichero es ignorado por Git y no será incluido en los comandos del sistema de control de versiones.

Las operaciones son las siguientes:

- a) *Comparar versiones de software:* esta opción se realiza a partir de la opción *Team → DIFF*. Dentro de esta opción existen cuatro subopciones: *Diff to Head*, *Diff to Tracked*, *Diff to*

*Repository Head y Diff to...* La primera opción permite comparar la versión actual del software (situada a la derecha del panel) con la versión subida en el repositorio (figura 6.38).

En la figura se puede observar cómo se comparan las dos versiones del fichero MiBase.java. En la versión de la parte izquierda se pueden observar dos líneas rojas que significan que han sido eliminadas de una versión a otra. Y las líneas azules significan que han sido modificadas desde la última versión.

```

MiBase.java [Diff] X
Diff: Local Changes          to: HEAD
File name      Status      Path
MiBase.java    Modified     src/mibase/MiBase.java

Graphical Textual

HEAD           Modified In Working Tree
---           -----
 1 and open the template in the editor.
 2
 3 package mibase;
 4 import com.db4o.*;
 5 /**
 6  * @author javier
 7  */
 8 public class MiBase {
 9
10 /**
11  * @author javier
12 */
13 public class MiBase {
14
15 /**
16  */
17 public static void main(String[] args) {
18     ObjectContainer db = Db4o.openFile("src\\mibase\\MiBase");
19     crearDB(db);
20
21     db.close();
22 }
  
```

Figura 6.38  
Diff

- b) *Reverting Changes*: permite eliminar los cambios realizados. Sirve para volver hacia atrás en caso de algún error cometido en una nueva versión del software y, más concretamente de un fichero. La opción se selecciona mediante *Team → Revert Modifications*. Por ejemplo, si se borra un comentario y se graba para subirlo, y se necesita volver hacia atrás, esta opción lo permite.
- c) *Branch (rama)*: Git permite trabajar con ramificaciones del proyecto, incluso unirlas y borrarlas. Primero se va a crear una rama y a partir de ahí se puede realizar las operaciones que se han comentado. El objetivo de esta opción es el mantenimiento de varias ramas sobre un mismo código de base sobre el que se trabaja. Se parte del código que existe en la actualidad y se pulsa la opción *Team → Branch/Tag → Create Branch*, visualizándose la pantalla de la figura 6.39.

Se ha creado una nueva rama llamada 1-Etapa que se puede observar con el comando *Team → Checkout → Checkout Revision*. Dentro de la ventana que se observará es necesario seleccionar la opción *Select*, y se visualizará la imagen donde se observarán las distintas ramas y versiones que existen tanto en local como en re-

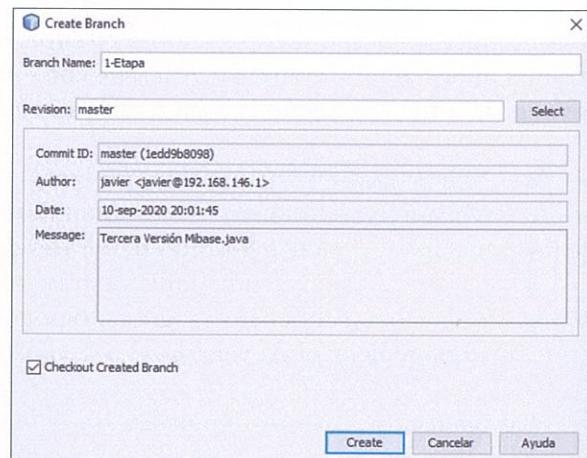
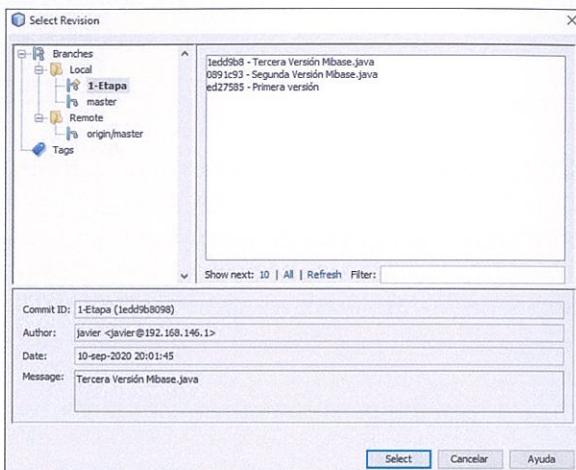


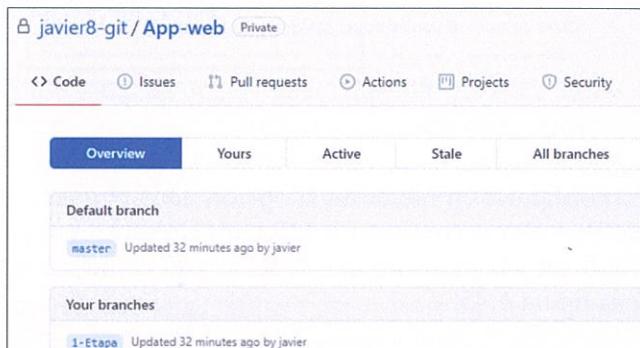
Figura 6.39  
Branch

moto. Además, se visualizan los datos del autor del commit, la identificación del commit, la fecha y la descripción que se ha realizado (figura 6.40).

Si se observa atentamente en remoto no está todavía subida la rama creada denominada 1-Etapa. Siempre que se desee subir algo al servidor de control de versiones es fundamental seleccionar la opción de *Push*. En la figura 6.41 ya se pueden observar las dos ramas en Github:



**Figura 6.40**  
Checkout



**Figura 6.41**  
Branch

- d) *Merge/patch*: se aplican los cambios realizados un fichero .java a otro. Normalmente, se utiliza cuando se tienen varias ramas, y algunas de ellas se necesitan tener en el mismo fichero.
- e) *Conflict*: se produce en un equipo de trabajo cuando dos programadores modifican el mismo fichero o recurso y provocan que el cambio sea distinto en el mismo código. Github te avisa y se soluciona el problema aceptando el cambio más correcto para el desarrollo de la aplicación.



Para profundizar con Git, sería necesario consultar este artículo de su página web.

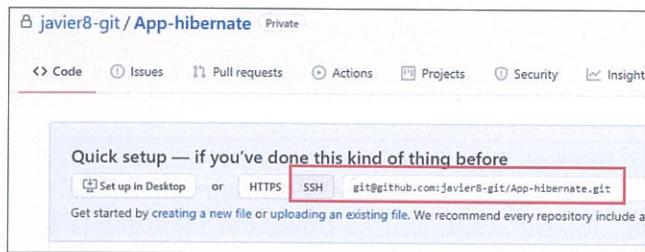


## 6.8. Seguridad de los sistemas de control de versiones

Como se ha comentado anteriormente, Git se encuentra dentro de los sistemas de control de versiones distribuidos. En cada equipo de trabajo se tiene una copia del repositorio que se conecta a la red para acceder al repositorio Github central. Pero esta forma de trabajar puede ser cambiada en cualquier momento con la inclusión de, por ejemplo, Subversion. El acceso se realiza mediante dos protocolos que pueden ser configurados en Netbeans al inicio de la configuración:

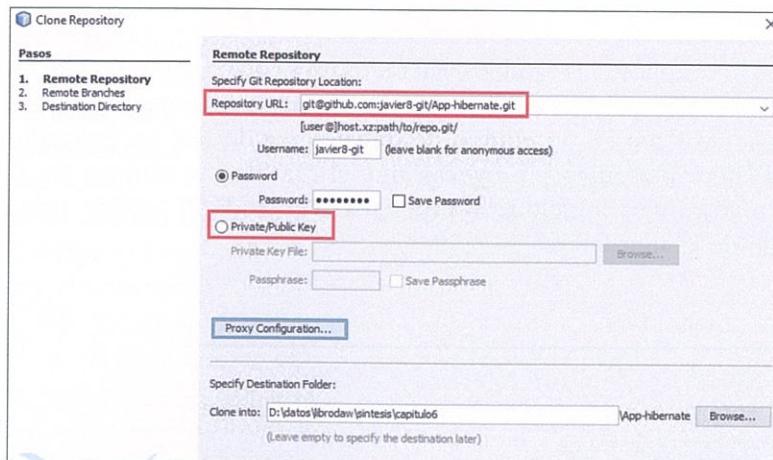
### 6.8.1. Protocolo SSH

Permite la conexión entre dos equipos de forma segura. La implementación sería creando en Github un repositorio y seleccionando la opción SSH. Como se puede observar en la figura 6.42 y, más concretamente, en el recuadro rojo:



**Figura 6.42**  
SSH

Es necesario configurar en Netbeans la opción de SSH seleccionada en el servidor de control de versiones. Para ello se configura la URL `git@github.com:javier8-git/App-hibernate.git` y el usuario `javier8-git` y la password que se ha creado en <https://github.com/>. La imagen será parecida a la de la figura 6.43.



**Figura 6.43**  
SSH Netbeans

Con esto terminaría la configuración de SSH y ya solamente se tendría que inicializar el repositorio, hacer el primer commit del proyecto correspondiente y realizar el comando push, y ya se copiaría el proyecto completo en el servidor de control de versiones. Además, existe la posibilidad de crear una clave privada para encriptar las comunicaciones (habría que crearla en local y también en el servidor). Por otro lado, también se pondría una palabra de paso que aumentaría la dificultad para descifrar la información.

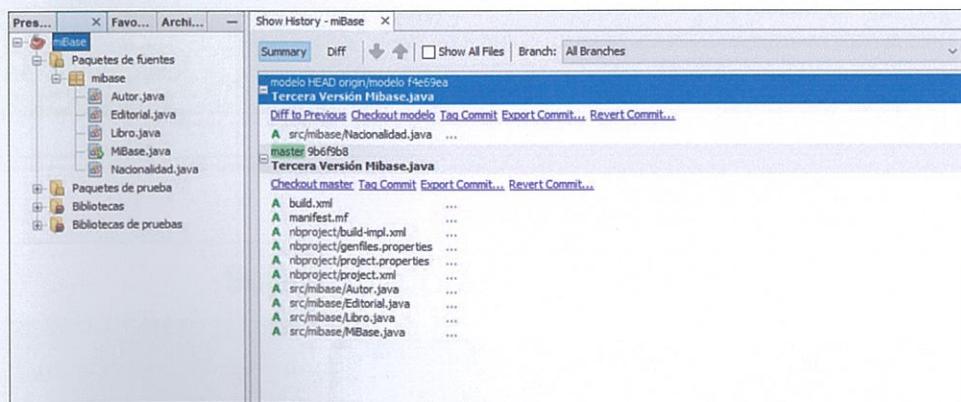
## 6.8.2. Protocolo HTTPS

Esta opción permite realizar el mismo procedimiento que el anterior, pero en lugar del protocolo SSH sería con el protocolo HTTPS. Es similar al protocolo HTTP de páginas web, pero con la inclusión del protocolo SSL, que permite encriptar las comunicaciones entre el cliente y el servidor. Este procedimiento está explicado en las figuras 6.30 y 6.31.

Además de estas configuraciones, Git permite configurar los dos protocolos mediante un usuario y contraseña que permitirá solamente conectarse a aquellos programadores a los que se dé acceso. Así como la configuración de un proxy de salida a Internet por parte de la compañía o empresa en la que se esté desarrollando el software de la aplicación.

## 6.9. Historia de un repositorio

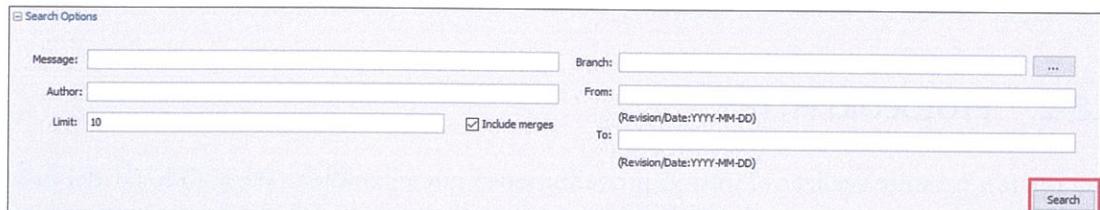
Después de haber llevado a cabo todas las operaciones anteriores, tanto básicas como avanzadas, Git permite la posibilidad de conocer todos los cambios realizados, ramas creadas y algunas opciones más. Para ello simplemente es necesario seleccionar el nombre del proyecto con el botón derecho del ratón y a partir de ahí *Git → Show History*. Aparecerán unas pantallas como las siguientes pulsando *Search* (figura 6.44):



**Figura 6.44**  
Show History

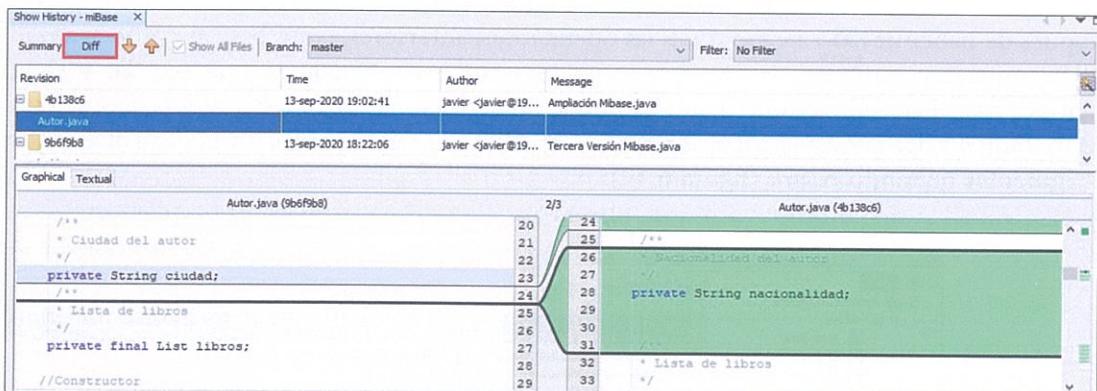
En la figura 6.44 se visualizan las dos ramas creadas, tanto master como modelo, en la que aparece que existe solamente una diferencia entre las dos ramas, es la creación del fichero Nacionalidad.java. En la figura 6.45 se pueden observar todas las opciones para filtrar la información, como son las siguientes:

- *Message*: filtrado por mensajes escritos cuando se realiza un commit de una rama concreta.
- *Author*: es el programador del equipo de trabajo que ha realizado la operación commit.
- *Branch*: búsqueda por un nombre de una rama determinada. En nuestro caso, se tienen dos ramas, máster y modelo.
- *From and To*: entre fechas de operaciones realizadas sobre una rama determinada.



**Figura 6.45**  
Búsqueda de eventos

Para concluir este apartado, se va a modificar el fichero Autor.java de la rama master para luego comprobar la diferencia con la otra rama llamada modelo mediante la visualización de Show History. En la ventana de Show History es necesario seleccionar DIFF y aparecerá una ventana como la siguiente, donde se puede observar la diferencia entre el fichero Autor.java de la rama master y la rama modelo (figura 6.46).



**Figura 6.46**  
DIFF

## Resumen

- En este capítulo final se ha explicado detalladamente la documentación relacionada con una aplicación web y el sistema de control de versiones para seguir el ciclo de vida a lo largo de su implementación a través de un equipo de trabajo.

- El capítulo comienza con la explicación de las distintas herramientas para generar documentación como Javadoc, phpDocumentor y Doxygen. Javadoc es la herramienta ideal para el lenguaje de programación de Java y se puede usar Netbeans. PhpDocumentor permite la generación de documentación en PHP. Y por último, se explica Doxygen una herramienta general que permite generar documentación en distintos lenguajes de programación, como pueden ser C++, Java, PHP, etc.
- El siguiente paso sería tener un formato estándar de documentación a la hora de documentar código, por ejemplo, sería la documentación de una clase en Java. En la cabecera de la clase se pondría el nombre de la misma, una descripción, el autor y la fecha de creación. Esto se extrapolaría a todo el código y habría una plantilla para que la documentación sea homogénea en toda la aplicación desarrollada.
- Otro concepto que se ha tratado son las herramientas colaborativas para la elaboración de documentación y mantenimiento. Una de las más usadas es Slack, que permite compartir todo tipo de documentación, controlar el workflow del proyecto, integrar aplicaciones del tipo Trello, Google Calendar, etc. Y además posee versiones de web, escritorio y móvil. Por otro lado, se tiene Office 365, pero es de pago y es usada por la mayoría del tejido productivo empresarial. Sin embargo, Slack tiene una versión gratuita que es bastante potente para controlar la documentación de una aplicación web.
- Con relación al sistema de control de versiones, se han tratado conceptos como repositorio, rama, conflicto, commit, push, etc. Se ha elegido uno de los sistemas más actuales y usados, que es Git como cliente y Github como servidor de control de versiones. Se ha explicado la instalación, configuración y uso del mismo, incluso se han explicado algunas operaciones avanzadas como Diff, crear una rama, resolver un conflicto, etc.
- La seguridad del sistema de control de versiones se realiza a partir de dos protocolos, SSH y HTTPS, incluso se controla el acceso mediante un usuario y contraseña. Existe la posibilidad de configurar una clave pública para controlar las comunicaciones mediante el cifrado de las mismas.
- Por último, se puede observar un resumen denominado historia del repositorio. Esta historia permite visualizar todas las ramas y cambios que se han realizado por parte de cualquier programador que tenga acceso al repositorio central. Además, se puede buscar por cualquier campo que permite Git, como el autor del commit, la fecha en que se realizó, etc.

## Supuestos prácticos

1. Aprovechando la instalación de Javadoc, personaliza tu proyecto con los siguientes campos en cada una de las clases y métodos:
  - Autor.
  - Fecha de la creación.
  - Descripción del objeto que se va a comentar.
  - Versión.

2. Configura phpDocumentor en Netbeans y comprueba que funciona perfectamente.
3. Configura Doxygen para que funcione correctamente, realiza un pequeño manual que te permita configurar las opciones más importantes.
4. De las herramientas colaborativas que existen en el mercado, escoge una de las más usadas. Configura y explica brevemente cómo se usaría.



## Ejercicios propuestos

1. Crea en phpDocumentor una pequeña documentación de cualquier programa que te sirva como modelo, con los campos típicos que se documentan.
2. Toma como ejemplo un proyecto completo en cualquier lenguaje de programación que permita generar la documentación en Doxygen, donde existan varias clases y métodos. Documenta cada uno de los métodos y clases que compongan la aplicación web. Muestra el fichero HTML de salida con todas sus opciones.
3. Crea un proyecto Java que posea varios ficheros .java y varios paquetes software. Una vez configurado Git con acceso a Github en sistema operativo Linux, crea el repositorio denominado App-web con un usuario y contraseña. Se deben crear al menos tres ramas y realizar modificaciones en varios ficheros de las ramas. Por último, muestra la historia de todos los commits realizados.
4. Crea un proyecto nuevo en Slack, configura todo lo relacionado con esta herramienta gratuita colaborativa. Sobre todo comparte información, crea dos cuentas mínimo para observar la compartición de recursos, como vídeos, archivos de todo tipo relacionados con la gestión de un proyecto software.

## ACTIVIDADES DE AUTOEVALUACIÓN

1. ¿Qué aspectos hay que documentar en una aplicación?:

- a) La interfaz.
- b) La implementación.
- c) La toma de decisiones.
- d) Todas las respuestas son válidas.

2. Hay herramientas para documentar de forma automática el código, ¿cómo se llama en PHP?:

- a) documentorPHP.
- b) phpDocumentor.
- c) phpCometary.
- d) phpDocumentary.

3. ¿Qué es Javadoc?:

- a) Es una herramienta para generar informes.
- b) Es una utilidad de Sun Microsystems empleado para generar documentación.
- c) Es una utilidad de JAVA para generar conexiones.
- d) Es una herramienta para generar código HTML.

4. Los comentarios de JavaDoc están destinados principalmente a:

- a) Describir API.
- b) Describir clases y métodos.
- c) Describir conexiones a bases de datos
- d) Ninguna de las respuestas es correcta.

5. ¿Cómo comienzan los comentarios en código fuente Java?:

- a) /\*\*
- b) /?
- c) /\*
- d) /+

6. Los sistemas de control de versiones son:

- a) Centralizados, online y distribuidos.
- b) Centralizados y online.
- c) Distribuidos y aislados.
- d) Centralizados y distribuidos.

7. ¿Qué es Git?:

- a) Un documentador para PHP y Java.
- b) Un sistema rápido de control de versiones.
- c) Un sistema de conexiones.
- d) Las respuestas a) y b) son correctas.

8. ¿Qué estados tiene Git?:

- a) Confirmado, modificado y preparado.
- b) Modificado y preparado.
- c) Confirmado, elaborado y preparado.
- d) Confirmado y preparado.

9. ¿Qué opción nos permite observar un listado de los commits, push en un sistema de control de versiones?:

- a) Show list.
- b) Show history.
- c) Show historic.
- d) Ninguna de las respuestas anteriores es correcta.

10. En las opciones avanzadas de Git, ¿de qué color es el fichero cuando se le añaden más líneas de código?:

- a) Rojo.
- b) Morado.
- c) Verde.
- d) Azul.

#### SOLUCIONES:

1. **a** **b** **c** **d**

2. **a** **b** **c** **d**

3. **a** **b** **c** **d**

4. **a** **b** **c** **d**

5. **a** **b** **c** **d**

6. **a** **b** **c** **d**

7. **a** **b** **c** **d**

8. **a** **b** **c** **d**

9. **a** **b** **c** **d**

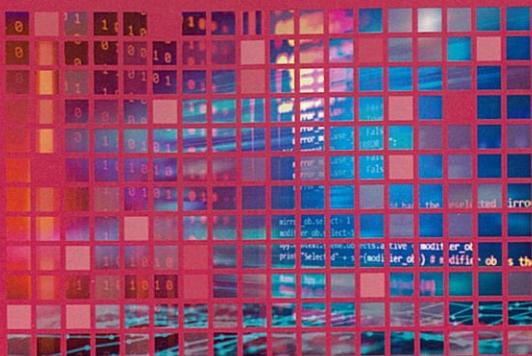
10. **a** **b** **c** **d**



# Despliegue de aplicaciones web



**INFORMÁTICA Y COMUNICACIONES**  
**G.S. Desarrollo de Aplicaciones Web**



La premisa de los contenidos que se estudian y practican en este libro es comenzar por los cimientos, como para cualquier aspecto de la vida. Por ello, se explican los servicios de red que intervienen en el despliegue de una aplicación web. Posteriormente, se detallan las aplicaciones que son vitales para desplegar la aplicación, como el FTP. Seguidamente, se hace un repaso por las arquitecturas web que existen en el mercado. Después se explica cómo administrar un servidor web y cómo desplegar una aplicación web en un servidor de aplicaciones. Finalmente, se practica con la documentación y el sistema de control de versiones.

Este libro, planteado desde dos perspectivas, la del aprendizaje en el aula y la profesional, contiene conceptos clave que de otro modo no se podrían adquirir, que permiten al lector asimilar los contenidos de forma global. Por ello, resultará de máximo interés tanto al alumnado del módulo como a los profesionales del sector e interesados en el despliegue de aplicaciones.

Esta obra incluye los siguientes recursos didácticos:

Objetivos • Mapa conceptual • Glosario • Recursos web  
Recursos didácticos (“Recuerda”, “Toma nota”, “Para saber más”)  
Actividades y ejercicios propuestos • Supuestos prácticos  
Resumen • Actividades de autoevaluación (tipo test)

ISBN: 978-84-1357-070-9

9 788413 570709