

## MÓDULO: DESPLIEGUE DE APLICACIONES WEB

Profesor: Sergio Cuesta Vicente.

<https://sites.google.com/site/sergiocuesta/>

profesor.scv@gmail.com

# Licencia



Los contenidos de este documento están compartidos bajo una [licencia Creative Commons](#) si no se indica lo contrario.

Puede conocer más sobre [Creative Commons](#) pulsando en el enlace.



El uso de este documento está sujeto a las [normas que se indican en esta página](#).

Si quiere saber más sobre Creative Commons puede visitar [su página Web](#) o la versión en [español](#).

Para saber más sobre el tipo de licencia aplicado a este documento puede visitar [esta página](#).

Todos los materiales usados para la confección de este documento están bajo licencia de Creative Commons y se hará referencia a su autor en la manera que sea posible.

Si quiere usar todo o parte de este documento, por favor, haga referencia al autor.

Sergio Cuesta Vicente

profesor.scv@gmail.com

LICENCIA.....	2
TEMA 1: IMPLANTACIÓN DE ARQUITECTURAS WEB .....	10
La arquitectura web y algunos modelos .....	11
Un modelo simple para el despliegue de aplicaciones Web.....	17
<i>Qué es una aplicación web</i> .....	18
<i>Fases de un proyecto de una aplicación web</i> .....	18
Servidores web.....	20
<i>Servicios Web</i> .....	21
<i>Alternativas</i> .....	21
<i>¿Qué necesito para montar un servidor web?</i> .....	23
Instalación y configuración básica de un servidor web: Apache.....	25
<i>Comprobación y eliminación de otras versiones</i> .....	26
<i>Instalación</i> .....	30
<i>Instalación manual de Apache 2</i> .....	31
Instalar un servidor con LAMP.....	39
Servidores de Aplicaciones.....	41
<i>Contenedores</i> .....	42
Tomcat.....	44
<i>Instalando Java</i> .....	44
<i>Instalar Apache Tomcat</i> .....	48
<i>Instalando paquetes adicionales</i> .....	51
<i>Usuarios de Tomcat</i> .....	52

Iniciar y parar Tomcat.....	53
Instalando Tomcat de forma manual.....	54
Hacer Tomcat auto arrancable.....	55
Aplicaciones web.....	58
Estructura.....	60
Descriptor de despliegue.....	61
TEMA 2: ADMINISTRACIÓN DE SERVIDORES WEB.....	63
Configuración del servidor web.....	64
Dividiendo y organizando el archivo de configuración.....	68
Directivas del archivo de configuración .....	70
Módulos.....	88
Módulos relacionados con el entorno.....	93
Módulos de autenticación y control de acceso.....	93
Módulos de generación dinámica de contenidos.....	94
Módulos de configuración del tipo de contenido.....	95
Módulos para el listado de directorios.....	96
Módulos para la gestión de las cabeceras HTTP de las respuestas.....	96
Módulos de información del servidor y de registro de la actividad.....	97
Módulos de mapeo de URLs.....	98
Otros módulos.....	99
Instalación, configuración y uso de los módulos de Apache.....	101
Los archivos de configuración de la instalación por paquete.....	101

Comprobación de los módulos ya instalados.....	102
Instalación.....	104
Uso y configuración.....	106
mod_status y mod_info.....	108
Directorios personales de usuarios.....	109
Hosts virtuales.....	111
Sitio por defecto.....	111
Modificando los mensajes de error.....	116
Alias a otros directorios.....	116
Redirecciones.....	117
Creación de hosts virtuales.....	117
Control de acceso.....	126
Control de acceso basado en la dirección.....	126
Control de acceso por variable de entorno.....	128
Control de acceso con el módulo rewrite.....	128
Autenticación y autorización.....	129
El módulo mod_auth.....	129
Los ficheros .htaccess.....	132
Agrupando usuarios para el control de acceso.....	134
El protocolo HTTPS.....	138
Certificados Digitales.....	140
Obtener un certificado digital.....	142

SSL/TSL.....	144
HTTPS en Apache.....	146
<i>Creando un sitio virtual con HTTPS.....</i>	149
Despliegue de aplicaciones sobre servidores web y Empaquetado de aplicaciones web.....	156
TEMA 3: ADMINISTRACIÓN DE SERVIDORES DE APLICACIONES.....	157
Arquitectura.....	159
<i>La estructura de directorios de Tomcat.....</i>	159
<i>Un vistazo más profundo a la arquitectura de Tomcat.....</i>	161
Configuración básica del servidor de aplicaciones.....	165
server.xml.....	165
context.xml.....	170
web.xml.....	172
Administrar aplicaciones web.....	174
<i>La estructura de archivos y directorios de una Aplicación Web.....</i>	177
Despliegue de aplicaciones en el servidor de aplicaciones .....	178
<i>Despliegue manual.....</i>	179
<i>Estableciendo nuestra aplicación como la principal para el servidor.....</i>	180
<i>Despliegue con Tomcat Web Manager.....</i>	180
Autenticación de usuarios.....	182
<i>Ejemplos de autenticación.....</i>	188
<i>Dominios de seguridad para la autenticación.....</i>	196
Administración de sesiones. Sesiones persistentes .....	209

Archivos de registro de acceso y filtro de solicitudes.....	215
Válvulas.....	216
Filtros.....	217
Configurar el servidor de aplicaciones para cooperar con servidores web.....	219
mod_jk.....	220
mod_proxy.....	227
Seguridad en el servidor de aplicaciones. Configurar el servidor de aplicaciones con soporte SSL/TSL.....	233
Prácticas finales.....	237
Práctica a entregar.....	238
TEMA 4: INSTALACIÓN Y ADMINISTRACIÓN DE SERVIDORES DE TRANSFERENCIA DE ARCHIVOS.....	239
Configuración del servicio de transferencia de archivos.....	240
Instalando el servidor.....	240
Configurando el servidor.....	242
Tipos de usuarios y accesos al servicio.....	244
Permisos y cuotas.....	264
Modos de conexión del cliente.....	268
Protocolo seguro de transferencia de archivos.....	271
Un cliente en modo texto para FTPS.....	273
Utilización de herramientas gráficas.....	275
Utilización del servicio de transferencia de archivos desde el navegador.....	277
Utilización del servicio de transferencia de archivos en el proceso de despliegue de la aplicación web.....	279
Desarrollo de operaciones remotas de gestión de contenidos: WebDAV.....	280

TEMA 5: SERVICIOS DE RED IMPLICADOS EN EL DESPLIEGUE DE UNA APLICACIÓN WEB.....	282
Resolutores de nombres.....	283
<i>Proceso de resolución de un nombre de dominio.....</i>	<i>284</i>
Parámetros de configuración y registros del servidor de nombres afectados en el despliegue.....	286
Servicio de directorios: características y funcionalidad.....	290
LDAP.....	292
Instalación de OpenLDAP.....	294
<i>Archivos básicos de configuración. Interpretación y uso.....</i>	<i>295</i>
Creando contenido en LDAP.....	302
<i>Modificando el contenido.....</i>	<i>312</i>
<i>Borrando entidades.....</i>	<i>319</i>
Clientes gráficos.....	320
Autenticación de usuarios en el servicio de directorios.....	322
<i>El atributo olcAccess.....</i>	<i>323</i>
<i>Contraseñas para los usuarios.....</i>	<i>326</i>
<i>Aplicando las directivas.....</i>	<i>329</i>
<i>Control de acceso en redes de ordenadores.....</i>	<i>340</i>
Adaptación de la configuración del servidor de directorios para el despliegue de la aplicación. Usuarios centralizados.....	342
LDAP y Apache.....	342
LDAP y Tomcat.....	347
TEMA 6: DOCUMENTACIÓN Y SISTEMAS DE CONTROL DE VERSIONES.....	354
Herramientas externas para la generación de documentación.....	355

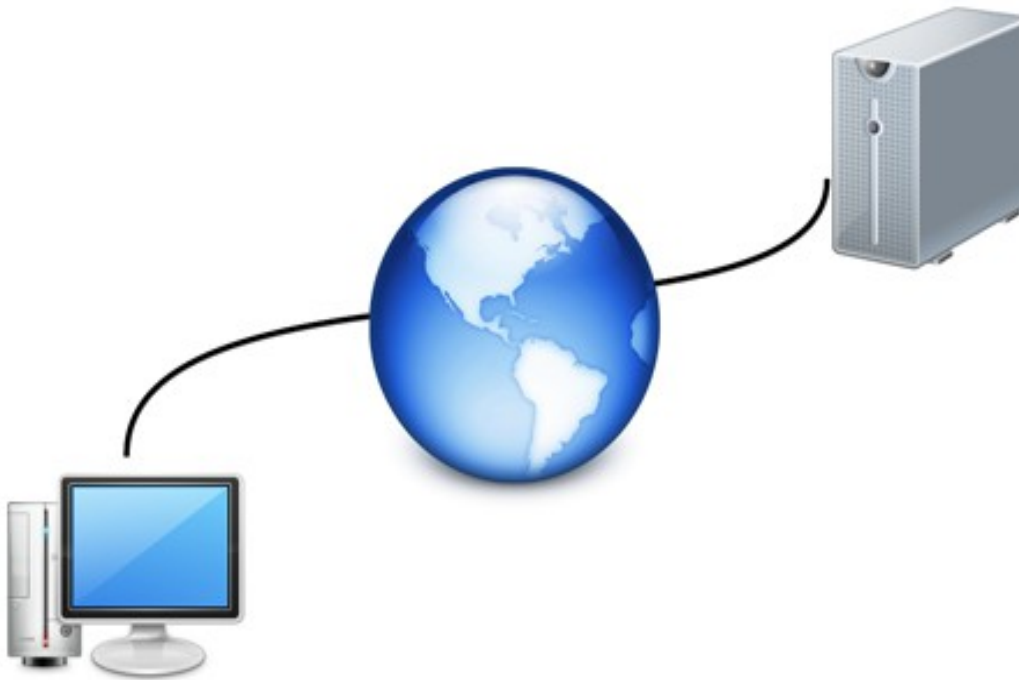


Instalación, configuración y uso.....	355
Creación y utilización de plantillas.....	356
Instalación, configuración y uso de sistemas de control de versiones. ....	357
APÉNDICES.....	358
Webmin: Una interfaz gráfica de administración.....	359
Un pequeño servidor DNS.....	361
Instalar MySQL.....	373

# Tema 1: Implantación de arquitecturas web

## La arquitectura web y algunos modelos

Una aplicación web, o web en general necesita de una estructura que permita su acceso desde diferentes lugares (máquinas). Esta estructura es lo que se denomina Arquitectura Web (realmente este nombre se da también al diseño de toda la estructura).



La gran mayoría de las arquitecturas web en la actualidad se basan en un modelo **cliente/servidor**: una comunicación asimétrica en la que uno de los extremos ofrece uno o más servicios y el otro hace uso de él. Éste es el modelo sobre el que centraremos el curso, pero no hay que olvidar otros modelos como **P2P (peer-to-peer)**, **B2B (business to business)**, etc.

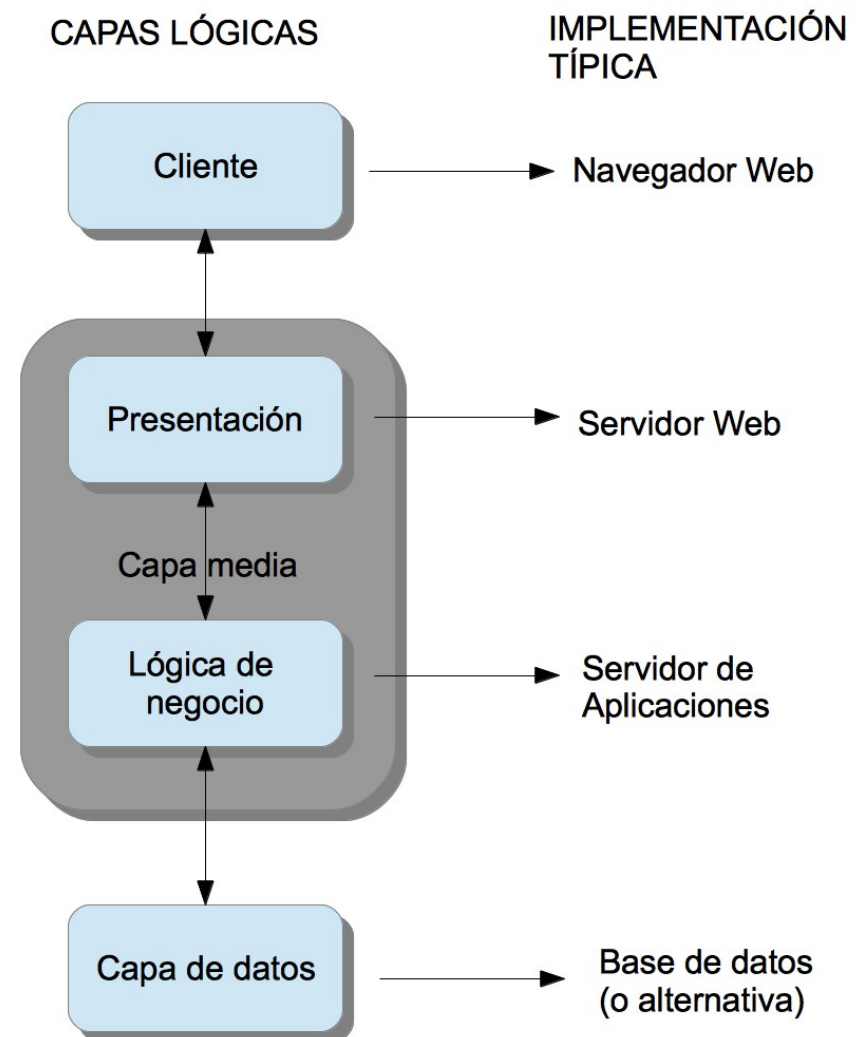
El término **servicio** es muy amplio y muchas veces confuso. Por ejemplo se puede considerar una web a la que acudimos a comprar productos un servicio en si misma, pero a la vez dicho servicio está compuesto de servicios de seguridad, de sesión, de transacciones, etcétera.

La estructura de una Arquitectura Web actual sigue el siguiente **modelo**:

1. Una **capa cliente**: es generalmente el navegador Web ejecutándose en el ordenador del usuario final. Existen otras opciones más básicas pero en la actualidad la potencia y diversidad de los navegadores existentes (así como su gratuidad) han relegado las demás opciones a la práctica desaparición.
2. Un servidor Web (**capa de negocio**): La capa cliente puede acceder a diferente lógica y procedimientos que existen en la capa de negocio. Aquí la lógica puede ser mucho más compleja que en la capa anterior. Los componentes de esta capa pueden ser desde simples archivos HTML hasta Servlets de Java. Existen muchas tecnologías que pueden usarse en este nivel: por ejemplo scripting web como PHP, ASP o JSP a lenguajes de programación como TCL, CORBA y PERL.
3. Una **capa de datos**: Se compone de un sistema de almacenamiento acceso a datos que se utilizan para confeccionar la página Web. Generalmente es un gestor de bases de datos relacionales (SGDB) pero pueden ser ficheros de texto plano, ficheros XML, etc. Una opción cada vez más usada es la creación de ficheros XML a partir de datos almacenados en una base de datos y su presentación mediante alguna de las opciones vistas el curso pasado como por ejemplo XSLT.

La capa de negocio puede estar a su vez dividida en dos partes si el sistema es suficientemente grande o complejo. Puede dividirse en una capa de presentación y una capa de lógica de negocio.

- La **capa de presentación** se encarga de componer las páginas integrando la parte dinámica en la estática. Además también procesa las páginas que envía el cliente (por ejemplo datos en formularios). Algunas soluciones para esta subcapa son los ASP de Microsoft o los JSP de Java. Esta parte la realiza generalmente un servidor web.



- La **capa de lógica de negocio** lleva a cabo operaciones más complejas. Se corresponde con la implantación de un servidor de aplicaciones. Realiza muchos tipos de operaciones entre los que destacan:
  - Realizar todas las operaciones y validaciones.
  - Gestionar el flujo de trabajo (workflow) incluyendo el control y gestión de las sesiones y los datos que se necesitan.
  - Gestionar todas las operaciones de accesos a datos desde la capa de presentación.

En el caso de estar usando páginas web estáticas (no cambian en función de diversas variables) no existiría la capa de datos ya que estos van incorporados en los propios archivos de marcas que serán las conforman las páginas web.

Este supuesto es cada vez menos común. Debido a la introducción de dinamismo en las páginas, la estructura vista anteriormente se ha visto alterada sensiblemente:

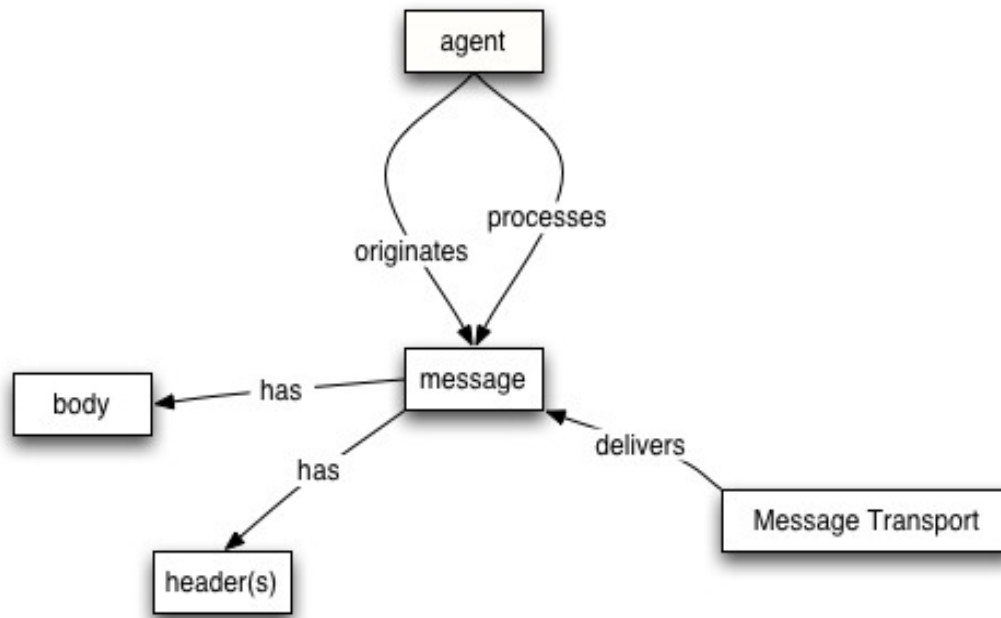
- Los navegadores Web son capaces de interpretar diferentes elementos dinámicos autónomamente o mediante plugins (javascript, flash, etc.)
- Los servidores Web también pueden interpretar código para generar las páginas web. Así se pueden introducir pequeños programas que alteren el contenido o aspecto final de una página web dependiendo de diferentes elementos como el usuario que accede o la información solicitada en cada momento. El servidor web necesita de algún módulo adicional para poder interpretar este código. Generalmente se empotra en el propio servidor web para lenguajes de script o se incorpora en un servidor a parte (de aplicaciones) para los lenguajes más potentes. Algunos lenguajes que típicamente se usan en las páginas dinámicas en el servidor son PHP, Python, Ruby o Java. Estos lenguajes también permiten el acceso a la capa de datos y la intercalación de estos datos entre los elementos de la página final.

Un ejemplo del modelo completo estaría compuesto por un servidor Apache y un Tomcat que se conecte a una base de datos. Un ejemplo del simplificado sería un servidor LAMP.

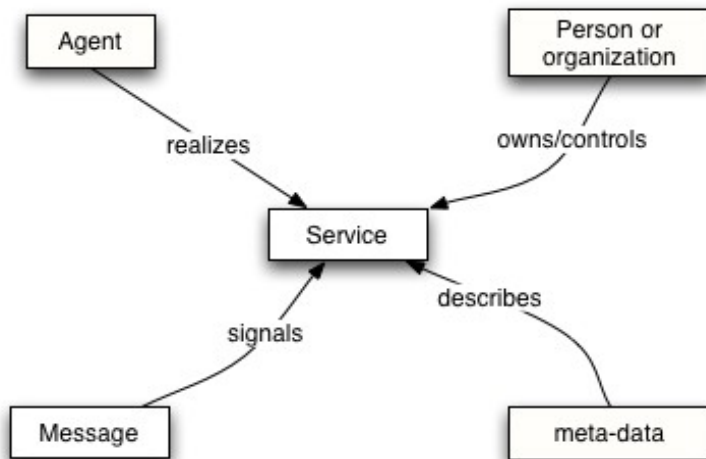
Dónde se ejecute el código de dinamización de la página determinará si el lenguaje de programación es de entorno cliente o de entorno servidor como se estudiará en los otros dos módulos de contenido informático de este curso del ciclo.

A pesar de que el modelo Cliente/Servidor es el más extendido, el W3C describe cuatro modelos de arquitectura de servicios web:

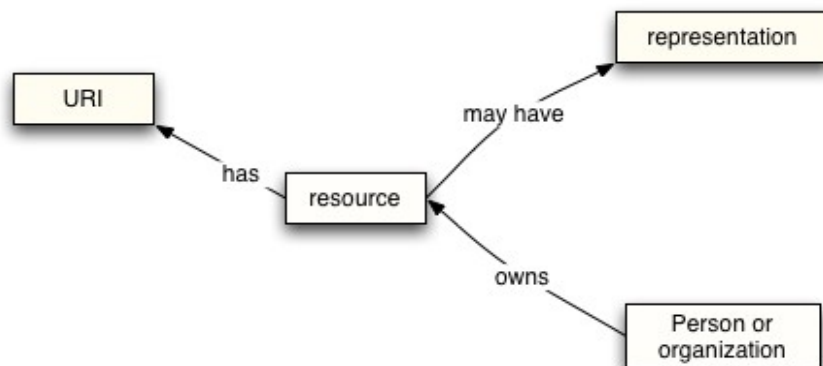
1. El [Modelo Orientado a Mensajes](#): Se centra en definir los mensajes, su estructura, la forma de transportarlos, etc. sin referencias al por qué de cada mensaje ni a su significado.



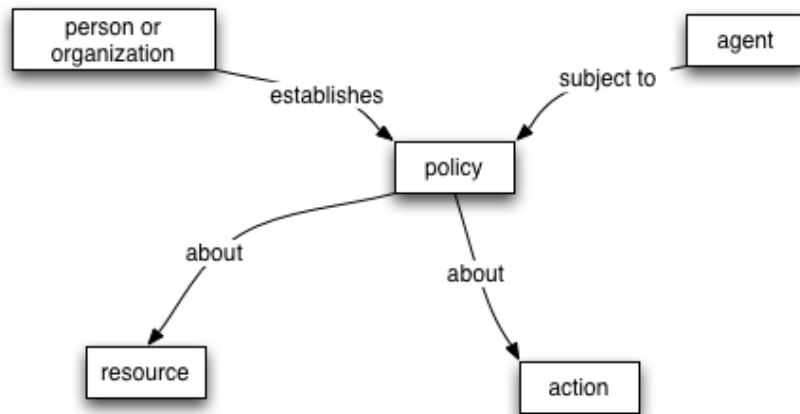
2. El [Modelo Orientado a Servicios](#): Es el más extendido y el más complejo de todos. Aunque usa mensajes, su definición no se basa en ellos si no en qué se proporciona a los receptores de dicho servicio. El servicio lo lleva a cabo un agente y lo usa otro agente, utilizando mensajes para comunicarse. En este modelo se usan metadatos para formar de y acordar muchos aspectos de la comunicación en si misma.



3. El [Modelo Orientado a Recursos](#): Este modelo se basa en la creación de recursos y su accesibilidad mediante redes.



4. El [Modelo de Políticas](#): se basa en definir los comportamientos de los agentes que utilizan los servicios definiendo cómo accederán a los recursos



Estos modelos son en muchos casos complementarios y la definición de una arquitectura web puede hacerse desde diferentes enfoques.

En un caso práctico lo más habitual es que nuestra arquitectura se base en una definición estándar.



## Un modelo simple para el despliegue de aplicaciones Web

En la actualidad la mayor parte de la información y lógica de un negocio debe ser accesible desde diferentes lugares. Aquí entran en juego las aplicaciones web.

Todos pensamos en tiendas online como modelo de una aplicación web, pero hay otros muchos como por ejemplo una aplicación de compra venta de activos entre dos bancos en el sector entre negocios (B2B).

Se puede imaginar inmediatamente que la **seguridad** es un aspecto clave en este tipo de aplicaciones, pero no el único. Muchas veces la **velocidad** y **estabilidad** de la comunicación y del servicio en sí pueden ser tanto o más críticas.

Cuando uno va a desarrollar e implantar una aplicación web debe tener en cuenta varios factores. Lo primero debe ser hacerse una **idea general de la aplicación** y de las diferentes **soluciones** que podemos utilizar. Se deben tener en cuenta las tres capas. Un error muy común es el uso de un único conjunto de tecnologías constantemente. Por supuesto conocer una tecnología es un punto a favor de su uso, pero muchas veces vamos a llevar a cabo una solución manifiestamente mejor únicamente por no haber considerado usar otras y afrontar su aprendizaje.

El siguiente aspecto a considerar sería el **coste**. Cuánto nos va a costar y qué presupuesto tenemos.

Estos factores hay que considerarlos **antes de firmar** ningún contrato e incluso antes de dar un presupuesto aunque sea orientativo.

Por ejemplo una compañía de venta de vehículos industriales quiere una aplicación web para publicar sus datos de ventas y que los comerciales puedan acceder a ella remotamente. Necesitaremos una base de datos en la que se almacenen los diferentes vehículos y sus ventas. También hará falta una lógica que mantenga todo el sistema actualizado y permita modificaciones. Además necesitaremos una capa de cliente con autenticación para que los diferentes vendedores puedan acceder al sistema, consultar y actualizar los datos.

Después de la evaluación se puede decidir no afrontar el proyecto por muchos motivos. Además de los costes ya mencionados podría darse el caso de que no tengamos los conocimientos o la infraestructura para llevar a cabo el proyecto.

Una vez vayamos a realizar el proyecto deberemos poner en práctica los conocimientos adquiridos durante el estudio de este ciclo, pero muchas veces es difícil hacerse una idea de cómo hacerlo; aquí [nos orientan con un ejemplo](#) y esta [pequeña guía](#) nos puede ser muy útil para organizar el front-end.

---

## Qué es una aplicación web

Es una aplicación que se va a ejecutar a través de internet. Constará de dos partes (al menos) una en el lado servidor y otra que se ejecutará en la máquina del cliente en un navegador web. Las aplicaciones web se engloban en el concepto superior de aplicaciones distribuidas. El servidor pone a disposición del cliente diferentes recursos. Ejemplos de aplicaciones web son el correo electrónico web, las tiendas online, las redes sociales, etc.

---

## Fases de un proyecto de una aplicación web

Se pueden considerar cuatro fases en el proyecto:

1. **Concepto:** Durante esta fase se debe obtener una idea clara y concreta de qué quiere el cliente. Además hay que obtener una idea general de cómo se llevará a cabo y de si es viable o no. Hay que determinar las limitaciones reales con que nos podemos encontrar. Por ejemplo la conexión a internet existente en la zona puede no ser suficiente para obtener los resultados previstos. Otro ejemplo de problema pueda ser que la tecnología necesaria sea demasiado cara. Es vital que al terminar esta fase se tenga una documentación que defina claramente los límites y objetivos del proyecto.
2. **Diseño:** Esta fase se centra en responder a cómo haremos la aplicación. Hay que concretar las tecnologías (tanto software como hardware) que usaremos y cómo se van a comunicar entre ellas. También hay que determinar los distintos módulos que usaremos y sus interfaces. Es muy importante realizar un plan de proyecto realista en el que se dividan las tareas y responsabilidades y se calculen los tiempos para cada elemento así como su secuencia y dependencias. También hay que obtener una especificación funcional en la que se detallen tanto el funcionamiento como el flujo de la aplicación.
3. **Desarrollo:** En esta fase se debe desarrollar el proyecto en sí. Es muy importante llevar a cabo pruebas tanto unitarias como de integración así como gestionar una documentación del desarrollo y un control de versiones.

4. **Pruebas e implantación:** Cuando el proyecto está totalmente terminado es necesario probarlo intensivamente antes de ponerlo en producción. Es necesario tener en cuenta tanto nuestra aplicación como su comunicación con otros sistemas informáticos. Cuanto más se parezca el sistema de pruebas al real mejor. El último paso es la instalación y puesta en carcha del sistema. Es un momento crítico.

Este modelo es una simplificación de los que se utilizan habitualmente en ingeniería de software.

Investiga en Internet las fases habituales en un proyecto de desarrollo de software y cuáles son los diferentes ciclos de vida que se usan. Discute en clase los descubrimientos con ayuda del profesor.

Una fase común a todos los proyectos informáticos y que no se incluye aquí es el **mantenimiento**. Este concepto incluye dos partes. El mantenimiento del servicio y corrección de errores y las mejoras. La primera consiste en asegurarse de que todo sigue funcionando y solucionar los posibles errores y “caídas” del servicio. La segunda consiste en ampliar el proyecto. Ambos casos suelen considerarse contratos a parte y por ello no se incluyen en el ciclo.

El orden correcto para el desarrollo es empezar de abajo a arriba. Es decir, primero la capa de datos, luego la de negocio y por último la presentación al cliente. Muchas veces se tiene la tentación de hacerlo al revés. Esto es debido a que no se han identificado bien las necesidades y objetivos de proyecto o no se ha realizado un diseño concreto. Es un error que nos conducirá a muchas más modificaciones y errores en nuestra aplicación.

## Servidores web

Un **servidor web** es un programa o conjunto de ellos que proporciona un servicio a través de una red. La comunicación con un servidor web suele hacerse mediante el protocolo http (hypertext transfer protocol) que está englobado en la capa de aplicación del [modelo OSI](#).

Muchas veces servidor web se usa como referencia también al **hardware** que lo aloja, pero esto es inexacto porque el mismo hardware puede albergar muchas otras funcionalidades o puede darse el caso de que un mismo hardware contenga varios servidores web (a veces simulados).

El objetivo de un servidor web es proporcionar los medios para permitir la comunicación entre dos o más programas o grupos de software sin importar la tecnología usada para crear y operar cada uno de ellos.

En la actualidad el [servidor web más extendido](#) con mucha diferencia es **Apache**. Por ello será en el que centraremos este curso. Existen muchos otros servidores web. Una forma fácil de consultar la lista y ver una [comparativa muy general](#) es visitando la Wikipedia. En [esta otra](#) se pueden leer las principales características de cada uno.

Los servidores web se engloban en un conjunto de sistemas más general que se denomina **modelo distribuido** porque el sistema no es unitario, está repartido entre diferentes máquinas o conjuntos de hardware. Este modelo tiene que afrontar algunos problemas que hay que tener siempre en cuenta:

1. La latencia y poca fiabilidad del transporte (por ejemplo la red).
2. La falta de memoria compartida entre las partes.
3. Los problemas derivados de fallos parciales.
4. La gestión del acceso concurrente a recursos remotos.
5. Problemas derivados de actualizaciones de alguna/s de las partes.

¿De qué se encarga cada nivel de la pila del modelo OSI? ¿Qué protocolos se incluyen en cada capa? Identifícalos y explica muy brevemente qué hace cada uno.

---

## Servicios Web

Un **servicio web** es un concepto abstracto que debe implementarse mediante un **agente**: un pedazo de software que envía, recibe y procesa mensajes mientras que el servicio es el concepto de qué hace. El agente solo debe ajustarse a la definición de una interfaz (dos realmente, una hacia dentro (pila OSI) y otra hacia fuera) y puede modificarse o incluso rehacerse en otro lenguaje de programación sin ningún problema. El diseño se realiza siguiendo normas de modularidad para permitir estas modificaciones.

Es de vital importancia que el servicio web esté bien definido para posibilitar la comunicación entre ambos extremos. Por ello hay muchos estándares sobre servicios web que permiten la comunicación de un cliente genérico (por ejemplo un navegador web) con diversos servicios.

Generalmente la definición de un servicio se realiza en una API que especifica cómo comunicarse con el servicio.

El proceso para usar el servicio es como sigue:

1. El cliente y el servidor deben ser conscientes de la existencia del otro. En el caso más habitual es el cliente el que informa al servidor de su intención de usar el servicio pero también puede ser el servidor el que inicie el contacto. Si es el cliente el que comienza, puede hacerlo o bien conociendo previamente cómo localizar el servidor o usando el servicio para descubrir servicios (Web Service Discovery).
2. Ambas partes deben ponerse de acuerdo sobre los parámetros que regirán la comunicación. Esto no significa que discutan, solo que las normas y protocolos deben ser las mismas en ambas partes.
3. Los agentes de ambos lados empiezan a intercambiar mensajes. El servidor web necesita componer las páginas en caso de que lleven elementos multimedia e incluso necesitará realizar otras acciones si la página se crea dinámicamente.

---

## Alternativas

Antes de decidirse a instalar nuestro propio servidor web, debemos tener en cuenta que no siempre es la mejor opción. Lo primero que debemos saber es qué quiere el cliente. Dependiendo del tamaño del servicio que vayamos a proporcionar y de la importancia de poder controlar todos los aspectos del servidor, podemos decidir usar otras posibilidades.

Por otro lado la máquina que necesitamos tendrá que tener mucha RAM y capacidad de almacenamiento a parte de soportar grandes cargas de trabajo. La conexión a internet también deberá ser potente y necesitaremos contratar una dirección IP estática.

Lo primero que se debe tener en cuenta es si nos interesa tener nuestro propio servidor web o contratar un servicio de almacenamiento web ([web hosting](#)). Realmente el término Web Hosting incluye el tener un servidor propio, pero en la actualidad se utiliza para denominar el alquilar espacio en un servidor de otra compañía. Generalmente esta compañía está dedicada a ello específicamente. Las ventajas de este caso son las obvias: no tenemos que preocuparnos de adquirir ni mantener ni el hardware ni el software necesario. Además la fiabilidad del servicio de una empresa especializada suele ser muy alta.

Existen casos en los que incluso hay tecnologías más específicas para nuestras necesidades. Cada vez es más habitual la existencia de sitios web en los que la apariencia no cambia pero el contenido es actualizado constantemente. Para estos casos se puede usar un **gestor de contenidos**. Con ellos se permite al usuario actualizar la información del sitio sin necesidad de que tenga conocimientos web concretos. Existen muchos gestores web, algunos comerciales y caros pero muy potentes. También hay dos alternativas gratuitas muy extendidas: [Joomla](#) y [Drupal](#).

Últimamente se considera incluso una alternativa más sencilla si se quiere una web informativa y bastante sencilla. [Wordpress](#) empezó siendo una plataforma para alojar blogs pero cada vez incluye más opciones y posibilidades de configuración permitiendo llevar a cabo páginas bastante atractivas.

<http://www.frameworkx.co.za/>

<http://www.webdesignerdepot.com/>

<http://designshack.net/>

Son algunos ejemplos de páginas hechas en Wordpress.

Busca comparativas entre montar tu propio servidor, contratar un hosting externo o usar wordpress. ¿Qué funcionalidades crees que son más importantes? ¿Por cuál te decantarías para crear la web de una panadería? ¿y de un centro deportivo municipal? ¿la de la consejería de deportes de la Comunidad de Madrid la realizarías con el mismo sistema?

Los gestores de contenido como Joomla, Drupal o Wordpress están generando bastantes ofertas de trabajo en la actualidad. Busca en Internet las principales características de cada uno y los motivos que te pueden llevar a decidirte por uno u otro.

## ¿Qué necesito para montar un servidor web?

Lo primero que necesitas es una **máquina** con una potencia capaz de atender las peticiones que vaya a procesar. Este punto es crítico y difícil de gestionar porque no sabemos cuál será la demanda y muchas veces es complejo estimar la carga de trabajo que soportará. Es muy recomendable que sea una máquina dedicada o que cumpla otras funciones relacionadas con intercambio de información en internet como gestionar correo electrónico o FTP.

También es vital que el **sistema operativo** que elijamos sea estable. No tiene ningún sentido elegir un sistema operativo que deje de estar funcional con facilidad. Es conveniente que lleve cierta seguridad y control de permisos integrado. Los sistemas más habituales son diferentes versiones de UNIX (por ejemplo Solaris) pero las diferentes distribuciones de Linux están tomando una posición fuerte por su bajo (o inexistente) coste. Windows también es una buena opción (sobre todo sus versiones servidor) pero es más caro que Linux y es más difícil de gestionar debido a la diversidad de configuraciones, opciones y funcionalidad que lleven integrados.

Lo siguiente que tendrás que conseguir es una **dirección IP estática**. Por supuesto debe ser una dirección de internet a no ser que tu objetivo sea montar una intranet. Nuestra máquina debe ser accesible desde redes remotas.

Consulta cómo se puede contratar una dirección IP estática y cuánto cuesta.

Los nombres y direcciones de internet que conocemos se basan en un sistema llamado DNS que lo que hace es convertir esas direcciones legibles para nosotros en direcciones IP y viceversa. Si nuestra dirección IP cambia frecuentemente cuando alguien fuera a acceder a nuestra página esta le aparecería como no disponible a pesar de que todo el resto del sistema estuviera trabajando.

¿Qué es DNS? ¿Quién lo gestiona? ¿Cómo se consigue un nombre de dominio?

Para alternativas como Wordpress o un hosting externo, ¿es necesario un nombre de dominio?

Existe la posibilidad de funcionar con una dirección IP dinámica mediante sistemas como <http://dyn.com/dns/> que mantienen siempre actualizada nuestra dirección pero solo es recomendado (incluso en la propia página) para servidores con muy poca carga de conexiones y trabajo.

La **conexión a internet 24 horas** se da por garantizada pero también implica ciertos problemas como la adquisición de dispositivos de red que aguanten ese horario sin sobrecalentarse o saturarse.

El **software** del servidor del que ya hemos hablado y sin el que no podríamos trabajar.

**Configurar** nuestra máquina para que sea accesible pero impidiendo la conexión de desconocidos a partes del sistema críticas o que no queremos publicar. En definitiva, hay que mejorar la seguridad.



## Instalación y configuración básica de un servidor web: Apache

¡Manos a la obra! Nosotros vamos a optar por instalar un servidor web Apache en un sistema operativo Linux. Es una de las opciones más extendidas y la posibilidad de obtener este software de manera gratuita disminuye mucho los costes pero no es la única razón. Apache destaca sobre otros servidores por:

- Tiene un diseño modular y altamente configurable.
- Es de código abierto por lo que existen muchas extensiones y herramientas de terceros.
- Funciona muy bien con Perl, PHP y otros lenguajes de script.
- Existen versiones para muchos sistemas operativos incluyendo Windows, Linux y Mac OS X.

La familia Apache 2 trajo muchas mejoras con respecto a la primera versión sobre todo en el plano de la flexibilidad, escalabilidad y portabilidad.

Lo más lógico sería instalar Apache en un sistema operativo de tipo servidor (Ubuntu Server) pero por motivos didácticos, vamos a instalarlo en una versión estándar con interfaz gráfica. Es menos seguro por lo que en un sistema en producción deberíamos optar por la otra opción. A pesar de usar un Linux con interfaz gráfica vamos a instalar todo desde la ventana de terminal, por lo que los pasos se podrán aplicar a un servidor.

Nosotros vamos a realizar una instalación manual para tener mayor control sobre el proceso y la configuración. Apache puede funcionar de forma autónoma, pero en la actualidad (más aún hablando de aplicaciones web) suele necesitar una base de datos y cada vez son más los sistemas que usan PHP. Por ello han proliferado los instaladores AMP (Apache, MySQL y PHP) que instalan y configuran los tres sistemas para que funcionen de manera conjunta. Esto requiere una gran configuración de los tres elementos para poder tener un servidor seguro. Para seguridad sobre PHP y MySQL habrá que referirse a los módulos propios.

[XAMPP](#) es la versión más usada para Windows y Linux. Dependiendo del sistema para el que vayan orientados, estos paquetes se llaman WAMP y LAMP genéricamente. La instalación de estos paquetes es tremendamente sencilla.

## Comprobación y eliminación de otras versiones

En la mayoría de las guías y tutoriales de internet verás que instalan primero MySQL. Esto es porque facilita la instalación, pero ¿y si tenemos que montar una base de datos en un servidor web que ya está funcionando? Nosotros instalaremos primero la opción más básica de nuestro servidor web.

Antes de nada, más aún en un servidor que vaya a publicar en la web, debemos asegurarnos de tener el control absoluto. Muchas versiones de Linux vienen con paquetes de Apache **preinstalados** así que lo primero será **eliminarlos** si existen. Esto mejora la seguridad porque nos aseguramos de estar instalando la última versión. Vamos a comprobar de manera conjunta si existen instalaciones de Apache, MySQL y PHP para poder dejar una máquina limpia.

```
dpkg --get-architecture | grep -e httpd -e apache -e mysql -e php
```

¿Qué hace la línea anterior?

En Ubuntu por defecto no viene instalado. Si está preinstalado el comando anterior devolverá algo. Si es así debemos comprobar las dependencias para eliminar todo. Si viéramos que aparece *ii* al lado del paquete estaría instalado y habría que eliminarlo. Supongamos que nos devuelve lo siguiente

ii	apache2	2.2.4-3build1	Next generation, scalable, extendable web se
ii	apache2-mpm-prefork	2.2.4-3build1	Traditional model for Apache HTTPD
ii	apache2-utils	2.2.4-3build1	utility programs for web servers
ii	apache2.2-common	2.2.4-3build1	Next generation, scalable, extendable web se

ii libapache2-mod-php5 scripting languag	5.2.3-1ubuntu6	server-side, HTML-embedded
ii libdbd-mysql-perl MySQL data	4.004-2	A Perl5 database interface to the
ii libmysqlclient15off	5.0.45-1ubuntu3	MySQL database client library
ii mysql-client-5.0	5.0.45-1ubuntu3	MySQL database client binaries
ii mysql-common	5.0.45-1ubuntu3	MySQL database common files
ii mysql-server dependin	5.0.45-1ubuntu3	MySQL database server (meta package
ii mysql-server-5.0	5.0.45-1ubuntu3	MySQL database server binaries
ii php5-common from the php	5.2.3-1ubuntu6	Common files for packages built
ii php5-mysql	5.2.3-1ubuntu6	MySQL module for php5

Buscamos los procesos de apache.

```
# ps -wef|grep apache2
```

Suponemos que nos muestra

```
root      4006      1  0 09:41 ?        00:00:00 /usr/sbin/apache2 -k start
```

```
www-data  4025  4006  0 09:41 ?          00:00:00 /usr/sbin/apache2 -k start
www-data  4026  4006  0 09:41 ?          00:00:00 /usr/sbin/apache2 -k start
www-data  4027  4006  0 09:41 ?          00:00:00 /usr/sbin/apache2 -k start
www-data  4028  4006  0 09:41 ?          00:00:00 /usr/sbin/apache2 -k start
www-data  4029  4006  0 09:41 ?          00:00:00 /usr/sbin/apache2 -k start
www-data  4140  4006  0 14:24 ?          00:00:00 /usr/sbin/apache2 -k start
www-data  4141  4006  0 14:24 ?          00:00:00 /usr/sbin/apache2 -k start
www-data  4142  4006  0 14:24 ?          00:00:00 /usr/sbin/apache2 -k start
root      4157  4125  0 14:36 pts/0    00:00:00 grep apache2
```

### Paramos apache

```
# apache2ctl stop
```

Volvemos a buscarlo para asegurarnos de que ya no se está ejecutando.

```
# ps -wef|grep apache2

root      4162  4125  0 14:36 pts/0    00:00:00 grep apache2
```

### Ahora buscamos MySQL

```
# ps -wef|grep mysql
```

Nos muestra

```
root      3857      1  0 09:41 ?           00:00:00 /bin/sh /usr/bin/mysqld_safe

mysql      3897    3857   0 09:41 ?           00:00:00 /usr/sbin/mysqld --basedir=/usr
--datadir=/var/lib/mysql --user=mysql --pid-file=/var/run/mysqld/mysqld.pid --skip-external-locking
--port=3306 --socket=/var/run/mysqld/mysqld.sock

root      3898    3857   0 09:41 ?           00:00:00 logger -p daemon.err -t mysqld_safe -i -t mysql

root      4355    4310   0 18:54 pts/1      00:00:00 grep mysql
```

Paramos MySQL

```
# /etc/init.d/mysql stop

* Stopping MySQL database server mysqld      [ OK ]
```

Y comprobamos que ya no está.

```
# ps -wef|grep mysql

root      4395    4310   0 18:55 pts/1      00:00:00 grep mysql
```

PHP no hay que pararlo.

Eliminamos todos los paquetes que aparecían.

```
# 3 remove php5-mysql libapache2-mod-php5 php5-common  
  
# apt-get remove libdbd-mysql-perl mysql-server mysql-server-5.0 mysql-client-5.0  
  
# apt-get remove libmysqlclient15off mysql-common  
  
# apt-get remove apache2 apache2-mpm-prefork apache2.2-common apache2-utils
```

Ahora que ya hemos visto cómo hacer una instalación y configuración de Apache 2 de forma manual, vamos a realizar otra instalación más estándar para continuar. Además esta instalación nos permitirá tener una versión limpia.

## Instalación

Antes de instalar algo es importante tener actualizada la lista de paquetes

```
sudo apt-get update
```

para luego actualizar la máquina entera

```
sudo apt-get upgrade
```

y luego ya instalamos

```
apt-get install apache2
```

Lo primero que debemos hacer es ver que ahora Apache está instalado en

```
cd /etc/apache2/
```

Existen diferentes versiones de Apache 2 por lo que es interesante comprobar cuál es la que estamos usando.

```
apache2 -v
```

lo que nos mostrará algo como esto

```
Server version: Apache/2.2.22 (Ubuntu)
```

```
Server built:   Feb 13 2012 01:37:45
```

Configura e instala Apache en tu máquina virtual siguiendo los pasos indicados y comprueba el acceso desde la máquina virtual y desde la máquina anfitrión. Comenta en clase los resultados.

Podemos ver que ya existe un archivo apache2 en /etc/init.d/ que hace que se inicie Apache cada vez que encendamos la máquina.

```
sudo gedit /etc/init.d/apache2
```

Nos muestra un contenido muy complejo pero al que podemos echar un vistazo sabiendo que se basa en establecer qué hacer ante diversas órdenes (las más importantes son start, stop, restart y graceful).

Echa un vistazo al script sin entrar en detalles. ¿Qué es graceful? ¿para qué se usa aquí?

## Instalación manual de Apache 2

Todo este apartado se mantiene por si es de interés para alguien pero teniendo en cuenta que el ciclo es de desarrollo y la escasez de tiempo con la que contamos he decidido pasar a la instalación automática y así ahorrar tiempo. Para saltarte esta parte de la documentación haz [click aquí](#).

Para descargar Apache lo primero es ir a su [página web](#) para comprobar cuál es la [última versión estable](#) publicada. No debes descargar las versiones en fase beta.

En el momento de crear este documento, la última versión estable es la 2.2.22

En [esta dirección](#) se encuentra la documentación oficial de Apache 2.2

La forma más fácil de instalar Apache pero que te limita mucho a la hora de elegir opciones y configurarlo es

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

**Nosotros no usaremos esta opción** porque queremos aprender a compilar el servidor. Más adelante, cuando vayamos a configurar el servidor utilizaremos una versión instalada de esta forma.

Una vez sabemos qué versión es, ya podemos descargar los paquetes. Vamos a instalar Apache desde el código fuente ya que es lo recomendado por la propia organización.

Como paso previo vamos a crear un directorio para descomprimir el código fuente después de descargarlo.

```
mkdir -p /usr/local/src/webserver
```

Una vez hemos hecho esto deberíamos comprobar que el directorio esté vacío ya que con la opción `-p` crea los padres si no existen pero no da error si el directorio ya existía.

```
cd /usr/local/src/webserver
```

```
ls -la
```

Ahora podemos proceder a descargar los archivos

```
wget http://apache.rediris.es/httpd/httpd-2.4.6.tar.gz
```



Para luego comprobar que se ha descargado correctamente debemos obtener una salida parecida a ésta, obviamente con los datos de la versión que usemos.

```
--2012-07-17 13:02:46--  http://apache.rediris.es/httpd/httpd-2.2.22.tar.gz
Resolviendo apache.rediris.es (apache.rediris.es)... 130.206.1.5
Conectando con apache.rediris.es (apache.rediris.es)[130.206.1.5]:80... conectado.
Petición HTTP enviada, esperando respuesta... 200 OK
Longitud: 7200529 (6,9M) [application/x-gzip]
Grabando a: "httpd-2.2.22.tar.gz"

100%[=====>] 7.200.529    363K/s   en 14s

2012-07-17 13:03:01 (492 KB/s) - "httpd-2.2.22.tar.gz" guardado [7200529/7200529]
```

Es muy importante comprobar que hemos hecho todo en el directorio creado para ello.

Ahora debemos descomprimir el código

```
tar xzpf httpd-2.4.6.tar.gz
```

Y comprobamos que existe el directorio.

## Instalar Apache

Hay muchas maneras de instalar Apache. Se pueden elegir módulos, configuración y rutas. Nosotros vamos a usar una que utiliza DSO (Dynamic Shared Objects). Esto es una configuración de Apache que permite añadir luego módulos sin necesidad de recompilar todo el servidor. Para nuestro curso es vital. El único pero es un descenso casi imperceptible del rendimiento por lo que considero muy recomendable usar este método.

Además vamos a activar un par de módulos. Más adelante estudiaremos para qué sirven.

Lo primero es movernos al directorio donde están los archivos con el código fuente

```
cd /usr/local/src/webserver/httpd-2.2.22
```

Ahora vamos a borrar previas compilaciones de Apache que hubiéramos hecho. Como en nuestro caso es la primera nos dará un error

```
make distclean
```

Ahora ya podemos configurar la instalación como queramos nosotros

```
./configure --prefix=/www --enable-shared=max --enable-wmodule=rewrite --enable-module=so
```

El prefijo es el directorio donde se instalará. Se puede configurar cualquiera, pero nosotros usaremos uno llamado www en la raíz. Lo demás son activaciones de módulos para poder usar DSO.

Ahora compilamos los y construimos el instalador de nuestro servidor

```
make
```

y después lo instalamos

```
make install
```

Ahora vamos a crear dos enlaces para poder usar los comandos de Apache desde cualquier lugar

```
ln -s /www/bin/apachectl /usr/bin/apachectl  
ln -s /www/bin/apachectl /usr/sbin/apachectl
```

Ya podemos arrancar Apache

```
apachectl start
```

Ahora abrimos el navegador web y cargamos la página para ver que todo ha ido bien. Escribimos <http://localhost> y debemos encontrar la página que pone “It Works!”

Podemos además colocar algún archivo HTML y comprobar que se muestra. Para ello iremos a [http://localhost/nombre\\_archivo.html](http://localhost/nombre_archivo.html)

Configura e instala Apache en tu máquina virtual siguiendo los pasos indicados y comprueba el acceso desde la máquina virtual y desde la máquina anfitrión.

### Hacer Apache auto arrancable

Lo primero que haremos será crear un archivo en la carpeta de scripts de inicio.

```
sudo gedit /etc/init.d/apache2
```

en el archivo, especificamos qué habrá que hacer en los casos de arranque, parada y reinicio del sistema.

```
#!/bin/sh  
  
case "$1" in  
  
start)
```

```
echo "Arrancando Apache ..."  
  
# Si has instalado Apache en otra ruta, pon la correcta  
  
/www/bin/apachectl start  
  
;;  
  
stop)  
  
echo "Deteniendo Apache ..."  
  
# Si has instalado Apache en otra ruta, pon la correcta  
  
/www/bin/apachectl stop  
  
;;  
  
graceful)  
  
echo "Reiniciando Apache suavemente"  
  
# Si has instalado Apache en otra ruta, pon la correcta  
  
/www/bin/apachectl graceful  
  
;;  
  
restart)
```

```
echo "Reiniciando Apache ..."  
  
# Si has instalado Apache en otra ruta, pon la correcta  
  
/www/bin/apachectl restart  
  
;;  
  
esac  
  
exit 0
```

Echa un vistazo al script y si no entiendes algo consúltalo en internet primero y luego al profesor. ¿Qué es graceful? ¿para qué se usa aquí?

Ahora es necesario que cambiemos los permisos del archivo para poder ejecutarlo

```
sudo chmod u+x /etc/init.d/apache2
```

El último paso es añadir Apache a las tareas de arranque del sistema.

```
sudo update-rc.d apache2 defaults
```

En caso de que quisiéramos hacer que dejara de arrancar automáticamente al iniciar el sistema

```
sudo update-rc.d -f apache2 remove
```

Ten en cuenta que esto no elimina el script, solo el hacerlo auto arrancable.

ahora arrancamos apache (sigue funcionando el comando de antes.

```
apachectl start
```

y podemos reiniciar la máquina.

También funcionaría ahora invocar al script para iniciar, parar, reiniciar, etc. la máquina. Pongo todas las órdenes juntas porque creo que es claro.

```
sudo /etc/init.d/apache2 start
```

```
sudo /etc/init.d/apache2 stop
```

```
sudo /etc/init.d/apache2 restart
```

```
sudo /etc/init.d/apache2 graceful
```

Haz que tu servidor con Apache arranque automáticamente al iniciar la máquina.

## Instalar un servidor con LAMP

Ahora vamos a empezar con una **nueva máquina virtual** a instalar un servidor con Apache, MySQL y PHP ya que como hemos comentado es la opción más habitual. Ni MySQL ni PHP son necesarios para montar un servidor de aplicaciones web.

Esta vez vamos a hacerlo con la opción más fácil para ver diferentes instalaciones. En el capítulo siguiente veremos las configuraciones necesarias para asegurar nuestro servidor.

Para aspectos de mayor seguridad en MySQL y PHP consulta los módulos del ciclo al respecto.

Lo primero es descargar e instalar tasksel.

```
sudo apt-get install tasksel
```

Luego instalamos directamente todo el paquete y seguimos las instrucciones.

```
sudo tasksel install lamp-server
```

Para probar Apache simplemente abrimos el navegador y consultamos localhost.

Para probar PHP, creamos un archivo prueba.php que solo contenga “<?php phpinfo(); ?>”

Para probar MySQL podemos usar cualquier método: conectarnos, instalar PHPMyAdmin, etc.

Si queremos instalar PHPMyAdmin escribimos

```
apt-get install phpmyadmin
```

y luego cargamos <http://localhost/phpmyadmin>

El directorio en el que se ha instalado Apache es

```
/etc/apache2
```

Copia una nueva máquina virtual e instala el servidor LAMP. Prueba el correcto funcionamiento de todo.



## Servidores de Aplicaciones

Un servidor de aplicaciones es un paquete software que proporciona servicios a las aplicaciones como pueden ser seguridad, servicios de datos, soporte para transacciones, balanceo de carga y gestión de sistemas distribuidos.

El término se acuñó para servidores de la plataforma Java en su versión Enterprise Edition, pero en la actualidad se extiende a muchas otras tecnologías.

Nosotros nos centraremos en Tomcat, un servidor de aplicaciones Java creado por Apache. Existen muchos otros como la integración de .NET en servidores de Microsoft, integración de PHP en un servidor para tener servidores de aplicaciones PHP, Zend Server, también para PHP, Barracuda, WebLogic de IBM, etc.

Apache Tomcat es un servidor de aplicaciones creado para alojar Servlets y Java Server Pages (JSP). Tomcat es gratuito y de código abierto pero no tiene nada que envidiar a otras soluciones comerciales. La versión que usaremos nosotros es la 7.

El funcionamiento de un servidor de aplicaciones necesita de un servidor web. Muchas veces vienen en el mismo paquete, pero realmente son dos partes diferenciadas.

Cuando un cliente hace una petición al servidor web, este trata de gestionarlo, pero hay muchos elementos con los que no sabe qué hacer. Aquí entra en juego el servidor de aplicaciones, que descarga al servidor web de la gestión de determinados tipos de archivo, en nuestro caso servlets y JSP.

Si un cliente hace una petición al servidor pidiendo un JSP, esta llega al servidor Web que lee un archivo XML que le proporciona el servidor de aplicaciones y determina que el archivo lo gestionará el servidor de aplicaciones.

En el archivo XML también se incluye la dirección del servidor de aplicaciones y el servidor web le envía la petición mediante HTTP.

## Contenedores

El término *contenedor* es otro bastante ambiguo como ya nos sucedió con *servidor*. En muchos casos se usa para referirse al propio servidor de aplicaciones e incluso al servidor web. Sin embargo, la acepción más extendida es otra.

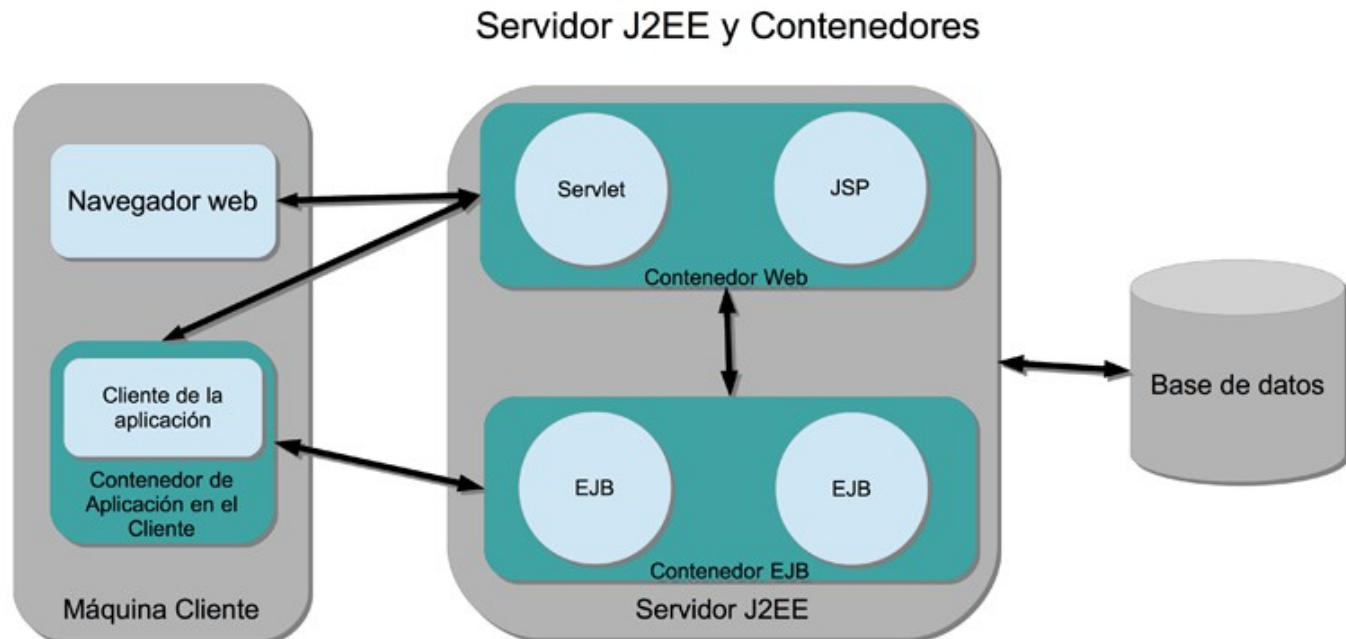
Hay que tener en cuenta que los conceptos de los que estamos hablando surgieron principalmente de J2EE.

Los contenedores en los servidores de aplicaciones son una forma de aislar la ejecución de cada aplicación o de cada

**instancia** de una aplicación del resto de instancias y de otras aplicaciones. Para cada ejecución proporcionan seguridad, soporte para transacciones, conexión remota y la gestión de los recursos precisos para la ejecución de la aplicación.

En referencia a la imagen anterior, conviene concretar algunos conceptos:

- El servidor J2EE es el programa que proporciona contenedores EJB y Web.
- El contenedor Enterprise JavaBeans (EJB) se encarga de la ejecución de los EJBs.
- El contenedor web se encarga de la ejecución de servlets y JSPs.



- El contenedor del cliente de la aplicación encarga de la ejecución de los componentes de las aplicaciones en la máquina del cliente.
- El contenedor de applets se encarga de ejecutar los applets en el cliente. Está compuesto por un navegador web y un plugin Java.

Los contenedores de Tomcat se denominan Catalina.

## Tomcat

Tomcat es un servidor de aplicaciones que puede funcionar por si mismo. De hecho es capaz de procesar peticiones en HTTP y servir archivos HTML con bastante eficiencia, pero no tan bien como lo hace Apache. Generalmente si lo que queremos es un servidor web con funcionalidad adicional lo mejor es disponer de ambos servidores trabajando conjuntamente, pero para casos donde casi todo va a ser lógica en java con Tomcat funcionando autónomamente es suficiente.

En este primer acercamiento a Tomcat lo instalaremos en una máquina virtual nueva. Más adelante veremos como integrarlo con Apache para que cada cual se encargue de hacer lo que gestiona mejor. En ese caso, Apache recibirá todas las peticiones y enviará a Tomcat lo que le corresponda a él.

En una máquina virtual de Java solo puede ejecutarse una instancia de Tomcat.

¿Qué hace la máquina virtual de Java? ¿En qué se diferencia de las máquinas virtuales que se crean con programas como Oracle Virtual Box o VMware?

## Instalando Java

Existen dos versiones de Java muy extendidas en entornos Linux. Una es la oficial de Oracle y la otra se denomina OpenJDK y es un versión de código abierto.

### OpenJDK

Esta versión es muy recomendable cuando se va a utilizar con otros sistemas de código abierto o de software libre por lo que nosotros **usaremos esta instalación.**

Actualizamos los repositorios e instalamos Java

```
sudo apt-get update  
  
sudo apt-get install openjdk-7-jdk
```

comprobamos la versión de Java

```
java -version
```

lo que debería mostrar algo como

```
java version "1.7.0_09"  
  
OpenJDK Runtime Environment (IcedTea7 2.3.3) (7u9-2.3.3-0ubuntu1~12.04.1)  
  
OpenJDK Client VM (build 23.2-b09, mixed mode, sharing)
```

Hay que establecer las variables de entorno de Java

```
sudo gedit /etc/environment
```

Y añadimos al principio las rutas de instalación de Java: el primero es el directorio donde están java y javac y el segundo el de jre.

```
JAVA_HOME=/usr/lib/jvm/java-7-openjdk-i386  
  
JRE_HOME=/usr/lib/jvm/java-7-openjdk-i386/jre
```

Y en el mismo archivo añadimos al final del path

```
:$JAVA_HOME:$JRE_HOME
```

El archivo entero debería quedar parecido a esto:

```
JAVA_HOME=/usr/lib/jvm/java-7-openjdk-i386  
JRE_HOME=/usr/lib/jvm/java-7-openjdk-i386/jre  
PATH="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:$JAVA_HOME:$JRE_HOME"
```

Aunque no es estrictamente necesario, reiniciamos la máquina virtual.

### Java de Oracle

Tomcat es un servidor de aplicaciones programadas en Java por lo que antes de instalar Tomcat, debemos tener Java instalado y funcionando.

Lo primero es añadir unos repositorios que contienen el instalador de Java

```
sudo add-apt-repository ppa:webupd8team/java
```

Actualizamos el software de los repositorios con

```
sudo apt-get update
```

Para luego instalar Java

```
sudo apt-get install oracle-java7-installer
```

Podemos comprobar que tengamos funcionando la versión correcta

```
java -version
```

Debe aparecer en pantalla algo similar a esto:

```
java version "1.7.0_05"

Java(TM) SE Runtime Environment (build 1.7.0_05-b05)

Java HotSpot(TM) Client VM (build 23.1-b03, mixed mode)
```

Por ultimo, podemos comprobar el correcto funcionamiento yendo a <http://www.java.com/es/download/installed.jsp> en el navegador web.

Ahora hay que establecer las variables de entorno para que Tomcat pueda encontrar Java. Para ello editamos el archivo *environment*

```
sudo gedit /etc/environment
```

Y añadimos al principio las rutas de instalación de Java: el primero es el directorio donde están java y javac y el segundo el de jre.

```
JAVA_HOME=/usr/lib/jvm/java-7-oracle/

JRE_HOME=/usr/lib/jvm/java-7-oracle/jre/
```

Y en el mismo archivo añadimos al final del path

```
:$JAVA_HOME:$JRE_HOME
```

El archivo entero debería quedar parecido a esto:

```
JAVA_HOME=/usr/lib/jvm/java-7-oracle/

JRE_HOME=/usr/lib/jvm/java-7-oracle/jre/
```

```
PATH="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:$JAVA_HOME:$JRE_HOME"
```

Aunque no es estrictamente necesario, reiniciamos la máquina virtual.

## Instalar Apache Tomcat

Como comentamos en la instalación manual, es necesario tener Java instalado y configurado para el correcto funcionamiento de Tomcat. Los pasos [son los mismos que ya vimos](#).

Ahora instalamos Tomcat

```
sudo apt-get install tomcat7
```

**NOTA:** En algunos casos, Tomcat no va a funcionar hasta que no configuremos las variables de entorno como se explica a continuación.

Ahora podemos abrir un navegador web y acceder a <http://localhost:8080>

Nos muestra la siguiente pantalla en la que es muy importante el siguiente párrafo:

*Tomcat7 veterans might be pleased to learn that this system instance of*



### It works !

If you're seeing this page via a web browser, it means you've setup Tomcat successfully. Congratulations!

This is the default Tomcat home page. It can be found on the local filesystem at: `/var/lib/tomcat7/webapps/ROOT/index.html`

Tomcat7 veterans might be pleased to learn that this system instance of Tomcat is installed with `CATALINA_HOME` in `/usr/share/tomcat7` and `CATALINA_BASE` in `/var/lib/tomcat7`, following the rules from `/usr/share/doc/tomcat7-common/RUNNING.txt.gz`.

You might consider installing the following packages, if you haven't already done so:

**tomcat7-docs:** This package installs a web application that allows to browse the Tomcat 7 documentation locally. Once installed, you can access it by clicking [here](#).

**tomcat7-examples:** This package installs a web application that allows to access the Tomcat 7 Servlet and JSP examples. Once installed, you can access it by clicking [here](#).

**tomcat7-admin:** This package installs two web applications that can help managing this Tomcat instance. Once installed, you can access the [manager webapp](#) and the [host-manager webapp](#).

**NOTE:** For security reasons, using the manager webapp is restricted to users with role "manager". The host-manager webapp is restricted to users with role "admin". Users are defined in `/etc/tomcat7/tomcat-users.xml`.



Tomcat is installed with CATALINA\_HOME in /usr/share/tomcat7 and CATALINA\_BASE in /var/lib/tomcat7, following the rules from /usr/share/doc/tomcat7-common/RUNNING.txt.gz.

En el que nos indican la ubicación de dos variables de entorno muy necesarias en la configuración y uso de Tomcat. En la instalación manual, ambas apuntarían al directorio en el que descomprimos Tomcat. Vamos a establecerlas como variables de entorno:

```
sudo gedit /etc/environment
```

en el añadimos (a continuación de las de Java)

```
CATALINA_HOME=/usr/share/tomcat7
```

```
CATALINA_BASE=/var/lib/tomcat7
```

Y las añadimos al path, quedaría como sigue:

```
JAVA_HOME=/usr/lib/jvm/java-7-openjdk-i386
```

```
JRE_HOME=/usr/lib/jvm/java-7-openjdk-i386/jre
```

```
CATALINA_HOME=/usr/share/tomcat7
```

```
CATALINA_BASE=/var/lib/tomcat7
```

```
PATH="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:$JAVA_HOME:$JRE_HOME:$CATALINA_HOME:$CATALINA_BASE"
```

```
LANGUAGE="es:en"
```

```
LANG="es_ES.UTF-8"
```

```
LC_NUMERIC="es_ES.UTF-8"

LC_TIME="es_ES.UTF-8"

LC_MONETARY="es_ES.UTF-8"

LC_PAPER="es_ES.UTF-8"

LC_IDENTIFICATION="es_ES.UTF-8"

LC_NAME="es_ES.UTF-8"

LC_ADDRESS="es_ES.UTF-8"

LC_TELEPHONE="es_ES.UTF-8"

LC_MEASUREMENT="es_ES.UTF-8"
```

CATALINA\_HOME indica el directorio de instalación de Tomcat.

CATALINA\_BASE indica el directorio de una instancia de Tomcat. Si tenemos más de una instancia, CATALINA\_BASE será diferente para cada una. En algunas instalaciones (aunque no en esta) ambas variables de entorno apuntan al mismo directorio.

Reiniciamos la máquina para que las variables se carguen.

El archivo que carga Tomcat por defecto es `/var/lib/tomcat7/webapps/ROOT/index.html`

Tomcat está instalado en `/etc/tomcat7`

Como podemos ver esta versión de Tomcat ya lo instala como servicio y hace que se inicia automáticamente al encender la máquina. El archivo donde está configurado esto es mucho más complejo que el que hicimos nosotros en la instalación manual; puedes consultarlo en

```
gedit /etc/init.d/tomcat7
```

## Instalando paquetes adicionales

Ya tenemos Tomcat instalado y funcionando, pero como pudimos leer en la página de inicio por defecto de Tomcat, no se han instalado ni la documentación, ni los ejemplos ni la aplicación de administración. En un servidor de producción esta puede ser la configuración correcta, pero para nuestro propósito didáctico es muy recomendable instalar los paquetes.

```
sudo apt-get install tomcat7-docs  
  
sudo apt-get install tomcat7-examples  
  
sudo apt-get install tomcat7-admin
```

Ahora podemos acceder a cada uno de ellos mediante el correspondiente enlace de la página de inicio.

Instala Tomcat y los paquetes adicionales en una máquina virtual nueva.

Ya he comentado antes cómo viene apareciendo en las noticias tecnológicas que Java es inseguro en la red y que se debe desactivar en los navegadores. Esto puede parecer el fin de Java, pero... desactiva Java en tu navegador y prueba a ejecutar los ejemplos que acabamos de instalar ¿y ahora qué?

Si pinchas en el enlace que aparece para ver el código de un ejemplo verás que contiene código pero si muestras el código de la página resultante de ejecutar el ejemplo en el navegador podrás observar que no, ¿qué indica esto?

## Usuarios de Tomcat

Lo primero es configurar los usuarios de Tomcat. Para ello debemos editar el archivo `tomcat-users.xml` que está en el directorio `conf`

```
sudo gedit /etc/tomcat7/tomcat-users.xml
```

Lo que estamos haciendo es añadir un usuario administrador así que el fichero, en la parte final debería quedar parecido a lo siguiente, con el nombre de usuario y contraseña que queramos, por supuesto.

```
<tomcat-users>

<role rolename="manager-gui"/>

<role rolename="manager-script"/>

<role rolename="manager"/>

<role rolename="admin-gui"/>

<role rolename="admin-script"/>

<role rolename="admin"/>


<user username="sergio" password="sergio" roles="manager-gui,admin-gui,manager,admin,manager-script,admin-script"/>

</tomcat-users>
```

Con este usuario y contraseña que ya hemos creado podemos cargar el manager, desde la página de inicio de Tomcat o directamente desde <http://localhost:8080/manager/html>

Se recomienda probar ejemplos de Tomcat para ver que todo vaya bien siguiendo el enlace en la página principal o en <http://localhost:8080/examples/>

---

## Iniciar y parar Tomcat

Para iniciar Tomcat usaremos

```
sudo /etc/init.d/tomcat7 start  
  
sudo /etc/init.d/tomcat7 stop  
  
sudo /etc/init.d/tomcat7 restart
```

según lo que queramos hacer.

Si quisiéramos que funcionara como servicio, sería:

```
sudo service tomcat7 start
```

y para pararlo

```
sudo service tomcat7 stop
```

## Instalando Tomcat de forma manual

Al igual que con la instalación de Apache, se omite esta opción por el poco tiempo disponible, se mantiene aquí por si alguien quiere consultarlo pero no se impartirá.

Ahora ya estamos listos para instalar Tomcat. Creamos el directorio y nos movemos a él.

```
mkdir -p /opt/tomcat  
  
cd /opt/tomcat/
```

Ahora descargamos Tomcat. Ten en cuenta que la dirección siguiente va cambiando a medida que salen nuevas versiones de Tomcat. Si no se encuentra lo mejor es en un navegador web ir a <http://apache.rediris.es/tomcat/tomcat-7/> y buscar el archivo equivalente con las nuevas versiones.

```
wget http://apache.rediris.es/tomcat/tomcat-7/v7.0.34/bin/apache-tomcat-7.0.34.tar.gz
```

y lo descomprimos

```
tar xzpf apache-tomcat-7.0.34.tar.gz -C /opt
```

A veces Tomcat no reconoce las variables de entorno de Java que añadimos en el punto anterior. Una forma de asegurarnos de que las encuentra es añadirlos en el archivo *catalina.sh* que está en el directorio de Tomcat en la carpeta bin.

```
sudo gedit /opt/apache-tomcat-7.0.34/bin/catalina.sh
```

Ya añadimos las dos variables detrás de la primera línea. El archivo quedará parecido a esto. ESTO NO ES NECESARIO, SOLO EN CASO DE QUE NO LAS LEA DE OTRA FORMA:

```
#!/bin/sh
```

```
JAVA_HOME=/usr/lib/jvm/java-7-oracle/  
  
JRE_HOME=/usr/lib/jvm/java-7-oracle/jre/  
  
#...  
  
#...  
  
#...
```

Vamos a crear los enlaces para poder arrancar y parar Tomcat con más facilidad

```
ln -s /opt/apache-tomcat-7.0.34/bin/startup.sh /usr/bin/tstartup  
  
ln -s /opt/apache-tomcat-7.0.34/bin/shutdown.sh /usr/bin/tshutdown
```

En este punto y **tras arrancar Tomcat** podemos probar si funciona yendo a <http://localhost:8080/> en el navegador Web.

Ya tendríamos instalado Tomcat, pero hay un par de cosas que debemos hacer antes de terminar todo.

---

### Hacer Tomcat auto arrancable

Lo primero que haremos será crear un archivo en la carpeta de scripts de inicio.

```
sudo gedit /etc/init.d/tomcat7
```

en el archivo, especificamos qué habrá que hacer en los casos de arranque, parada y reinicio del sistema.

```
# autoarranque de Tomcat
```

```
case $1 in
start)

sh /opt/apache-tomcat-7.0.34/bin/startup.sh

;;

stop)

sh /opt/apache-tomcat-7.0.34/bin/shutdown.sh

;;

restart)

sh /opt/apache-tomcat-7.0.34/bin/shutdown.sh

sh /opt/apache-tomcat-7.0.34/bin/startup.sh

;;

esac

exit 0
```

Ahora es necesario que cambiemos los permisos del archivo para poder ejecutarlo



```
sudo chmod 755 /etc/init.d/tomcat7
```

El último paso es crear dos enlaces simbólicos para enlazar realmente el archivo con las carpetas de inicio.

```
sudo ln -s /etc/init.d/tomcat7 /etc/rc1.d/K99tomcat
```

```
sudo ln -s /etc/init.d/tomcat7 /etc/rc2.d/S99tomcat
```

Cuando hicimos Apache auto arrancable este último paso lo hicimos de manera diferente. Esa manera es más actual y cómoda, pero he incluido la anterior como una muestra ya que no todos los sistemas UNIX y derivados incluyen la otra. De cualquier forma también podríamos usarla en este caso:

```
sudo update-rc.d tomcat7 defaults
```

Para quitar el servicio

```
sudo update-rc.d -f tomcat7 remove
```

Instala Tomcat en una máquina virtual nueva pero en la carpeta /var/tomcat. Haz que arranque automáticamente cada vez que se inicia la máquina y prueba el acceso desde dentro y desde fuera de la máquina virtual.

## Aplicaciones web

Una aplicación web se diferencia de una estándar en que se accede a ella a través de una red como Internet o una intranet por ejemplo. En muchos casos es una aplicación que se escribe en lenguajes soportados por los navegadores web (por ejemplo Javascript + HTML) y que necesita de un navegador web para ejecutarse.

El ejemplo más sencillo que se nos ocurre de una aplicación web es un programa que permita acceder a datos de una empresa desde el exterior de ésta. Aquí ya queda patente el principal problema que tienen las aplicaciones web: la **seguridad**. Muchas veces el problema es la necesidad de encontrar un compromiso entre la seguridad y la eficiencia en la aplicación; demasiada seguridad puede ralentizar el uso, aumentar el tráfico, etc.

Las principales **desventajas** de este tipo de aplicaciones son las que se derivan del uso de una red y del acceso simultáneo de varios (a veces muchos) usuarios. Las principales **ventajas** son el poder usar un navegador web como cliente (algo de lo que disponen todos los ordenadores en la actualidad) y la simplificación de las actualizaciones por no tener que actualizar los ordenadores uno a uno. Vamos a concretar un poco más:

### Ventajas:

- No es necesario ningún tipo de distribución, instalación o actualización complejo de la aplicación. Simplemente el uso de un navegador compatible nos permitirá usar la aplicación. Muchas veces se crea la aplicación para un único navegador web lo que en mi opinión es un error. No es necesario cubrir todo el espectro pero dar al menos dos o tres opciones sería recomendable.
- No se necesitan máquinas clientes especialmente potentes. Prácticamente cualquier ordenador en la actualidad es capaz de ejecutar un navegador web. Si la aplicación es más pesada en el cliente se pueden necesitar recursos un poco más altos.
- Son fáciles de integrar con otras funcionalidades de servidor como el correo electrónico.
- Generalmente permiten eliminar los problemas derivados del uso de diferentes plataformas informáticas (arquitecturas, sistemas operativos, etc.).
- El uso de HTML5 aumenta mucho la funcionalidad que puede ejecutarse nativamente en un navegador web.

### Desventajas:

- Generalmente las interfaces de usuario de las aplicaciones web son menos intuitivas y tienen un comportamiento peor que las clásicas.

- Las tecnologías web son muy dinámicas y cambiantes por lo que podemos usar alguna funcionalidad que desaparezca o se modifique drásticamente obligándonos a rehacer la interfaz (¿Flash en un futuro no muy lejano?)
- La ausencia de estándares en archivos “de oficina” puede dificultar el compartir datos e información.
- Dependen totalmente del correcto funcionamiento de la red (Internet y/o intranet).
- Desde el punto de vista de un usuario es preocupante la privacidad y seguridad de sus datos. Uno de los primeros ejemplos de esto es el correo web, pero en la actualidad todas las aplicaciones de Google y muchas otras como Facebook controlan absolutamente todo lo que haces por lo que la ausencia de privacidad es notable.

La lista de ventajas y desventajas vista es un poco ambigua. Para cada caso piensa en lo que implica la afirmación e intenta buscar un ejemplo que suponga una excepción.

Como ya hemos comentado antes una de las principales preocupaciones a la hora de desarrollar una aplicación web debe ser la seguridad. En muchos casos se deben proteger tanto **información crítica de la empresa** como los **datos privados de los usuarios**.

A la hora de implementar la seguridad de una aplicación web debemos tener en cuenta cinco áreas:

- La **autenticación** de los usuarios: el uso de un método efectivo para asegurar que se conecten los usuarios autorizados y dificultar la suplantación de identidades es algo primordial. ¿Es una aplicación abierta a todo el mundo? ¿se pueden registrar los usuarios por si mismos?
- La **autorización** de cada usuario: muchas veces deben existir diferentes tipos de usuarios y no todos los usuarios deben poder acceder a todos los datos o realizar todas las operaciones (consulta, inserción, modificación o eliminación) sobre los datos. La identificación y creación de diferentes roles es un aspecto a tener en cuenta desde las primeras etapas de diseño de la aplicación.
- La **gestión de los recursos**: es necesario proteger los datos tanto cuando se encuentran en la base de datos como cuando se encuentran en tránsito entre la máquina cliente y el servidor. Generalmente se evitan las conexiones directas de forma remota (en una red abierta) a la base de datos. Además es muy importante la encriptación de los datos tanto en la base de datos como en las comunicaciones. Los datos se utilizan como ejemplo de recurso, pero es aplicable a otros tipos.
- La **entrada de datos**: otro aspecto a tener en cuenta es qué puede escribir el usuario en cada punto de entrada de datos. Cuanto más limitemos los caracteres y tipo de entrada que puedan realizar los usuarios más fácil será impedir que la gente acceda a puntos no deseados de nuestra aplicación e incluso a otras aplicaciones de la misma red.

- **Auditorías y registros:** Para poder controlar y corregir las brechas en la seguridad es muy importante poder saber qué ha pasado. Por ello debemos tener métodos de registro (por ejemplo archivos *log*) de todo lo que suceda en nuestra aplicación. El tamaño de los archivos debe permitirnos revisar situaciones de hace bastante tiempo ya que muchas veces un agujero en la seguridad tarda en detectarse.

Aunque no conocemos cómo afrontar cada problema en detalle, ya deberíamos tener conocimientos como para plantear soluciones para las áreas delicadas que acabamos de ver. Comenta un escenario que controle los problemas generados en cada área.

Para controlar todos estos problemas hay dos **recomendaciones** básicas:

- **Pruebas:** cuanto más probemos una aplicación menos fallos tendrá. Si la aplicación no es muy pequeña, es prácticamente imposible evitar todos los agujeros de seguridad pero hay que intentar minimizarlos. Lo ideal es tener una persona o un equipo que sepan lo que hacen intentando atacar la aplicación. Este es el motivo por el que muchos hackers han terminado trabajando para compañías muy potentes.
- Usa un marco de trabajo (**Framework**): Si tenemos un equipo de trabajo no queremos que cada miembro “haga la guerra por su lado”. Todos los integrantes deben usar uno o varios métodos comunes para gestionar la seguridad de la aplicación. Un marco común mejorará enormemente la seguridad de la aplicación. Un marco de trabajo para aplicaciones web debe tener en cuenta los siguientes aspectos: persistencia de datos, gestión de sesiones y autenticación de usuarios, seguridad, uso de cachés, uso de plantillas (con tipos de datos, etc.) e incluir una interfaz de administración.

Aunque escapa al contenido del curso, en [ésta página](#) puedes consultar estos aspectos sobre los frameworks de aplicaciones web y ver una lista con muchos de ellos.

## Estructura

La estructura de una aplicación web es **equivalente a la de un servidor web** como vimos al principio del tema. La aplicación debe funcionar en todos los niveles por lo que tendremos una estructura en capas.

La más común es en tres capas (aunque la del medio puede estar subdividida como ya comentamos): presentación, aplicación y datos.

Otra forma de ver una aplicación web es en dos capas: el cliente y el servidor. Dependiendo de dónde se realiza la mayor parte del trabajo la aplicación será de cliente o de servidor pesado. En la actualidad, con la proliferación de dispositivos con poca capacidad y potencia (móviles, tabletas, netbooks, etc.) existen muchas aplicaciones pesadas en el servidor, pero cuando la aplicación requiere mucha lógica se tiende a distribuir el trabajo en los clientes para mejorar el rendimiento.

La estructura de una aplicación web también se usa para referirse a la **distribución en directorios** de todos los elementos que componen dicha aplicación. Aunque más adelante volveremos sobre este punto es importante enfatizar que la estructura de directorios que usemos debe estar contenida dentro de un punto de inicio y que no debe revelar aspectos sobre la distribución del resto de la máquina.

---

## Descriptor de despliegue

El descriptor de despliegue de una aplicación web es un archivo de configuración en el contenedor o servidor de aplicaciones. En J2EE este archivo se escribe usando sintaxis XML. Describe cómo un componente, módulo o aplicación debe desplegarse especificando aspectos como la seguridad, las opciones del contenedor y aspectos de la configuración.

Existe además un descriptor de despliegue propio de cada servidor web. Por ejemplo en Tomcat, este otro descriptor se encuentra en `<TOMCAT_HOME>/conf/web.xml`. En el caso de la instalación vista anteriormente está en `/opt/tomcat/apache-tomcat-7.0.29/conf/web.xml`.

Si estamos usando un entorno de desarrollo (por ejemplo Eclipse) el descriptor de despliegue lo crea el propio entorno, pero siempre es conveniente revisarlo para asegurarnos de que utiliza las opciones que queramos.

Para las aplicaciones web J2EE debe llamarse web.xml y ubicarse en el directorio WEB-INF en el directorio raíz de la aplicación. El esquema al que debe ajustarse el descriptor de despliegue se puede consultar en la [página correspondiente de java](#).

Podemos ver un buen ejemplo de despliegue de aplicación web [aquí](#). Ten en cuenta que si pinchas sobre una etiqueta te mostrará una breve explicación de su uso.

Se pueden consultar ejemplos más completos después de instalar Tomcat en las siguientes rutas:

- `<TOMCAT_HOME>/webapps/jsp-examples/WEB-INF/web.xml`
- `<TOMCAT_HOME>/server/webapps/manager/WEB-INF/web.xml`

Vamos a proyectar la página [vista antes](#) y a discutir sobre la estructura y diferentes elementos del descriptor de despliegue que nos ponen como ejemplo. Para ello determina previamente qué indica cada una de las etiquetas de segundo nivel (los hijos de <web-app>)

## Tema 2: Administración de servidores web

## Configuración del servidor web

En Apache, la configuración por defecto está en un archivo que se llama *httpd.conf*. Al instalar Apache se crea un directorio *conf* donde tenemos más archivos de ejemplo de configuración. Hasta el nombre y la ruta del archivo de configuración en Apache se pueden cambiar. En la instalación que hicimos se encuentra en

```
/etc/apache2/httpd.conf
```

Antes de hacer ninguna modificación sobre este archivo debemos hacer una copia de seguridad del mismo (aquí vemos el del ejemplo pero es aplicable a cualquier archivo de configuración).

```
cd /etc/apache2/conf  
  
cp httpd.conf httpd.conf.old
```

En el archivo de configuración se incluyen directivas para el servidor y comentarios. Los comentarios son las líneas que comienzan por *#*. El servidor ignora todos los comentarios y las líneas en blanco así que podemos usar ambos para mejorar la legibilidad del archivo para el administrador.

Lo mejor es echar un vistazo al archivo de configuración

```
gedit httpd.conf
```

Este archivo está vacío porque nuestra instalación no lo utiliza directamente. Es importante conocerlo porque en una instalación manual o en alguna distribución en la que no se haya modificado la configuración será el archivo a revisar. Si pruebas a buscar el archivo *httpd.conf* del servidor LAMP que instalamos verás que también está vacío.

Sin embargo en nuestra distribución, el archivo principal de configuración es *apache2.conf*



```
gedit apache2.conf
```

El archivo contiene directivas para el servidor en sí mismo: dónde estará el directorio que contendrá los documentos web (por ejemplo archivos HTML), qué módulos se cargarán, etc. También puede contener información sobre hosts virtuales, pero eso lo veremos un poco más adelante.

Lo importante ahora es ver qué ha pasado con la configuración. Si vamos al final del archivo veremos unas directivas *Include* que apuntan a otros ficheros o directorios. Esta distribución de Apache 2 utiliza una división de la configuración en diferentes archivos para mejorar la organización.

Investiga y explica el uso predeterminado de cada uno de los archivos de configuración que aparecen en *apache2.conf*, incluyendo éste mismo. No hace falta que aprendas el uso de cada uno, solo que puedas definir en una línea o dos el propósito de cada archivo.

Un gran cambio que se produjo en Apache 2 es la introducción de los **módulos multiproceso** (Multiprocessing modules o MPMs). Hasta la versión 1.3 Apache funcionaba con un sistema “prefork” en el que un proceso creaba procesos hijo para atender las peticiones y él simplemente se encargaba de monitorizarlos para crearlos o destruirlos según fuera necesario en cada momento. Este sistema no funcionaba bien en determinados sistemas operativos (por ejemplo Windows) así que a partir de la versión 2 se usó otra solución:

Cada módulo MPM crea hilos (threads) o procesos hijo (mediante prefork) para atender las peticiones. Realmente hay cuatro módulos prefork MPM, threaded MPM, perchild MPM y winnt MPM. Dependiendo de cuál usemos se usarán unas configuraciones u otras.

Una forma de saber qué módulo está usando Apache es usar la opción -V

En el servidor con Apache únicamente

```
apachectl -V
```

o

```
apache2 -V
```

nos muestra

```
Server version: Apache/2.2.22 (Unix)

Server built:   Jul 17 2012 13:50:13

Server's Module Magic Number: 20051115:30

Server loaded:  APR 1.4.5, APR-Util 1.4.1

Compiled using: APR 1.4.5, APR-Util 1.4.1

Architecture:   32-bit

Server MPM:      Prefork

    threaded:     no

        forked:   yes (variable process count)

Server compiled with....

    -D APACHE_MPM_DIR="server/mpm/prefork"

    -D APR_HAS_SENDFILE

    -D APR_HAS_MMAP

    -D APR_HAVE_IPV6 (IPv4-mapped addresses enabled)
```

```
-D APR_USE_SYSVSEM_SERIALIZE

-D APR_USE_PTHREAD_SERIALIZE

-D APR_HAS_OTHER_CHILD

-D AP_HAVE_RELIABLE_PIPED_LOGS

-D DYNAMIC_MODULE_LIMIT=128

-D HTTPD_ROOT="/www"

-D SUEXEC_BIN="/www/bin/suexec"

-D DEFAULT_PIDLOG="logs/httpd.pid"

-D DEFAULT_SCOREBOARD="logs/apache_runtime_status"

-D DEFAULT_LOCKFILE="logs/accept.lock"

-D DEFAULT_ERRORLOG="logs/error_log"

-D AP_TYPES_CONFIG_FILE="conf/mime.types"

-D SERVER_CONFIG_FILE="conf/httpd.conf"
```

Si lo probamos en el servidor LAMP nos mostraría lo mismo.

Vamos a echar un vistazo al archivo *apache2.conf* e intentar ver para qué servirá cada directiva que aparece.

---

## Dividiendo y organizando el archivo de configuración

---

Lo primero que podemos observar es que *httpd.conf* está vacío.

```
sudo gedit /etc/apache2/httpd.conf
```

Como ya he comentado en una instalación manual, la configuración por defecto irá toda en ese archivo, pero viendo lo que sucede en la instalación que estamos utilizando, podemos aprender a organizar la configuración para hacerla más manejable.

Esto es porque el archivo principal de configuración puede cambiarse cuando compilamos Apache. Por defecto es *httpd.conf* pero en la instalación LAMP que usamos lo han cambiado a *apache2.conf* y han dejado el otro por si queremos hacer modificaciones. De cualquier forma podemos editar el archivo principal.

```
sudo gedit /etc/apache2/apache2.conf
```

Al final del archivo aparece una sección de inclusiones que permiten subdividir la configuración del servidor Apache como mejor nos parezca. En este caso se muestran:

```
# Include module configuration:

Include mods-enabled/*.load

Include mods-enabled/*.conf


# Include all the user configurations:

Include httpd.conf
```

```
# Include ports listing

Include ports.conf


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


# Include generic snippets of statements

Include conf.d/


# Include the virtual host configurations:

Include sites-enabled/
```

La mayoría de ellos iremos viendo para qué sirven en sucesivas secciones. Baste aclarar que por ejemplo en *ports.conf* va información que ya conocemos y que esto puede hacerse con todo. Si nuestro servidor es pequeño quizá nos baste con usar un único fichero de configuración con todas las directivas pero en cuanto vaya a ser un poco complejo nos irá mejor si lo dividimos siguiendo algún criterio lógico.

Una curiosidad es que en el penúltimo *include* se añade un directorio. Suele usarse para crear en él archivos de configuración de elementos que no tienen otra sección lógica o para pruebas en la configuración y que así sea fácil revertir la situación al estado anterior en caso de fallo. Cuando instalemos Apache para el uso de módulos hablaremos de todos los archivos de configuración en más detalle.

## Directivas del archivo de configuración

Como ya hemos comentado antes, el archivo de configuración está compuesto por directivas que le indican al servidor cómo actuar. Ten en cuenta que cada vez que haces un cambio en el archivo de configuración es necesario reiniciar el servidor para que surta efecto. No todas las directrices que vamos a ver están en nuestro archivo de configuración. También existen otras y sobre algunas hablaremos más adelante.

En [esta dirección](#) podemos ver un ejemplo de un archivo httpd.conf más completo y con enlaces a la documentación oficial de cada directiva. Es [esta otra](#) [está la referencia](#) de las directivas de apache. Por último [aquí podemos obtener una pequeña explicación y forma de uso](#) de ellas con una indicación de para qué sirven. En esta última página en la tercera columna se indican los contextos en los que puede aplicarse cada directiva. [Aquí se concreta](#) un poco más a qué hace referencia cada contexto.

```
apachectl restart
```

Es importante destacar que las líneas que comienzan por **#** **son comentarios o están comentadas**.

### ServerRoot

Lo que viene a continuación es la configuración manual, no lo voy a cambiar por la que usamos porque es más completo así.

Ahora vamos a comentar formalmente las directivas que aparecen en el archivo de configuración.

```
ServerRoot "/www"
```

Que indica el directorio raíz de la instalación de Apache. No se refiere al directorio donde colocaremos las páginas web. Conviene recordar que al realizar la instalación manual, antes de compilar el servidor lo configuramos con la siguiente línea

```
./configure --prefix=/www --enable-shared=max --enable-wmodule=rewrite --enable-module=so
```

Si nos fijamos, el directorio coincide con el valor de `--prefix`.

Esta directiva solo se modificaría en caso de mover el servidor Apache a otra ubicación en la estructura de directorios y como iremos viendo habría que cambiar más. Lo mejor es elegir bien desde el principio dónde instalaremos Apache.

En el caso que estamos viendo, no aparece ninguna otra ruta, pero para configurar rutas relativas al directorio de instalación de Apache, una vez especificado con la directiva anterior, podríamos usar `%ServerRoot%` en lugar de la ruta completa.

### PidFile

PidFile establece la ruta al archivo en el que el servidor graba su ID de proceso (pid). Por defecto, el PID se coloca en `%ServerRoot%/logs`

```
PidFile logs/httpd.pid
```

No se recomienda cambiar la ruta si no se sabe muy bien lo que se está haciendo.

### Timeout

Son los segundos que se esperan las respuestas durante la comunicación. Por defecto es 300 segundos y se recomienda no cambiarlo.

```
Timeout 300
```

### KeepAlive, MaxKeepAliveRequests y KeepAliveTimeout

Determina si el servidor va a permitir que cada conexión haga más de una petición. El problema de activarlo es que un único cliente puede consumir demasiados recursos y saturar el servidor por lo que en caso de establecerlo a `on` se recomienda configurar cuidadosamente `KeepAliveTimeout`, generalmente a un nivel bajo.

```
KeepAlive Off
```

`MaxKeepAliveRequests` determina el número de peticiones que podrá realizar cada conexión. Evidentemente solo tiene sentido si `KeepAlive` está activada.

```
MaxKeepAliveRequests 100
```

*KeepAliveTimeout* determina el tiempo que el servidor esperará antes de atender una nueva petición del mismo cliente en la misma conexión.

```
KeepAliveTimeout 15
```

Estas directivas son un buen ejemplo de elementos que pueden hacer que nuestro servidor no funcione como esperamos una vez en producción. Si no probamos con un número de peticiones superior al máximo no podremos comprobar si el funcionamiento es el esperado.

### IfModule

Es un contenedor que permite establecer determinadas opciones solo si se ha cargado un módulo determinado. Si se escribe ! (cierre de exclamación) antes del nombre del módulo se ejecutan las opciones si no se ha cargado el módulo.

```
<IfModule mod_mime_magic.c>

    MIMEMagicFile conf/magic

</IfModule>
```

### StartServers, MaxClients y MaxRequestsPerChild

Apache crea y destruye servidores automáticamente según el tráfico que tenga que atender en cada momento. Apache es muy eficiente en esto por lo que no deberíamos preocuparnos demasiado de este y de los siguientes parámetros. Además deberían ir dentro de un *IfModule* según al que queramos aplicarlo.

Esta directriz establece cuantos servidores se crearán al arrancar.

```
StartServers 5
```

*MaxClients* establece el número máximo de cliente que podrá atender Apache a la vez.



```
MaxClients 150
```

MaxRequestsPerChild indica cuantas peticiones podrá atender cada cliente antes de ser matado.

```
MaxRequestsPerChild 30
```

Dependiendo de si usamos hilos o prefork tendremos que configurar otros elementos:

- MinSpareThreads, MaxSpareThreads, ThreadsPerChild
- MinSpareServers, MaxSpareServers

### Port

Nos permite cambiar el puerto en el que el servidor espera la peticiones estándar.

```
Port 7000
```

### Listen

Este elemento indica al servidor en qué dirección y puerto debe escuchar las peticiones http que lleguen **además de los de por defecto**. El puerto por estándar que un servidor web reciba peticiones http es el 80 por lo que la línea que nos encontramos es

```
Listen 80
```

En nuestro caso hemos instalado Apache en una máquina virtual así que está configurado para escuchar en la interfaz de loopback (127.0.0.1). Si queremos que el servidor sea visto desde la máquina host debemos configurarlo situando la dirección IP que configuramos para la tarjeta de red que añadimos como solo-anfitrión. La dirección IP y el puerto se separan por dos puntos.

```
Listen 192.168.56.101:80
```

Si configuramos Apache para escuchar en otro puerto solo podremos acceder a las páginas web añadiendo dicho puerto detrás de la dirección. Como vimos al instalar Tomcat, éste escucha por defecto en el puerto 8080 y por ello accedíamos a él con <http://localhost:8080>

### LoadModule

Cuando instalamos Apache, habilitamos la opción de Dynamic Shared Object (DSO) que permite añadir módulos dinámicamente sin necesidad de recompilar el servidor. La directiva LoadModule indica qué módulos dinámicos cargar. No es necesario incluir los módulos que se compilaron con el servidor.

En el archivo aparece comentada porque no añadimos ningún módulo. Más adelante hablaremos de los módulos.

### User y Group

Estas dos directivas indican con qué usuario y grupo se lanzarán los procesos hijos que genere Apache. Estos procesos hijo no deben lanzarse con el usuario root por razones de seguridad ya que crearían una brecha perfecta para los hackers.

Si no arrancamos el servidor con el usuario root, los procesos hijo que se lanzarán con ese mismo usuario ya que solo root puede cambiar el usuario y el grupo de un proceso por lo que estas directivas se ignorarán.

En Linux por defecto el usuario *nobody* y el grupo *nogroup* tienen muy pocos privilegios por lo que son buenos candidatos para lanzar los procesos hijo.

Si queremos usar el número del grupo o del usuario en lugar del nombre debemos añadir # justo antes del número.

```
User nobody
```

```
Group #-1
```

En nuestro archivo aparecen estas dos directivas dentro de dos IF anidados que indican que se cargarán los procesos con ese usuario y grupo si se cumplen las condiciones. Si lo ponemos fuera de los IF se usarán para todos los procesos hijo. Las condiciones en este caso son los módulos que se estén usando.

### ServerAdmin

Esta opción permite configurar la dirección del administrador del servidor web que se mostrará si el servidor genera una página de error. Evidentemente debe ser una dirección real y generalmente es del mismo dominio que el propio servidor web. Nosotros no disponemos de un nombre de dominio, pero podemos configurar una dirección de correo real.

```
ServerAdmin sergio.cuesta@domenicoscarlatti.es
```

### ServerName

Indica el nombre del servidor. Debe cumplir con las especificaciones DNS y estar en nuestro poder.

```
ServerName www.ejemplo.es:80
```

### DocumentRoot

Con esta opción indicamos el directorio raíz donde colocaremos las páginas web. Podemos crear subdirectorios dentro de éste y accederemos a los documentos que pongamos en el subdirectorio con una ruta relativa.

```
DocumentRoot "/www/docweb"
```

El problema es que no basta con cambiar esta ruta, es necesario cambiar también otra de la directiva Directory que veremos luego.

```
<Directory "/www/docweb">
```

```
...
```

Si queremos comprobar que funciona lo mejor es modificar el archivo index.html del nuevo directorio o añadir un archivo nuevo y cargar ese.

**En el caso que nos ocupa** ahora, esta directiva aparece en el archivo *default* del directorio */etc/apache2/sites-available*. Esto es así porque la instalación que usamos nosotros viene prepeorada por defecto para funcionar con sitios virtuales.

### Directory

Esta directiva también ha sido derivada al archivo de configuración del sitio por defecto, para que no se aplique a todos los sitios del servidor ya que no permite sobrescritura de las directivas de seguridad. Si se pusiera en el servidor afectaría a todos los sitios que utilizaran el directorio especificado.

Esta opción se usa para configurar cómo se comportará y qué se permitirá en cada directorio al que tiene acceso el servidor Apache. Esta configuración se aplica a un directorio y los subdirectorios que contiene si no se sobrescribe en otra definición sobre un directorio más concreto.

Nos encontramos dos veces esta etiqueta. La primera hace referencia al directorio raíz y se configura siempre con opciones muy restrictivas.

```
<Directory />

    Options FollowSymLinks

    AllowOverride None

    Order deny,allow

    Deny from all

</Directory>
```

El contenido lo vemos a continuación. Primero vamos a mostrar también la otra vez que aparece, haciendo referencia al directorio *DocumentRoot*

```
<Directory "/www/docweb">

    Options Indexes FollowSymLinks

    AllowOverride None

    Order allow,deny

    Allow from all

</Directory>
```

Aunque ambos casos estén configurados igual, esto no es necesario.

Pueden añadirse directivas para otros directorios según lo vayamos necesitando.

La primera línea permite a Apache seguir enlaces simbólicos y la segunda que las opciones de acceso de cada directorio (archivo `.htaccess` del que hablaremos en la directiva `AccessFileName`) no priman sobre éstas. La tercera línea indica si las órdenes para permitir o para denegar irán primero; en este caso se ejecutan primero las permisivas y luego las restrictivas. `Allow` (y su antónimo `Deny`) especifican quién puede acceder y quién no a un directorio determinado. El solicitante puede ser `all`, una dirección IP, un par Red/máscara de red, etc.

Puedes ver más ejemplos [aquí](#).

### UserDir

Indica si se debe permitir que cada usuario de nuestro sistema tenga su propia carpeta personal en el servidor web y establece cuál será la ruta desde el servidor para acceder a dicha carpeta. Esto tiene sentido por ejemplo si se monta un servidor Apache en un instituto y cada profesor tiene su propio usuario. Para que cada uno tenga su propia web personal se puede activar esta opción.

```
UserDir public_html
```

Es la opción más frecuente ya que crearía en nuestro servidor rutas para cada usuario. Si por ejemplo hay un usuario Sergio y otro Maria tendríamos las rutas:

## Index of /xml

- [Parent Directory](#)
- [003\\_enlaces.html](#)
- [EjemploCd.xml](#)
- [EjemploCdAtt.xml](#)
- [EjemploCdAtt.xsl](#)
- [EjemploCdIF.xml](#)
- [EjemploCdIF.xsl](#)
- [EjemploCdOtrosNodos.xsl](#)
- [csDaw.xml](#)
- [ejemploCabeceraXSDLocal.xml](#)

```
www.servidordeprueba.es/~Sergio
```

```
www.servidordeprueba.es/~Maria
```

Y cada usuario tendría en su carpeta personal un subdirectorio `public_html` para publicar lo que quisiera. Como se puede ver el argumento detrás de `UserDir` indica cómo se llamará ese subdirectorio.

```
UserDir disable
```

Es la opción por defecto y hace que los usuarios no tengan su propio espacio.

En caso de activar los directorios de usuarios se recomienda deshabilitar el de `root`

```
Userdir disabled root
```

Este tipo de configuración se ha utilizado mucho por ejemplo en universidades donde cada profesor tenía una página web disponible solo con tener un usuario. Todavía no puedes probar la configuración de esta directiva en tu servidor, pero sí leer la documentación de la directiva ¿Qué usuarios tienen ahora carpeta personal? La ruta por defecto a la carpeta personal no es muy fácil de escribir con un teclado en muchos países, ¿cómo la cambiarías? Teniendo en cuenta un servidor remoto, ¿qué inconveniente ves a esta forma de funcionar? Si tuvieras que plantear alguna funcionalidad similar en la actualidad, ¿cómo lo harías?

## DirectoryIndex

Especifica la página por defecto que se buscará al acceder a un directorio de la jerarquía de nuestro sitio. Acceder a un directorio es usar una dirección web que acaba con una barra `"/`. El directorio raíz está incluido en esa jerarquía por lo que cuando accedemos a nuestro sitio web (en nuestros casos <http://localhost>) carga el archivo `index.html` que es el valor por defecto de `DirectoryIndex`.

```
DirectoryIndex index.html
```

Puede establecerse una sucesión de archivos y el servidor mostrará la primera que encuentre del orden establecido en la directiva.

```
DirectoryIndex index.html, index.htm, inicio.html, inicio.htm
```

Si accedemos a un directorio que no contiene ninguno de los archivos especificados, Apache crea dinámicamente un archivo que lista los contenidos.

### AccessFileName

Indica el nombre del archivo en el que se deben buscar las directivas de acceso determinadas en cada directorio. Por defecto es `.htaccess` y no se recomienda cambiarlo en absoluto.

```
AccessFileName .htaccess
```

### Files

Es un contenedor que sirve para establecer directivas para tipos de archivo. Por lo menos es necesario añadir un grupo que impida el acceso a los archivos que empiezan por `.ht` por motivos de seguridad.

```
<FilesMatch "^\.ht">

    Order allow,deny

    Deny from all

    Satisfy All

</FilesMatch>
```

Otro ejemplo más sencillo es

```
<Files .htaccess>

    Order allow,deny
```

```
Deny from all

</Files>
```

Se puede usar para otros tipos de archivo según creamos conveniente.

### UseCanonicalName

Indica a Apache que cada vez que tenga que construir direcciones que hagan referencia a si mismo lo haga con el nombre canónico (el indicado en la directriz ServerName) y el puerto en lugar del nombre de la máquina (hostname) y el puerto.

```
UseCanonicalName On
```

### TypesConfig

Indica dónde estará el archivo que describe los tipos MIME (mime.types) o su equivalente si lo hemos cambiado. No hay muchas razones para ello, por lo que deberíamos dejar el valor por defecto.

```
TypesConfig conf/mime.types
```

### DefaultType

Establece el tipo MIME para todos aquellos archivos a los que no se les pueda asignar uno mediante su extensión, etc. Si tenemos un servidor donde la mayoría del contenido son páginas HTML, XML, etc es buena idea usar el valor por defecto, pero si la mayoría del contenido son archivos binarios (fotografías, programas, etc) sería conveniente establecer *application/octet-stream*.

```
DefaultType text/plain
```

Se puede especificar más:

```
DefaultType text/html
```



Investiga en Internet qué son los tipos MIME y cuáles hay. ¿Qué organismo se encarga de gestionarlos? ¿De qué otras cosas se encarga?

### HostnameLookups

Indica al servidor si debe hacer una consulta DNS para cada petición. Esto consume mucho tiempo por lo que por defecto está deshabilitada.

```
HostnameLookups off
```

### ErrorLog

Esta directiva es muy importante ya que indica dónde ubicar el archivo de registro de los errores que se produzcan en el servidor. El lugar por defecto es `%ServerRoot%/logs/error_log`. Muchos administradores crean una partición exclusivamente para situar este tipo de archivos y así tener más probabilidades de poder consultarlos en caso de un error fatal.

```
ErrorLog logs/error_log
```

### LogLevel

Establece cuánta información se guardará en el archivo de registro de errores. El nivel por defecto es suficiente para empezar pero cuanto mayor sea el servidor y más importante su función más información necesitaremos. Los valores posibles incluyen: debug, info, notice, warn, error, crit, alert, emerg.

```
LogLevel warn
```

### LogFormat

Establece qué y en qué formato se registrará. No vamos a entrar en más detalles en este curso.

### CustomLog

Establece la ruta al archivo de que registra las visitas a nuestro sitio web.

```
#CustomLog logs/access_log combined
```

Combined indica el uso de un único archivo para guardar toda la información ya que es posible dividir este registro en varios.

### ServerSignature

Indica si al mostrarse una página generada automáticamente por el servidor (no las generadas mediante lenguajes a partir de algo establecido por el usuario sino las páginas de error, listado de directorios FTP, etc) debe mostrarse el nombre y la versión del servidor. Esto puede ser usado maliciosamente así que si no estás conforme establécelo a off.

```
ServerSignature On
```

Hay otra opción (EMail) que añade la dirección del administrador de la web a la información mostrada.

### Alias

Permite crear alias para archivos o directorios. Se pueden añadir los que queramos pero el más habitual es el de la carpeta de iconos que usa Apache.

```
Alias /icons/ "/var/www/icons/"
```

```
<Directory "/var/www/icons">
```

```
Options Indexes MultiViews
```

```
AllowOverride None
```

```
Order allow,deny
```

```
Allow from all
```

```
</Directory>
```

La directiva es solo la primera línea. Lo demás es la directiva de configuración del directorio.

### ScriptAlias

Indica dónde se ubicará la carpeta de scripts CGI. El funcionamiento es similar al anterior pero solo debemos incluirlos si vamos a usar scripts CGI.

```
ScriptAlias /cgi-bin/ "/var/www/cgi-bin/"
```

```
<Directory "/var/www/cgi-bin">
```

```
    AllowOverride None
```

```
    Options None
```

```
    Order allow,deny
```

```
    Allow from all
```

```
</Directory>
```

### [IndexOptions, AddIconByEncoding, AddIconByType, AddIcon, DefaultIcon, ReadmeName, HeaderName e IndexIgnore](#)

Son directivas asociadas a la creación de índices de manera automática por el servidor. No vamos a profundizar en ellas.

### [AddEncoding](#)

Especifica un tipo particular de codificación para determinadas extensiones de archivos. También se puede usar *AddEncoding* para indicar a los navegadores que descompriman ciertos archivos mientras los descargan:

```
AddEncoding x-compress Z
```

```
AddEncoding x-gzip gz
```

### AddLanguage

Sirve para asociar extensiones a idiomas determinados para el contenido. Esta directriz es útil para la negociación de contenidos entre el servidor y el navegador web del cliente ya que Apache puede devolver contenidos en diferentes idiomas dependiendo de la configuración del idioma del navegador Web. Es útil sobre todo en los casos en los que tenemos el sitio escrito en varios idiomas ya que permitirá que Apache sirva la adecuada en cada caso en función de la configuración del navegador del cliente. Esto se realiza de forma transparente para el usuario final.

Los códigos de idiomas se definen en la especificación ISO 639. Más concretamente en la ISO 639-1 los códigos de dos letras y en la ISO 639-2 los de tres letras que incluyen más idiomas.

```
# Danés (da) - Holandés (nl) - Inglés (en) - Estonio (et)

# Francés (fr) - Alemán (de) - Griego Moderno (el)

# Italiano (it) - Noruego (no) - Coreano (kr)

# Portugués (pt) - Luxemburgués (ltz)

# Español (es) - Sueco (sv) - Checo (cz)

# Polaco (pl) - Portugués de Brasil (pt-br) - Japonés (ja)

# Ruso (ru) - Croata (hr)

#

AddLanguage es .es

AddLanguage da .dk
```

```
AddLanguage nl .nl  
AddLanguage en .en  
AddLanguage et .et  
AddLanguage fr .fr  
AddLanguage de .de  
AddLanguage el .el  
AddLanguage it .it  
AddLanguage ja .ja  
AddLanguage pl .po  
AddLanguage kr .kr  
AddLanguage pt .pt  
AddLanguage no .no  
AddLanguage pt-br .pt-br  
AddLanguage ltz .ltz  
AddLanguage sv .se
```

```
AddLanguage cz .cz  
  
AddLanguage ru .ru  
  
AddLanguage tw .tw  
  
AddLanguage hr .hr
```

Ten en cuenta que el indicador del idioma no tiene por qué ser idéntico al sufijo como se ve en varios de los ejemplos.

### LanguagePriority

Permite establecer una prioridad de los idiomas en caso de que no se especifique uno o haya un empate en la negociación por diferentes motivos. Por defecto viene en inglés, pero en la mayoría de los casos nosotros querremos establecerlo en español.

```
LanguagePriority es en fr de
```

Para poder probar este tipo de configuración es necesario tener varios archivos en diferentes idiomas con el mismo nombre. En [ésta página](#) podemos ver cómo hacerlo, pero aunque es bastante completa es antigua. En el [W3c](#) nos proporcionan una información más reciente y en la documentación oficial de Apache [aparece todo especificado](#).

### AddCharset

Es igual que *AddLanguage* pero para añadir nuevos juegos de caracteres.

```
AddCharset ISO-8859-1 .iso8859-1 .latin1  
  
AddCharset ISO-8859-2 .iso8859-2 .latin2 .cen
```

### AddDefaultCharset

Debería establecerse al juego de caracteres que mejor se ajuste a la zona en la que se sitúa el servidor y al idioma del contenido. En caso de no estar seguros es mejor dejarlo como está.

```
AddDefaultCharset ISO-8859-1
```

### BrowserMatch

Sirve para modificar la respuesta dependiendo de la configuración del cliente en cuanto a navegador y plugins. Esto suele usarse para evitar problemas con navegadores que no siguen algún estándar o evitar agujeros de seguridad. Las siguientes líneas son muy comunes porque incluyen configuraciones con problemas de sobra conocidos.

```
BrowserMatch "Mozilla/2" nokeepalive  
  
BrowserMatch "MSIE 4\.0b2;" nokeepalive downgrade-1.0 force-response-1.0  
  
BrowserMatch "RealPlayer 4\.0" force-response-1.0  
  
BrowserMatch "Java/1\.0" force-response-1.0  
  
BrowserMatch "JDK/1\.0" force-response-1.0
```

Configura tu servidor Apache para que la carpeta raíz sea /pagweb y prueba a consultarlo desde la máquina host. Solo admitirá consultar por el puerto 9090. Además la página por defecto que cargará en cada directorio debe ser inicio.html. Por último añade el idioma español, catalán, gallego y vasco en este orden. El juego de caracteres por defecto debe ser el europeo occidental. Prueba el correcto funcionamiento de todo.

Consulta los archivos de registro de errores y de acceso de tu servidor Apache.

## Módulos

El diseño de Apache es modular, algo que ya hemos dejado entrever cuando hemos hablado de Dynamic Shared Object (DSO). El núcleo de Apache incluye la funcionalidad necesaria para establecer un servidor web pero existen muchos módulos adicionales que permiten añadir funciones extra.

El uso de DSO tiene unas ventajas evidentes y la única desventaja es una pequeña disminución del rendimiento del servidor por lo que se recomienda su utilización excepto en casos en los que el rendimiento sea crítico.

Cada módulo tiene un conjunto de directivas específicas que permiten su gestión. Nosotros ahora nos centraremos en ver qué módulos hay y más adelante estudiaremos algunos casos en profundidad.

Puedes consultar los módulos que están instalados en cada servidor mediante el comando

```
apachectl -l
```

que en el caso de la instalación manual nos devuelve lo siguiente.

```
Compiled in modules:

  core.c

  mod_authn_file.c

  mod_authn_default.c

  mod_authz_host.c

  mod_authz_groupfile.c
```



```
mod_authz_user.c  
mod_authz_default.c  
mod_auth_basic.c  
mod_include.c  
mod_filter.c  
mod_log_config.c  
mod_env.c  
mod_setenvif.c  
mod_version.c  
prefork.c  
http_core.c  
mod_mime.c  
mod_status.c  
mod_autoindex.c  
mod_asis.c
```

```
mod_cgi.c
mod_negotiation.c
mod_dir.c
mod_actions.c
mod_userdir.c
mod_alias.c
mod_so.c
```

Otra forma de consultarlo es

```
apachectl -M
```

Que nos muestra los módulos por nombre y determinando si se añadieron de forma estática (cuando se compiló Apache) o de manera dinámica.

```
Loaded Modules:

core_module (static)

authn_file_module (static)

authn_default_module (static)

authz_host_module (static)
```

```
authz_groupfile_module (static)

authz_user_module (static)

authz_default_module (static)

auth_basic_module (static)

include_module (static)

filter_module (static)

log_config_module (static)

env_module (static)

setenvif_module (static)

version_module (static)

mpm_prefork_module (static)

http_module (static)

mime_module (static)

status_module (static)

autoindex_module (static)
```

```
asis_module (static)

cgi_module (static)

negotiation_module (static)

dir_module (static)

actions_module (static)

userdir_module (static)

alias_module (static)

so_module (static)

Syntax OK
```

Es fácil reconocer el mismo módulo en la terminología del primer listado y en la del segundo.

Puedes consultar todos los módulos [aquí](#). Si pinchas en el nombre de alguno te llevará a una explicación más completa y con ejemplos.

Vamos a presentar algunos de los diferentes módulos en grupos con funciones similares.

Para los siguientes módulos debes leer la pequeña explicación que viene aquí, prepararte para explicar qué hace y pensar uno o dos ejemplos de utilidad que tenga el módulo. Vamos a irlo haciendo grupo a grupo.

---

## Módulos relacionados con el entorno

Estos módulos nos permiten controlar qué parte del entorno del servidor estará disponible para otros módulos o programas. Al decir entorno nos referimos al conjunto de variables dinámicas de entorno de Apache que modifican el comportamiento del servidor.

### mod\_env

Este módulo se compila por defecto con Apache así que está disponible en nuestras instalaciones. Permite pasar el valor de variables de entorno a programas de script CGI, perl, PHP, etc.

### mod\_setenvif

También se compila por defecto por lo que está disponible en nuestras instalaciones. Este módulo nos posibilita la creación de variables de entorno a partir de datos que nos envía en cliente con el protocolo HTTP. La directiva *BrowserMatch* vista anteriormente pertenece a este módulo.

### mod\_unique\_id

Este módulo sin directivas se encarga de establecer un identificador único para cada petición que llega a la máquina con el servidor Apache en caso de que lo necesitemos. Realmente puede crear un identificador único para cada petición que llegue a un conjunto de máquinas configuradas correctamente para funcionar como un cluster.

---

## Módulos de autenticación y control de acceso

Apache implementa varios módulos para realizar autenticación y control de acceso lo que se suele aplicar al filtrado de usuarios que pueden visitar un directorio de nuestro web basándose en la dirección IP o el nombre de usuario.

El mayor problema que presentan estos módulos es que utilizan el protocolo HTTP que transmite texto plano por lo que las contraseñas se envían sin cifrar creando un agujero de seguridad muy preocupante.

Hablaremos más de varios de estos módulos cuando veamos

### mod\_auth, mod\_auth\_dbm y mod\_auth\_db

Es el módulo básico de autenticación de Apache. Usa el protocolo HTTP para ello.

mod\_auth\_dbm y mod\_auth\_db son similares pero utilizan archivos DBM y Berkley DB en lugar de archivos de texto para mejorar la eficiencia del servidor.

### mod\_auth\_anon

Este módulo permite a los usuarios conectarse de manera anónima. Para ello deben usar *anonymous* como nombre de usuario y es posible pedirles una dirección de correo como contraseña. Esta dirección puede usarse para controlar qué usuarios se conectan o para crear listas de correo de posibles clientes de nuestro servidor.

### mod\_auth\_digest

Este módulo gestiona la autenticación mediante Message Digest 5 (MD5)

### mod\_access

Permite activar la autenticación mediante dirección IP o nombre de la máquina cliente.

---

## Módulos de generación dinámica de contenidos

Estos módulos permiten delegar la atención de determinadas peticiones a diferentes scripts o programas externos.

### mod\_cgi

Sirve para ejecutar scripts de tipo CGI (Common Gateway Interface)

### mod\_include

Permite usar filtros SSI (Server-Side Includes)

### mod\_actions

Dependiendo del tipo MIME o en el método de la petición HTTP, permite usar diferentes scripts para procesar dichas peticiones.

### mod\_ext\_filter

Permite filtrar una respuesta mediante un programa externo antes de enviársela al cliente.

---

## Módulos de configuración del tipo de contenido

---

Este conjunto de módulos permiten al servidor detectar o negociar el tipo de contenido más adecuado para el cliente, entendiendo como cliente la máquina y el navegador que recibirán la respuesta HTTP.

### mod\_mime

Permite que Apache determine el tipo MIME a partir de la extensión del archivo. Se compila por defecto y nos permite activar un *Handler* para gestionar tipos de archivo. Varias de las directivas que vimos de tipos de codificación y de idiomas del contenido están relacionadas con este módulo.

### mod\_mime\_magic

Este módulo permite a Apache determinar el tipo MIME a partir de un patrón de bytes que se almacena en un archivo y se compara con la petición. Sólo se activa si el módulo anterior no es capaz de determinar el tipo MIME.

### mod\_negotiation

La negociación de contenido entre cliente y servidor típica consiste en que el cliente le envía al servidor qué tipos de contenido (idioma, codificación, etc) puede manejar y el servidor busca el más adecuado para responderle. Este módulo se compila por defecto.

---

## Módulos para el listado de directorios

---

Cuando un cliente hace una petición a nuestro servidor con una dirección web que indica un directorio el servidor intenta servir primero el archivo o archivos establecidos para ello (con la directiva *DirectoryIndex*). En caso de no encontrar ninguno en ese directorio muestra una lista de los archivos contenidos. Estos módulos nos permiten configurar cómo serán esos listados.

### mod\_dir

Módulo básico de manejo de directorios que se incluye en la compilación estándar de Apache. Por defecto realiza dos funciones básicas: añade una barra “/” al final de cada dirección que no termina en un nombre de archivo (redirige `www.misitio.es/undirectorio` a `www.misitio.es/undirectorio/`) y busca un archivo por defecto para cargar en los casos en los que se intenta acceder a un directorio. Si no definimos nada con la directiva *DirectoryIndex* se busca *index.html* por defecto.

### mod\_autoindex

Este modulo también se incluye en la configuración de Apache por defecto. En caso de que al acceder a un directorio no se encuentre ninguno de los archivos especificados *mod\_autoindex* se encarga de generar el listado. Además podemos configurar cómo se generará dicho listado.

Configurar este módulo es útil en caso de que tengamos un servidor de archivos (por ejemplo en una intranet) al que accedan muchos usuarios. No hay que confundir este tipo de acceso con el que proporciona un servidor FTP como veremos en el tema 4.

---

## Módulos para la gestión de las cabeceras HTTP de las respuestas

---

Las cabeceras en el protocolo HTTP incluyen mucha información importante para la comunicación. Éste conjunto de módulos nos permite modificar dichas cabeceras. El uso de estos módulos requiere conocimientos avanzados del funcionamiento del protocolo HTTP por lo que solo los listaremos: *mod\_asis*, *mod\_headers*, *mod\_expires* y *mod\_cern\_meta* son los módulos que se incluyen en este grupo. Solo el primero se incluye en la compilación por defecto.



---

## Módulos de información del servidor y de registro de la actividad

---

Estos módulos proporcionan información sobre el estado del servidor y permiten configurar el registro de la actividad.

### mod\_log\_config

Permite configurar el registro del acceso de usuarios al servidor.

### mod\_status

Muestra información sobre el estado del servidor.

### mod\_info

Muestra información de configuración del servidor.

### mod\_usertrack

Permite identificar usuarios y registrarlos de manera individual usando HTTP Cookies. El identificar usuarios de forma individual nos permite tratar a cada usuario de forma única o servirle información personalizada por ejemplo.

¿Qué son las cookies? Investiga en Internet y explica de forma básica cómo funcionan. Recientemente Google y Microsoft han manifestado su intención de dejar de usar cookies, ¿significa esto que están dispuestos a renunciar a la funcionalidad que proporcionan? Entérate en Internet de qué plantean.

¿Dónde se almacenan las cookies en tu ordenador? Accede a alguna para ver su contenido. La extensión Edit this Cookie the Google Chrome puede ayudarte si exportas alguna y la pegas en un documento de texto.

---

## Módulos de mapeo de URLs

---

Con este conjunto de módulos podemos manejar y modificar las URLs de nuestro sitio: a partir del nombre de dominio hacia adelante. Nos permitirá crear alias, nos ayudará a tener varios sitios web en el mismo servidor y podremos describir las direcciones para que lleven a diferentes lugares de la estructura de archivos y directorios de nuestro servidor.

### mod\_userdir

Ya comentamos al revisar el archivo de configuración de Apache cómo podíamos crear sitios personales para cada usuario usando la directiva *Userdir*. Éste módulo es el que nos permite hacerlo.

### mod\_alias

Si queremos establecer “alias” o enlaces simbólicos entre dos rutas de la estructura de archivos. Incluso permite crear redirecciones de un archivo o directorio a otro. Este módulo se compila por defecto con Apache.

### mod\_rewrite

Con este módulo podemos modificar la URL de la petición que hace el cliente para que sea una que configuremos nosotros. Para ello, se establece un patrón con el que se compara la URL y si coincide se cambia por otra según otro patrón que se establece. Esto permite entre otras muchas cosas modificar la estructura de archivos y directorios de nuestro navegador web y que las URLs de la estructura antigua sigan funcionando. Es más potente que los alias creados en el módulo anterior porque un patrón puede englobar muchas URLs (todas las que cumplan con el patrón establecido).

### mod\_speling

Este módulo corrige posibles pequeños errores en las URLs de peticiones por parte de los clientes. Tiene dos tipos de corrección: en la primera permite un error como la introducción de un carácter de más, la omisión de un carácter o el cambio de un carácter por otro (solo uno) mientras que la segunda funcionalidad busca errores provocados por el incorrecto uso de mayúsculas y minúsculas.

El módulo compara la petición con los directorios y archivos que encuentra en la estructura. Si hay una coincidencia se realiza una petición de redirección al cliente y si hay varias se le envía al cliente la lista de coincidencias.

El gran problema que tiene la activación de este módulo es el tremendo impacto negativo que puede llegar a tener en el rendimiento del servidor por lo que hay que estar muy seguros de la necesidad de activarlo.

### mod\_vhost\_alias

Este módulo está relacionado con el uso de hosts virtuales que veremos más adelante. Básicamente sirven para tener varios sitios web en el mismo servidor Apache. Sin embargo este módulo no se usa muy habitualmente porque lo que permite es la creación de hosts virtuales de forma dinámica y solo se recomienda cuando se van a crear muchísimos y la configuración manual vaya a ser demasiado lenta. Por ejemplo si un proveedor de servicios de Internet (ISP) decide crear sitios web para todos los clientes que lo soliciten.

---

## Otros módulos

En este grupo vamos a hablar de varios módulos que no encajan en ninguno de los grupos anteriores pero que merece la pena tener en cuenta.

### mod\_so

Este es el módulo que nos permite añadir otros sin la necesidad de recompilarlos. Es el módulo que nos permite el uso de DSO (Dynamic Shared Object) y todos los demás módulos de Apache se pueden usar de esta manera menos este. Recuerdo que cuando compilamos Apache añadimos este módulo para poder usar otros sin necesidad de recompilar el servidor cada vez que quisiéramos añadir o eliminar algún módulo de nuestro servidor web.

### mod\_imap

Con éste módulo que se compila por defecto con Apache, se incluye el soporte para mapas de imágenes el archivos HTML.

### mod\_proxy

Este módulo nos permite convertir Apache en un servidor proxy. Un servidor proxy se sitúa entre el cliente y el servidor y básicamente actúa como el servidor para el cliente y como un cliente para el servidor. Los motivos para usar un servidor proxy pueden ser muy variados, pero incluyen dar acceso a determinados recursos de Internet a ordenadores sin direcciones viables, hacer la función de caché para los usuarios de una red o control, registrar y/o restringir el uso de determinados recursos de Internet.

### mod\_file\_cache

Si usamos este módulo permitimos que Apache ponga en caché determinados archivos estáticos y no cambien frecuentemente. El problema con el uso de esta técnica es que si el archivo cambia puede tardar bastante en enviarse el archivo modificado a los clientes debido a la copia que tenemos en caché.

### mod\_dav

Si queremos usar WebDAV tendremos que activar este módulo. Hablaremos de WebDAV en el tema 4.

### mod\_example

Es un módulo de vital importancia para todo aquel que quiera o necesita aprender a programar un módulo de Apache, pero para nadie más. Hay que tener conocimientos del lenguaje de programación C para poder comprenderlo.

## Instalación, configuración y uso de los módulos de Apache

Después de la introducción a las funciones de algunos módulos de Apache, vamos a aprender a usar algunos de ellos. Aunque cada uno tiene sus propias peculiaridades la práctica con algunos nos llevará a comprender el funcionamiento y poder usar cualquiera consultando la documentación pertinente.

### Los archivos de configuración de la instalación por paquete

Este tipo de instalación lleva asociada una distribución de las directivas de configuración en varios archivos. Es mucho más racional y simplifica la administración del servidor Apache con respecto a la instalación manual que utiliza un único fichero de configuración aunque mediante la directiva *Include* se puede configurar al gusto de cada uno.

#### apache2.conf

El fichero de configuración principal en este tipo de instalación es `apache2.conf` y se encuentra en `/etc/apache2/apache2.conf` e incluye las directivas principales del servidor. Al final del archivo y mediante órdenes *include* se le indica a Apache que debe cargar otros archivos donde se encuentra la configuración de otros componentes.

Debemos echar un vistazo al archivo de configuración de Apache

```
sudo gedit /etc/apache2/apache2.conf
```

Es muy parecido al de la instalación de un servidor LAMP.

#### ports.conf

En este fichero se establecen las direcciones IP y los puertos que va a usar el servidor. En la instalación que hemos llevado a cabo se establece que se escuche en el puerto 80 (http por defecto) y en caso de que activemos el protocolo SSL se usaría el puerto 443.

### envvars

Se utiliza para configurar las variables de entorno del servidor.

### httpd.conf

Debemos recordar que este es el fichero de configuración principal en las instalaciones de apache manuales. Aquí se mantiene por compatibilidad y se recomienda que el administrador del servidor introduzca en él las configuraciones adicionales en lugar de incluirlas en apache2.conf.

### /etc/apache2/conf.d/

En este directorio se incluyen configuraciones adicionales y algunas aplicaciones web ubican aquí sus archivos de configuración. También incluye algunos archivos de configuración en nuestra instalación por defecto así que vamos a  **echar un vistazo a los existentes**.

---

## Comprobación de los módulos ya instalados

Para comprobar qué módulos están instalados por defecto

```
apachectl -M
```

Vemos lo siguiente

```
Loaded Modules:

  core_module (static)

  log_config_module (static)

  logio_module (static)

  mpm_worker_module (static)
```

```
http_module (static)

so_module (static)

alias_module (shared)

auth_basic_module (shared)

authn_file_module (shared)

authz_default_module (shared)

authz_groupfile_module (shared)

authz_host_module (shared)

authz_user_module (shared)

autoindex_module (shared)

cgid_module (shared)

deflate_module (shared)

dir_module (shared)

env_module (shared)

mime_module (shared)
```

```
negotiation_module (shared)

reqltimeout_module (shared)

setenvif_module (shared)

status_module (shared)

Syntax OK
```

Por lo que al llevar activo el `so_module` quiere decir que soporta DSO así que se pueden activar los módulos de manera dinámica.

Nosotros podemos encontrar los módulos en la carpeta `/usr/lib/apache2/modules`

```
cd /usr/lib/apache2/modules

ls -la
```

Existen muchos módulos adicionales que no se incluyen en la instalación estándar de Apache. Para consultar estos módulos debemos ejecutar la orden

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

---

## Instalación

Para instalar un módulo de Apache hay que usar la directiva [LoadModule](#) del módulo [mod\\_so](#). Cuya sintaxis viene definida como

```
LoadModule module filename
```

Donde *module* es el nombre del módulo y *filename* la ruta y nombre del archivo `.so` dónde se encuentra.



Por ejemplo para activar el módulo *mod\_speling*

```
LoadModule speling_module /usr/lib/apache2/modules/mod_speling.so
```

Siendo necesario después reiniciar Apache

```
apachectl restart
```

### Alternativa

En algunas distribuciones de Linux hay otra forma de activar los módulos. Si nos damos cuenta tenemos dos directorios */etc/apache2/mods-available* y */etc/apache2/mods-enabled*. En el primero están todos los módulos que vienen incluidos en esta versión de Apache mientras que en el segundo están los que tenemos activos en nuestro servidor mediante enlaces simbólicos a los módulos en el directorio anterior.

Para activar un módulo usamos el comando *a2enmod* con el nombre del módulo.

```
sudo a2enmod speling
```

que nos informa de lo que está sucediendo y de que es necesario reiniciar el servidor

```
Enabling module speling.
```

```
To activate the new configuration, you need to run:
```

```
service apache2 restart
```

y para desactivarlo se hace igual pero con el comando *a2dismod*.

```
sudo a2dismod speling
```

para lo que obtenemos una información equivalente.

```
Module spelling disabled.
```

To activate the new configuration, you need to run:

```
service apache2 restart
```

después de cada activación o desactivación es necesario reiniciar el servidor.

```
apachectl restart
```

ten en cuenta que la línea anterior es equivalente a

```
apache2 restart
```

---

## Uso y configuración

Cada módulo en Apache 2 tiene un uso y configuración diferentes. Para saber cómo usar cada módulo lo mejor es consultar la documentación de la página oficial. Desde la [lista de módulos](#) se puede hacer click en cada uno para ver sus directivas y uso.

Nosotros vamos a seguir con el ejemplo del módulo [mod\\_speling](#). Podemos ver que solo tiene dos directivas que podemos usar: *CheckSpelling* y *CheckCaseOnly*. La primera activa o desactiva el módulo y la segunda hace que solo se corrijan los errores de mayúsculas/minúsculas. Para activarlo la corrección completa debemos añadir al archivo de configuración la siguiente línea:

```
CheckSpelling on
```

Después de cada cambio en la configuración es necesario reiniciar la máquina para que la modificación surta efecto.

```
apache2 restart
```

Para probar si todo funciona correctamente podemos intentar cargar la página web <http://localhost/insdex.html> antes de los cambios en el servidor. Nos devuelve una página de error generada automáticamente con Apache con el siguiente texto:

```
Not Found
```

```
The requested URL /insdex.html was not found on this server.
```

```
Apache/2.2.22 (Ubuntu) Server at localhost Port 80
```

Si después de los cambios intentamos acceder a la misma página web veremos que corrige la dirección a <http://localhost/index.html> y carga la página de inicio de Apache.

Prueba a activar el módulo que corrige la sintaxis modificando el archivo `httpd.conf` (no el `apache2.conf` ya que así es más fácil comprobar lo que hemos hecho). Prueba que todo funciona correctamente y borra las modificaciones en la configuración.

Ten en cuenta la directiva `IfModule`

Devuelve la configuración al estado anterior a este ejercicio.

### Archivos de configuración asociados a cada módulo

En el directorio `/etc/apache2/mods-enabled` podemos ver que hay dos archivos asociados a cada módulo activo. Los archivos `.load` incluyen la directiva para que se cargue el módulo en cuestión mientras que los archivos `.conf` incluyen directivas de configuración que únicamente se aplicarán si se carga un módulo determinado. Esto se consigue con la directiva `IfModule` que funciona como un `if` de programación.

Realmente lo que encontramos en el directorio son enlaces simbólicos a los archivos en `/etc/apache2/mods-available`

## *mod\_status y mod\_info*

Estos dos módulos nos permiten obtener información muy útil sobre nuestro servidor.

`mod_status` está activado por defecto en nuestra distribución de Apache, pero se activa y desactiva como cualquier otro módulo. Podemos ver y modificar su configuración en `/etc/apache2/mods-available/`

```
gedit status.conf
```

Vamos a aprovechar para ver otra directiva de configuración. Si nos fijamos ahora solo nos permite consultar el estado desde nuestra máquina.

```
Allow from 127.0.0.1 ::1
```

Podemos añadir una línea debajo con la dirección IP de otra máquina para que nos dé acceso desde ella.

Se consulta dicha información en la dirección o nombre de nuestro servidor con: <http://192.168.56.101/server-status> si nuestra dirección fuera esa.

`mod_info` no viene activado por defecto

```
sudo a2enmod info
```

podemos consultar o modificar su configuración

```
gedit /etc/apache2/mods-enabled/info.conf
```

Esta información se puede consultar en <http://192.168.56.101/server-info> modificando la dirección IP por la tuya o el nombre de tu servidor.

Recuerda que para poder usar los cambios en la configuración hay que reiniciar el servidor.

Consulta los dos archivos de configuración asociados al módulo status. Estudia para qué se utilizan las directivas que se ejecutan en caso de que se cargue el módulo y coméntalo en clase.

Habilita ambos módulos y permite que se consulten desde la máquina anfitrión.

Prueba el módulo mod\_status. Comprueba que funciona y activa la opción que muestra el estado extendido.

Activa el módulo mod\_info y comprueba que funciona. Añade a la información estándar información sobre el módulo mod\_status.

Prueba a consultar ambas páginas de información y echa un vistazo a qué información se puede obtener.

## Directorios personales de usuarios

En determinadas circunstancias (por ejemplo una institución educativa como un instituto o una universidad) es útil que cada usuario tenga un directorio en el que pueda crear su propio conjunto de páginas web. Si el número de usuarios es grande, esto puede cargar innecesariamente de trabajo al administrador del servidor. Apache ofrece una alternativa automatizada para esta situación mediante el módulo [mod\\_userdir](#).

Si activamos el módulo (por defecto no viene activo) cada usuario tendrá un espacio al que se accederá mediante la URL <http://sitioejemplo.com/~nombreusuario> donde nombreusuario será el nombre que tiene en el servidor Linux.

Cada usuario puede crear ahora en su carpeta personal un directorio `public_html` donde ubicar sus páginas. [Aquí se explica](#) en más detalle cómo se utiliza la carpeta personal.

No se activa esta opción para el usuario root.

En la documentación del módulo puedes observar que el directorio en el que cada usuario podrá crear sus páginas es altamente configurable.

Activa el módulo `mod_userdir` y crea una carpeta para el usuario con el que instalaste Linux. Dentro de la carpeta crea un archivo `prueba.html` y configura el módulo para que al acceder a la carpeta personal del usuario desde un navegador web se muestre dicho archivo por defecto.

Por último desactiva el módulo.

---

## Hosts virtuales

---

Apache puede servir varios sitios web desde un único servidor web. El cliente nunca diferenciará si son sitios en servidores diferentes o en la misma máquina. Apache es un servidor muy potente para la utilización de esta opción.

Si por ejemplo somos los gestores de dos dominios DNS ([www.laempresa.es](http://www.laempresa.es) y [www.mipagina.es](http://www.mipagina.es)) podemos alojar ambos sitios en el mismo servidor Apache. Uno de estos dominios se considerará el sitio principal y todos los demás serán los hosts virtuales.

Ventajas:

1. Aprovechar el hardware existente.
2. Aprovechar las direcciones IP públicas disponibles.

Una gran ventaja en el uso de hosts virtuales en Apache es que **permite heredar** la configuración del sitio principal por lo que no habrá que reconfigurar todas las directivas, solo las que cambien.

Al igual que pasaba con los módulos, existen dos directorios para contener los sitios virtuales, uno para los disponibles y otro para los activos: */etc/apache2/sites-available* y */etc/apache2/sites-enabled* respectivamente. El segundo contiene enlaces simbólicos a los sitios del primero que estén activos.

---

## Sitio por defecto

---

La configuración del servidor virtual por defecto se puede consultar

```
gedit /etc/apache2/sites-available/default
```

lo que nos mostrará

```
<VirtualHost *:80>

    ServerAdmin webmaster@localhost

    DocumentRoot /var/www

    <Directory />

        Options FollowSymLinks

        AllowOverride None

    </Directory>

    <Directory /var/www/>

        Options Indexes FollowSymLinks MultiViews

        AllowOverride None

        Order allow,deny

        allow from all

    </Directory>
```



```
ScriptAlias /cgi-bin/ /usr/lib/cgi-bin/

<Directory "/usr/lib/cgi-bin">

    AllowOverride None

    Options +ExecCGI -MultiViews +SymLinksIfOwnerMatch

    Order allow,deny

    Allow from all

</Directory>

ErrorLog ${APACHE_LOG_DIR}/error.log

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

LogLevel warn

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

```
Alias /doc/ "/usr/share/doc/"

<Directory "/usr/share/doc/">

    Options Indexes MultiViews FollowSymLinks

    AllowOverride None

    Order deny,allow

    Deny from all

    Allow from 127.0.0.0/255.0.0.0 ::1/128

</Directory>

</VirtualHost>
```

Podemos comprobar que en el directorio de sitios activos se encuentra un archivo *000-default* que es el enlace simbólico al sitio por defecto.

Estudia y comenta en clase las directivas incluidas en la configuración del sitio virtual por defecto.

En el archivo podemos observar que el directorio raíz es */var/www/* y la configuración que tiene el directorio. Ten en cuenta que es necesario reiniciar Apache para que los cambios en la configuración surtan efecto.

Un directorio hereda las directivas del superior si no se sobrescriben. Por ello, todos los directorios que creamos dentro de `/var/www/` tendrán la misma configuración que el raíz excepto en aquellas directivas que especifiquemos en la configuración propia del directorio.

Vamos a modificar la configuración del sitio por defecto. Crea un archivo `DAW.html` en el directorio raíz con tu nombre y tus datos. Ahora queremos que el índice o página principal del directorio sea la que acabas de crear. ¿Qué directiva debes añadir? ¿Es posible establecer más de una página principal? En caso afirmativo, ¿cuál se cargará si accedemos al sitio sin especificar página? ¿Qué sucede si no hay ninguna de las páginas establecidas como página principal? Busca una directiva que deshabilite el listado de archivos incluso si no encuentra ninguna de las páginas por defecto.

Crea un nuevo directorio dentro del directorio raíz. El nombre del directorio será tu apellido. Crea varios archivos html pero no establezcas ninguno como el principal. Accede al directorio desde un navegador. Ahora modifica la configuración del directorio para que liste el contenido. ¿Qué sucede? Establece uno de los archivos como el principal para el directorio y vuelve a probar.

Recuerda que después de cada cambio en la configuración es necesario reiniciar Apache.

Como ya debes haber visto en la práctica, para deshabilitar el listado de los contenidos de un servidor, host virtual o directorio se usa

```
Options -Indexes
```

Si quieres activarlo para un servidor, directorio o host virtual se utiliza

```
Options Indexes FollowSymLinks
```

La opción de seguir los enlaces no es obligatoria pero se suele añadir.

Ten en cuenta que hay que mirar que la configuración no esté sobrescrita en la configuración del host virtual por defecto o de algún directorio en particular. Lo que debe hacerse en estos casos es una comprobación de la directiva de lo particular a lo general: directorios, host virtual y servidor.

---

## Modificando los mensajes de error

---

La directiva [ErrorDocument](#) sirve para establecer mensajes de error personalizados para diferentes situaciones. Los códigos de error que pueden utilizarse son los definidos por la organización w3 para el protocolo http en este caso la versión 1.1 y se pueden [consultar aquí](#). En esta [otra página](#) se explican de una manera más comprensible. Ten en cuenta que el código es solo el último número de cada apartado y tiene tres cifras. Esta directiva se usará tantas veces como se crea necesario. Se establece de manera independiente **para cada sitio virtual** y se suele escribir debajo de CustomLog.

En el apartado anterior hemos visto que se usa una variable de entorno llamada `APACHE_LOG_DIR` para determinar el directorio donde se ubicarán los archivos de log. En esta instalación es `/var/log/apache2/` pero si no podemos buscar los archivos en la estructura de directorios o consultar el valor de la variable de entorno.

Establece acciones personalizadas para el error 404. Primero redirige las consultas a páginas no encontradas a la página principal del sitio. Luego prueba a redirigir estos errores a la página principal del instituto. En la tercera prueba debes mostrar un mensaje directamente que diga “página no encontrada, consulte a Nombre Apellido” con tu nombre y apellido. Por último crea un archivo html que explique el error y haz que se muestre si se da el caso.

Consulta la lista de errores definidos y apunta los que creas más útiles. Coméntalos en clase.

---

## Alias a otros directorios

---

Por defecto la estructura de un sitio web será la que demos a los directorios desde el punto de montaje del directorio raíz del sitio, sin embargo es posible incluir otros directorios del árbol de nuestro servidor y hacer que parezcan parte del sitio como cualquier otra carpeta que esté contenida físicamente dentro. Es algo similar a tener un enlace simbólico a archivos o carpetas en otro directorio.

En el archivo de configuración del sitio principal visto antes ya existe un alias a un directorio que se encuentra fuera de la carpeta definida para el sitio.

Para realizar esta operación se utiliza la directiva [Alias](#) y justo debajo (no es obligatorio pero sí una buena práctica) se configura con la directiva [Directory](#) como cualquier otro directorio.

En la guía rápida de directivas, se pueden aplicar a directorios todas las que llevan “d” en la tercera columna.

Crea un directorio /var/extras y coloca dentro una página HTML que te permita diferenciarla de las demás. Configura el directorio para que aparezca como /extras en nuestro sitio web y aplica las directivas de directorio que consideres importantes.

---

## Redirecciones

Algunas veces es útil poder redirigir las llamadas a una dirección web para que se procesen en otro punto. Por ejemplo si hemos cambiado la estructura de nuestro sitio y hemos reubicado la página de contacto podemos querer que los usuarios que accedan mediante la antigua dirección sean redirigidos automáticamente a la nueva. También puede ser útil si hemos dividido nuestro sitio para facilitar la gestión o el mantenimiento.

Para ello se usa la directiva [Redirect](#) que nos permite asociar una dirección absoluta o relativa a otra.

Crea una redirección en el sitio principal del servidor para que en caso de que alguien quiera acceder a /aficiones se redirija a una página que visites muy a menudo.

Crea ahora una redirección que en el caso de alguien quiera acceder a /ext se le muestre el contenido de /extras que creaste en el punto anterior.

---

## Creación de hosts virtuales

Existen tres formas diferentes de crear hosts virtuales en Apache:

1. Basados en **nombres**: Este tipo es la opción más común. Se configura todo para que múltiples dominios DNS apunten a una única máquina con Apache. Requiere configuración del servidor DNS para que funcione. Este método hace muy fácil migrar nuestro servidor a otra dirección IP. Es el **único método que vamos a estudiar**.
2. Basados en **IP**: En este método se necesita configurar las direcciones IP de cada sitio en Apache. El servidor físico tendrá varias direcciones IP, una para cada sitio.
3. Basados en **puertos**. Cada sitio se atenderá en la misma IP o nombre pero en distintos puertos. Es una extensión de cualquiera de las alternativas anteriores.
4. Varios **servidores principales**: En esta opción se mantienen varias configuraciones principales en el servidor. Solo se recomienda su uso si es necesario tener un archivo de configuración diferente para cada sitio. Apenas se usa y es la opción menos recomendada para configurar nuestro servidor.

El uso de estas alternativas no es excluyente. Pueden combinarse varias o todas en el mismo servidor Apache.

En [esta dirección](#) está la documentación sobre hosts virtuales de apache. En [esta otra](#) se pueden ver ejemplos de configuración.

Estudia y comenta en clase los primeros ejemplos del enlace anterior. Céntrate solo en los primeros, que tratan de los tipos listados anteriormente.

Con cualquiera de las tres alternativas es necesario activar primero el módulo de hosts virtuales.

```
sudo a2enmod vhost_alias
```

lo que activa el módulo

```
Enabling module vhost_alias.
```

To activate the new configuration, you need to run:

```
service apache2 restart
```

y nos informa de que es necesario reiniciar el servidor.

```
apachectl restart
```

Comprobamos que el módulo se encuentre activo.

```
apachectl -M
```

Vemos que aparece al final de la lista.

```
Loaded Modules:  
  
core_module (static)  
  
log_config_module (static)  
  
logio_module (static)  
  
mpm_worker_module (static)  
  
http_module (static)  
  
so_module (static)  
  
alias_module (shared)  
  
auth_basic_module (shared)  
  
authn_file_module (shared)
```

```
authz_default_module (shared)

authz_groupfile_module (shared)

authz_host_module (shared)

authz_user_module (shared)

autoindex_module (shared)

cgid_module (shared)

deflate_module (shared)

dir_module (shared)

env_module (shared)

mime_module (shared)

negotiation_module (shared)

reqtimeout_module (shared)

setenvif_module (shared)

status_module (shared)

vhost_alias_module (shared)
```



Syntax OK

Para usar estos métodos es necesario poder hacer que todos los nombres de dominio asociados a los hosts virtuales que vamos a crear apunten a la dirección IP o las direcciones IP de nuestro servidor web. Si hemos contratado un nombre de dominio con un proveedor éste nos proporcionará la manera de hacerlo.

En caso de que tengamos o queramos publicar nuestros propios sitios debemos configurar un servidor DNS. En los apéndices encontrarás una manera muy simple de configurar DNS en nuestro servidor. Otra opción para probarlo es editando el archivo *hosts*

```
sudo gedit /etc/hosts
```

y en él añadir los alias necesarios.

Los sitios virtuales utilizan una estructura de directorios similar a la que hemos visto para los módulos. En

```
/etc/apache2/sites-available/
```

Encontramos los sitios disponibles, que al ser activados crean un alias en

```
/etc/apache2/sites-enabled/
```

### Hosts virtuales basados en nombres

Este método es el más recomendado ya que requiere una única dirección IP para poder alojar múltiples sitios.

Podemos comprobar que en el archivo */etc/apache2/ports.conf* se encuentra la directiva

```
NameVirtualHost *:80
```

Esta directiva es la necesaria para poder usar los hosts virtuales basados en nombres. Al activarse deshabilita el servidor principal (el que habíamos usado en la instalación manual) y por ello en este tipo de instalación el servidor virtual es realmente un sitio virtual por defecto.

El primer paso es crear los registros DNS para que el nombre de dominio apunte a nuestra máquina o añadirlos en el fichero *hosts*.

Debemos crear un directorio para cada sitio y así tener los documentos separados. Además habría que crear un archivo *index.html* en el directorio para que lo cargue al consultar el sitio.

```
mkdir /var/www/ejemplo2.es
```

En este punto podemos acceder a los datos de ejemplo2 mediante la dirección <http://localhost/ejemplo2.es/> pero nuestro objetivo es poder hacerlo escribiendo <http://ejemplo2.es/>

Lo tercero que deberíamos hacer es escribir los datos del nuevo host virtual en la configuración de Apache. Recuerdo que hemos visto dos casos diferentes de dónde se encontraba la configuración de Apache, pero con la opción que estamos usando actualmente la configuración principal estaba en el archivo */etc/apache2/apache2.conf* y desde él se importaban otros muchos. Podemos ver que se importa una carpeta completa de la que coge la configuración de los hosts virtuales.

```
...  
  
# Include the virtual host configurations:  
  
Include sites-enabled/  
  
...
```

Sin embargo, antes de hacer esto debemos crear el host virtual en la carpeta de hosts disponibles y lo activaremos cuando esté terminado. La carpeta en la que podemos añadir los datos de nuestros hosts virtuales es */etc/apache2/sites-available*

Si vamos a la carpeta que nos indican podemos consultar el archivo de la configuración de un host virtual, el sitio por defecto. Esto puede servirnos ejemplo y guía para crear otros.

Creamos el archivo de configuración

```
sudo gedit /etc/apache2/sites-available/ejemplo2.es
```

y escribimos lo siguiente

```
NameVirtualHost 10.0.2.15:80

<VirtualHost 10.0.2.15:80>

    ServerName ejemplo2.es

    ServerAlias www.ejemplo2.es

    ServerAdmin alguien@ejemplo2.es

    DocumentRoot /var/www/ejemplo2.es

    #

    # Aquí pueden ir otras directivas.

    # Por defecto hereda las del archivo principal.

    #

</VirtualHost>
```

Entre todas las directivas que se pueden incluir son especialmente útiles las referentes a **directorios** ya que si el sitio consta de varios (lo más habitual) puede interesarnos tener configuraciones diferentes para cada uno.

También es habitual configurar registros de error independientes para cada sitio mediante la directiva adecuada. Esto se hace dentro de la configuración de cada sitio virtual dejando el principal para el servidor en si. Ya vimos que en el sitio por defecto se incluían.

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

Pero por supuesto debemos **dar otros nombres a nuestros registros** para que no se usen los mismos.

Es buena idea **revisar la configuración del sitio principal** para hacernos una idea de qué configurar ya que en el ejemplo anterior he reducido las directivas al mínimo.

Si en lugar de la dirección IP escribimos \* se realizará para todas las IP que tenga la máquina.

En los dos casos en los que hemos escrito la dirección IP y el puerto podríamos haber puesto un asterisco (\*) en lugar de la dirección pero se estaríamos indicando a Apache que escuche todas las peticiones lo que podría generar conflictos. Siempre es preferible concretar los datos.

Ahora ya hemos configurado el nuevo host así que podemos activarlo.

```
sudo a2ensite ejemplo2.es
```

y recargar los sitios de Apache

```
service apache2 reload
```

si al hacerlo nos da el error

```
* Reloading web server config apache2                                apache2: Could not  
reliably determine the server's fully qualified domain name, using 127.0.1.1 for ServerName
```

Debemos abrir el archivo *httpd.conf* y añadir

```
ServerName LocalHost
```

O el nombre completo de nuestro servidor.

Ya podemos acceder a nuestro documento mediante [www.ejemplo2.es](http://www.ejemplo2.es)

Para desactivar un sitio virtual se hace con (por ejemplo para desactivar el sitio por defecto).

```
sudo a2dissite default
```

En caso de recibir un **error “Apache2 NameVirtualHost \*:80 has no VirtualHosts”** suele ser porque has puesto la directiva [NameVirtualHost](#) en ambos archivos de configuración de los sitios. Al poder ser una directiva de servidor, con ponerla en el `httpd.conf` vale.

Crea un nuevo host virtual basado en nombre y prueba que todo sea correcto. Crea otro y verás que ahora tienes la posibilidad de acceder a tres sitios en la misma máquina. Haz la asociación mediante el archivo `hosts` de Linux. Posteriormente eliminaremos la entrada del archivo `hosts` y realiza la asociación mediante un servidor DNS. Por último desactiva el sitio virtual.

Para esta práctica vas a necesitar configurar tu servidor DNS. [En los apéndices explico cómo.](#)

## Control de acceso

En este punto comenzamos con varios aspectos relacionados con la seguridad de nuestro servidor web.

Con este primer apartado vamos a considerar los aspectos que nos van a permitir filtrar el acceso a determinados recursos.

El control de acceso se refiere a cualquier método que nos permita filtrar el acceso a algún recurso determinado. Hay tres módulos implicados en el control de acceso en Apache: [mod\\_authz\\_host](#), [mod\\_setenvif](#) y [mod\\_rewrite](#).

También existen tres métodos para gestionar el control de acceso relacionados con estos módulos.

### Control de acceso basado en la dirección

Este tipo de control se basa en el uso del módulo [mod\\_authz\\_host](#) y en las direcciones IP de las máquinas que quieran acceder a nuestro servidor. Podemos comprobar que esté módulo ya está activo en nuestra instalación y de hecho ya hemos usado anteriormente este tipo de control de acceso para permitir acceder a partes de nuestro servidor desde la máquina anfitrión en las prácticas.

Las directivas que se utilizan para este caso son *Allow* y *Deny*. Generalmente van asociadas con otra, *Order*. El uso de estas directivas es

```
Allow from dirección
```

En lugar de una dirección IP se puede utilizar el nombre de la máquina a la que queremos permitir o denegar el acceso.

Si queremos evitar que desde una dirección IP determinada se acceda a nuestra máquina (o a un recurso determinado, recuerda que nosotros ya lo hemos usado dentro de configuración de directorios específicos) escribiríamos

```
Deny from 23.43.234.22
```

También podemos filtrar por un nombre de dominio (DNS)

```
Allow from maq_ejemplo.instituto.es
```

Investiga cómo se puede filtrar un dominio entero y un rango de direcciones IP en lugar de máquinas sueltas.

¿es posible filtrar direcciones con una máscara que no sea múltiplo de 8? ¿cómo?

La directiva *Order* es casi más importante que éstas que hemos visto. Indica si se aplicarán primero las cláusulas *Deny* o las *Allow*, pero además indica que se hará para lo que no esté especificado.

Si usamos

```
Order Allow, Deny
```

Primero se procesan todas las directivas *Allow* y si no ninguna encaja, la petición se rechaza. Luego se procesan todas las directivas *Deny*, si encaja con alguna se rechaza la petición. Si encaja con una directiva *Allow* y una *Deny* se aplica la última por lo tanto se rechaza. Todas las peticiones que no coincidan con ninguna directiva se rechazan.

Si usamos

```
Order Deny, Allow
```

Primero se intenta que encajar la petición con todas las directivas *Deny* y si coincide se rechaza la petición. Luego se procesan las directivas *Allow* y si encaja con alguna se permite el acceso. Si alguna petición encaja con una directiva *Deny* pero también con una *Allow*, se permite. Todas las peticiones que no coincidan con ninguna directiva se permiten.

Crea un directorio con tu nombre en el sitio por defecto. Permite el acceso desde la máquina anfitrión pero deniégalo desde la máquina en la que se encuentra el servidor.

Ten en cuenta que estas directivas no son exclusivas de los directorios. Pueden aplicarse al sitio web completo por ejemplo. Para ello irán en la configuración general del sitio y no dentro de unas etiquetas *Directory*.

---

### Control de acceso por variable de entorno

Este tipo de control de acceso se realiza mediante el uso de los módulos [mod\\_authz\\_host](#) y [mod\\_setenvif](#). Se basa en permitir el acceso según la configuración de alguna variable de entorno de la máquina del usuario y por ello no es muy recomendable.

---

### Control de acceso con el módulo rewrite

Mediante el uso del módulo [mod\\_rewrite](#) podemos controlar el acceso según criterios arbitrarios. Por ejemplo si queremos denegar el acceso durante el periodo de las ocho de la tarde a las 6 de la mañana, escribiremos

```
RewriteEngine On

RewriteCond %{TIME_HOUR} >20 [OR]

RewriteCond %{TIME_HOUR} <07

RewriteRule ^/fridge - [F]
```

El uso de este método se basa en las directivas [RewriteCond](#) y [RewriteRule](#) pero queda fuera del alcance de este curso y se indica solo como introducción a las posibilidades que permite.



---

## Autenticación y autorización

---

Estos dos términos van ligados pero no son lo mismo a pesar de que mucha gente los confunde. La autenticación consiste en comprobar que alguien es quien dice ser mientras que la autorización es comprobar que alguien tiene permiso para acceder a un lugar o recurso determinado.

Por ejemplo, si quieres viajar al extranjero (fuera de los países que ha firmado el Acuerdo de Schengen) necesitas un pasaporte y un visado. El pasaporte es un documento general que sirve para demostrar que eres quien dices ser mientras que el visado te autoriza a visitar un país determinado.

En informática, la autenticación puede darnos acceso a diferentes recursos para los que estemos autorizados, e incluso estas autorizaciones pueden variar dependiendo de diferentes circunstancias. Por ejemplo, podemos tener permiso para acceder a determinado recurso en una franja horaria determinada o desde la oficina de trabajo pero no desde casa.

El proceso de autorización suele implicar la autenticación. Por ejemplo si un cliente quiere acceder a determinado recurso el servidor le pide que se autentique (mediante un mensaje de estado 401: Authoritation Required) por ejemplo con una solicitud de usuario y contraseña. Si esta autenticación es positiva se permitirá el acceso y si no se responderá con otro mensaje 401.

Un problema de esta forma de autenticación es que la contraseña ni se encripta ni se oculta, por lo que más adelante hablaremos del protocolo https. Cualquiera con un sniffer podría interceptar los nombres de usuario y contraseñas.

---

### El módulo `mod_auth`

---

Este módulo nos permite realizar autorización de una manera bastante básica. Nos va a permitir establecer acceso mediante usuario y contraseña a secciones de nuestro sitio.

Por ejemplo podemos establecer un directorio al que haya que acceder con contraseña. Podemos elegir cualquier directorio siempre que el **usuario** que lanza apache (directiva `User`) tenga acceso a él. No es necesario que el directorio esté en la estructura básica de nuestro sitio, pero si no está tendremos que establecer un *Alias*.

```
mkdir /var/www/privado
```

```
mkdir /var/secreto
```

Ahora añadimos al sitio en el que queremos gestionar la autenticación

```
gedit /etc/apache2/sites-available/default
```

la configuración para el directorio

```
<Directory "/var/www/privado">

    AuthName "Acceso privado: Introduzca su usuario y contraseña"

    AuthType Basic

    AuthUserFile /var/secreto/.miembros

    Require valid-user

</Directory>
```

- *AuthName* le indica al usuario qué hacer. Es un mensaje para el usuario.
- *AuthType* es el tipo de autenticación que usaremos; http solo admite **Basic**. La otra opción que existe es **Digest** que a diferencia de la opción *Basic* no transmite los nombres de usuario y contraseña como texto plano (y por lo tanto es una opción de seguridad mejor) pero que no está disponible para todos los navegadores Web como opción “out-of-the-box”. El cifrado que usa la opción *Digest* es bastante débil por y aunque su uso no es idéntico al de la opción *Basic* es bastante similar, por lo que no lo veremos.
- *AuthUserFile* es el archivo que se utilizará para guardar las contraseñas. Indica la ruta y se llamará *.miembros*.
- *Require* especifica que será necesario acceder con un usuario válido.

Ahora necesitamos crear el archivo de contraseñas.

```
/usr/bin/htpasswd -c /var/secreto/.miembros srsergio
```

Donde deberás especificar la ruta a los ejecutables de apache si no es la misma, -c es la opción para crear el archivo así que si vas a añadir otro miembro debes quitarla. La ruta al archivo de contraseñas debe ser la misma especificada para la configuración del sitio y *srsergio* será el nombre de usuario que queremos crear. Nos pedirá que creemos una contraseña.

Podemos asegurarnos de que se ha creado el usuario abriendo el archivo. Date cuenta de que la contraseña está codificada.

```
gedit /var/secreto/.miembros
```

Reinicia apache

```
apachectl restart
```

Ahora ya tienes una sección para acceso solo con usuario y contraseña.

Ten en cuenta que la debilidad de **http** en la transmisión de la información hace que este método **no sea seguro** pero se puede solucionar usando **https** como veremos más adelante.

Crea una sección privada en tu sitio e introduce algunos archivos html. Crea al menos dos usuarios que tengan acceso y prueba a acceder tanto con ellos como con datos no válidos.

Siempre se habla de que la información que se transmite por Internet como texto plano es muy fácil de interceptar. ¿Cómo se haría esto es nuestra red interna? ¿Qué programas necesitas? ¿Serías capaz de interceptar un usuario y contraseña enviados por un usuario para acceder a la sección privada creada en el punto anterior?

Ampliación: Investiga cómo se haría lo mismo con la opción *Digest*. Parece ser que Apache trae Digest desactivado por defecto y lo único que hace es codificar pero no encriptar los datos por lo que no merece la pena ni activarlo.

## Los ficheros .htaccess

Las soluciones vista hasta ahora no son muy adecuadas si queremos poder delegar la creación y control de zonas privadas para miembros determinados. Esto puede ser muy útil si por ejemplo hemos montado una web para una empresa que tiene su propio administrador de sistemas ya que evitará que tengamos que gestionar todo nosotros. También nos facilitará el trabajo si hay muchos cambios en las zonas privadas o miembros que se conecten a ellas.

Podemos conocer más sobre htaccess en [este artículo](#). Otra web nos introduce a algunos de los [usos más comunes](#) de estos archivos con una [segunda parte](#).

Para permitir el uso de ficheros .htaccess en nuestro servidor o sitio virtual (la directiva se puede usar en ambos entornos) lo primero que debemos hacer es modificar el archivo de configuración. En mi caso voy a hacerlo en el sitio virtual por defecto.

Debemos modificar la directiva

```
AllowOverride None
```

Y cambiarla a

```
AllowOverride AuthConfig
```

El lugar en el que modificar la directiva depende de lo que necesitemos. Ten en cuenta que las directivas se heredan si no se encuentra otra más específica, por ello, en nuestro archivo de configuración del sitio virtual por defecto debería ir en el directorio /var/www por lo menos ya que si lo ponemos en el raíz pero no en el primero, se mantendría la configuración anterior.

```
<Directory /var/www/>

    Options Indexes FollowSymLinks MultiViews

    AllowOverride AuthConfig
```

```
Order allow,deny

allow from all

</Directory>
```

Lo que permite que modifiquemos las directivas de autorización mediante un fichero `.htaccess`. En muchos sitios indican que hay que permitir la sobre escritura de todas las directivas mediante `AllowOverride All` pero es evidente que es peor opción.

Luego reiniciamos el servidor Apache

```
apachectl restart
```

Ahora podríamos crear ficheros directorios y configurar su control de acceso en cada uno de ellos.

```
mkdir /var/www/ficheros/

cd /var/www/ficheros/
```

En cada directorio que queremos gestionar así debemos crear un fichero `.htaccess` y darle un contenido similar al siguiente.

```
AuthName "Sección Privada: Prueba de .htaccess"

AuthType Basic

AuthUserFile /var/secreto/.miembros

Require valid-user
```

Ten en cuenta que:

- Lo que hemos hecho ha sido sacar la configuración del control de acceso del archivo de configuración del servidor/sitio virtual a un archivo independiente.
- He usado el mismo archivo de usuarios y contraseñas que en el punto anterior para darle coherencia a los ejemplos, pero esto no es necesario.
- Los usuarios y sus contraseñas se crearían igual que en el apartado anterior.
- La creación o modificación de un fichero de este tipo no implica reiniciar el servidor.
- Se debe mejorar la seguridad cambiando los permisos de acceso al fichero `.htaccess`. Por lo menos el usuario de Apache debe tener acceso.

Crea dos directorios en tu sitio por defecto y configúralos para que en el primero puedan acceder dos usuarios y en el segundo solo uno de ellos. Para ello en *Require* debe ir su nombre de usuario.

## Agrupando usuarios para el control de acceso

Si queremos refinar más el control de acceso tenemos varias alternativas. Por ejemplo podemos permitir el acceso a solo algunos usuarios especificando sus nombres en la cláusula *Require* en cualquier sitio que la usemos.

```
require user sergio maria
```

Otra alternativa es crear archivos de usuarios diferentes para distintos directorios de nuestro sitio.

```
<Directory "/var/www/ventas">

    AuthName "Acceso privado: Introduzca su usuario y contraseña"

    AuthType Basic

    AuthUserFile /var/secreto/.miembros-ventas
```

```
        Require valid-user

    </Directory>

    <Directory "/var/www/finanzas">

        AuthName "Acceso privado: Introduzca su usuario y contraseña"

        AuthType Basic

        AuthUserFile /var/secreto/.miembros-finanzas

        Require valid-user

    </Directory>
```

La última opción pasará por usar la directiva *AuthGroupFile*. Así podremos tener todos los usuarios en el mismo fichero y luego asignarles grupos en otro archivo.

```
<Directory "/var/www/ventas">

    AuthName "Acceso privado: Introduzca su usuario y contraseña"

    AuthType Basic

    AuthUserFile /var/secreto/.miembros

    AuthGroupFile /var/secreto/.grupos
```

```
        Require group ventas

</Directory>

<Directory "/var/www/finanzas">

    AuthName "Acceso privado: Introduzca su usuario y contraseña"

    AuthType Basic

    AuthUserFile /var/secreto/.miembros

    AuthGroupFile /var/secreto/.grupos

    Require group finanzas

</Directory>
```

Ten en cuenta que:

- Se usa el mismo archivo de miembros para todos los directorios.
- Hay que añadir la directiva y el fichero de grupos.
- En la cláusula Requiere se indica el grupo que puede acceder.

El archivo .grupos será un fichero de texto con algo como

```
ventas: sergio maria

finanzas: carlos roberto
```



Modifica la configuración del ejercicio anterior para que puedan acceder dos grupos de usuarios diferentes a cada directorio con el último método. ¿Puede un usuario estar en más de un grupo?

Añade un tercer directorio y otro grupo. Da acceso al tercer directorio a dos de los tres grupos. ¿Cómo se hace esto?

## El protocolo HTTPS

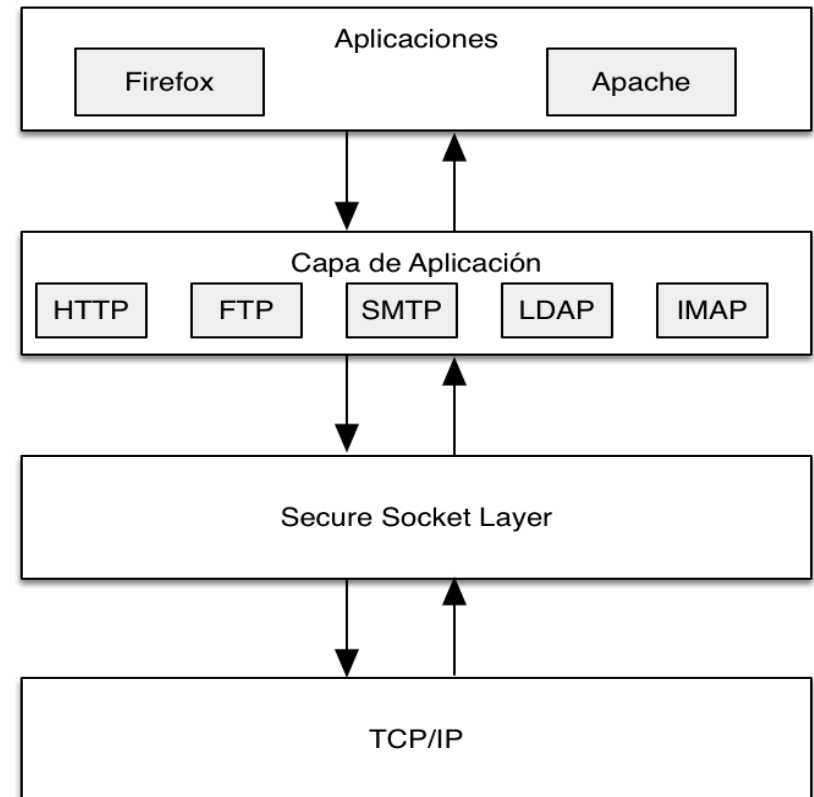
Ya hemos visto que el protocolo HTTP es muy inseguro para transmitir información sensible. Es muy fácil capturar esta información y leer datos como nombres de usuarios o claves. Cualquier información que se envía va sin cifrar por lo que se transmite como un texto que cualquiera que capture nuestra comunicación puede leer.

El protocolo HTTPS se utiliza para evitar este problema. Sus siglas se corresponden con *Hypertext Transfer Protocol **Secure*** lo que ya indica que añade seguridad a HTTP. Realmente no es un protocolo sino que se basa en introducir una capa con el protocolo SSL (o derivados, más adelante se habla de todo esto) entre la capa de transporte y la capa de aplicación en el conjunto de protocolos TCP/IP.

El protocolo HTTPS proporciona **autenticación** de los usuarios con el servidor con el que se conecta. Por ejemplo cuando nos conectamos a nuestro correo web (por ejemplo Gmail) suele hacerse mediante una conexión HTTPS.



Con el uso de este protocolo también se consigue **encriptación** de la información que enviamos y recibimos. Por ello cuando nos conectamos a un servidor que solo utiliza HTTPS para la autenticación el navegador nos avisa (una vez nos hemos conectado) de que el resto de la comunicación no usa HTTPS y por lo tanto la transferencia de información puede ser insegura. Esto nos protege contra los ataques de tipo *Man-In-The-Middle*. Todo lo que se envía en un mensaje HTTPS está encriptado, incluyendo las cabeceras. Por supuesto la encriptación también está sujeta a ataques que intenten descifrar la clave y el algoritmo usados para cifrar el contenido.



Investiga qué algoritmos y qué tipo de claves se utilizan en Internet actualmente.

¿Qué son los algoritmos de clave simétrica y los algoritmos de clave asimétrica? ¿Qué con las claves privadas y claves públicas? ¿Cómo se utilizan?

[Esta página de la Wikipedia](#) te puede ayudar a comprender estos conceptos y los que veremos a continuación. Necesitarás tener una idea de los puntos anteriores de este ejercicio para entenderla.

Un sitio que use HTTPS debería tener todos sus contenidos protegidos por este protocolo para evitar posibles ataques o robos de información a través de las partes inseguras.

HTTPS es un poco menos eficiente que HTTP por lo que si se utiliza en sitios donde la transferencia de información es muy grande puede notarse en el rendimiento. Por supuesto en el caso de que la información transferida sea sensible, las ventajas compensan con creces a los inconvenientes.

El protocolo HTTPS usa el **puerto 443** por defecto. Al igual que sucedía con HTTP y el puerto 80, en caso de que se utilice este puerto no es necesario que el cliente lo indique en la barra de direcciones del navegador.

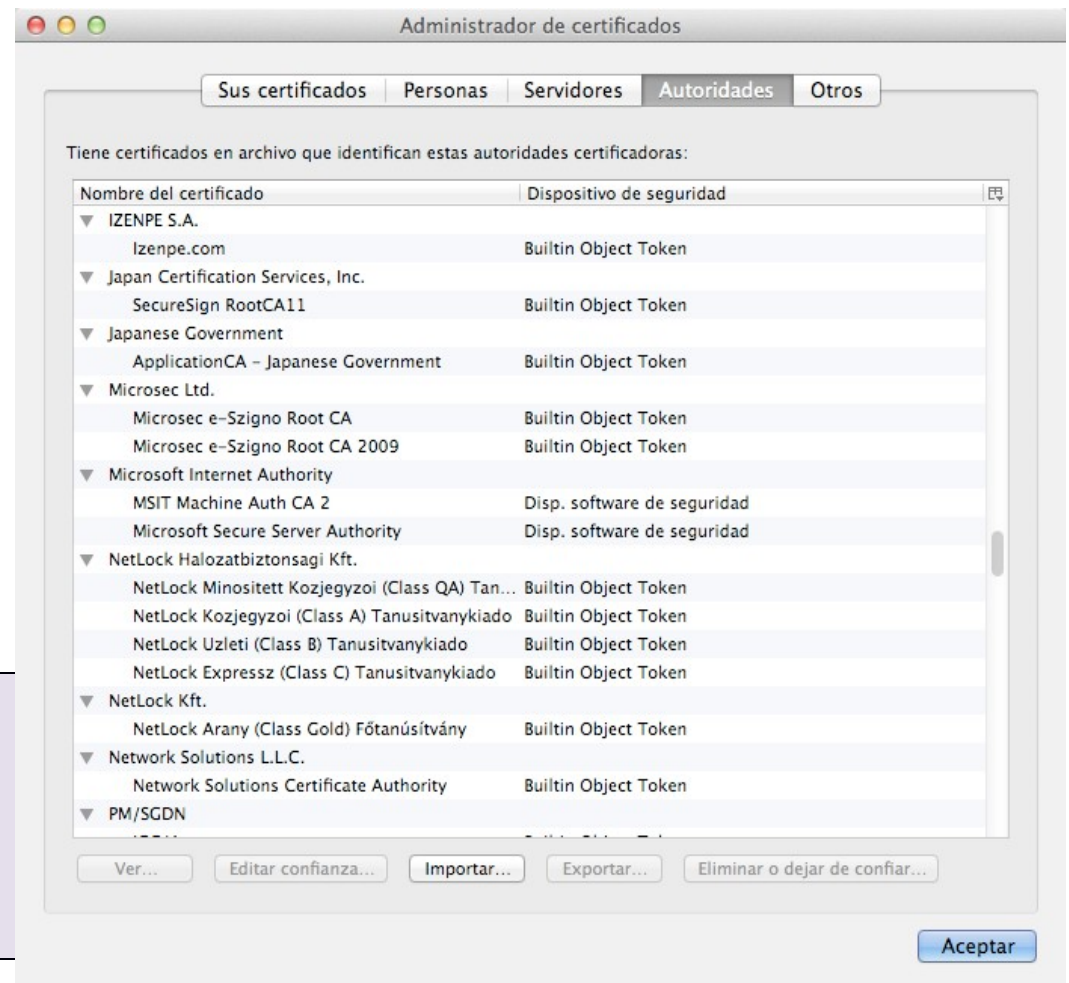
## Certificados Digitales

El protocolo HTTPS encripta la comunicación para que quien capture tramas de ella no pueda ver los contenidos. Los navegadores web actuales basan el uso de HTTPS en el conocimiento de Autoridades Certificadoras que emiten Certificados Digitales para asegurar que el servidor al que nos conectamos es quien dice ser. Estas Autoridades Certificadoras son o agentes dedicados a ello específicamente o empresas como Microsoft.

El uso de HTTPS se basa en la confianza que nos proporcionen las entidades que emiten los certificados. Cuando usamos un navegador web determinado, la empresa que lo ha desarrollado ya ha introducido en él determinadas Autoridades Certificadoras que considera de confianza. Estas entidades se pueden consultar en el propio navegador y podemos modificarlas. En la imagen puedes ver parte de la lista incluida en Firefox.

Investiga en los diferentes navegadores que tengas instalados cómo se consulta la lista de Autoridades Certificadoras en las que se confía.

En la captura de imagen se pueden ver otras pestañas para el administrador de certificados. ¿Para qué sirve cada una de ellas?



Existen muchos servidores que usan certificados no emitidos por estas autoridades. En esos casos depende del usuario el aceptarlos o no. El navegador muestra una advertencia cuando vamos a conectarnos y tendremos que decidir si queremos seguir o preferimos no conectarnos con el destino.

En la imagen podemos ver un ejemplo con varias partes:

**Esta conexión no está verificada**

Ha pedido a Firefox que se conecte de forma segura a **sede.muface.gob.es**, pero no se puede confirmar que la conexión sea segura.

Normalmente, cuando se intenta conectar de forma segura, los sitios presentan información verificada para asegurar que están en el sitio correcto. Sin embargo, la identidad de este sitio no puede ser verificada.

**¿Qué debería hacer?**

Si normalmente accede a este sitio sin problemas, este error puede estar ocurriendo porque alguien está intentando entrar al sitio, y no debería continuar.

**▼ Detalles técnicos**

sede.muface.gob.es usa un certificado de seguridad no válido.

No se confía en el certificado porque no se ha proporcionado la cadena de emisor.

(Código de error: sec\_error\_unknown\_issuer)

**▼ Entendiendo los riesgos**

Si sabe lo que está haciendo, puede obligar a Firefox a confiar en la identificación de este sitio. Incluso aunque confíe en este sitio, este error puede significar que alguien esté interfiriendo en su conexión.

No añada una excepción a menos que sepa que hay una razón seria por la que este sitio no use identificación confiable.

1. Si no confiamos en el destino debemos pulsar el botón para salir.
2. Podemos obtener más información. En este caso vemos que la entidad es gov.es por lo que podemos decidir confiar en él.
3. Si queremos confiar debemos leer los riesgos que conlleva.
4. Podemos añadir una excepción de seguridad para confiar en este sitio a partir de ahora.

¿Qué es la firma digital y cuál es su relación con los certificados digitales?

Un **Certificado Digital** es un documento electrónico que enlaza una clave pública con una Firma Digital e información personal sobre la persona u organización que quiere usar la clave pública. Sirve para asegurar que esa clave pertenece a dicha persona u organización. La información personal que se adjunta suele ser el nombre, la dirección, el correo electrónico, etc. La Firma Digital suele ser la de una Entidad Certificadora reconocida para que los clientes puedan confiar en que la Clave Pública y la información personal corresponden a la misma persona/organización. Este tipo de certificados se conocen como **Certificados Raíz** ya que

se encuentran en el nivel más alto de un árbol de certificados. El usuario debe fiarse de la entidad emisora del Certificado Raíz ya que es el que asegura la autenticidad de todos los certificados emitidos por ella. La lista de certificados que se incluyen por defecto en un navegador web corresponden a esta categoría y el usuario confía en los desarrolladores del navegador para que se aseguren de que esos certificados son de confianza.

¡Como se puede ver la seguridad en Internet se basa en muchos niveles de confianza! Nadie debe asustarse. Las autoridades que incluyen los principales navegadores están verificadas y con revisadas con regularidad.

En esta [página de la Wikipedia](#) se puede obtener mucha más información sobre los Certificados Digitales.

En [esta imagen](#) se puede ver el uso de certificados digitales.

Los **Servidores de Certificados** son los que se encargan de validar o certificar las claves.

En la actualidad se utilizan varios formatos de Certificados Digitales en Internet. El más extendido es el X.509

Lee [esta página](#) y comenta en clase los puntos que consideres más interesantes sobre la información obtenida.

En nuestro navegador podemos consultar los detalles de un certificado. Si nos ponemos sobre el certificado determinado y pulsamos “Ver” suele aparecer una opción “Detalles” en la que podemos consultar datos como el algoritmo y el valor de la firma.

## Obtener un certificado digital

Para poder utilizar HTTPS en nuestro servidor es necesario disponer de un Certificado Digital. Podemos obtener un Certificado Digital de una Autoridad Certificadora (*certificate authority*, CA) o crear nuestra propia Autoridad Certificadora y generar nuestros Certificados Digitales.

En España, podemos obtener Certificados Digitales a través de la [Fábrica Nacional de Moneda y Timbre](#) o mediante empresas como [Verisign](#).

Para obtener un Certificado de una Autoridad Certificadora generalmente hay que demostrar que somos quienes decimos ser (cómo lo establece cada autoridad) y generar una petición para nuestro servidor (*certificate signing request*, *CSR*) que se debe enviar a la autoridad. Cuando nuestra petición haya sido aceptada, ya podemos instalar el certificado en nuestro servidor.

## SSL/TLS

Secure Socket Layer (SSL) fue desarrollado por Netscape en los años 90. Han existido tres versiones de SSL pero actualmente se utiliza la 3.0. Ha derivado en otro protocolo *Transport Layer Security* (TLS) que también tiene tres versiones, la 1.0, la 1.1 y la 1.2. La mayoría de los navegadores actuales usan la versión 1.0

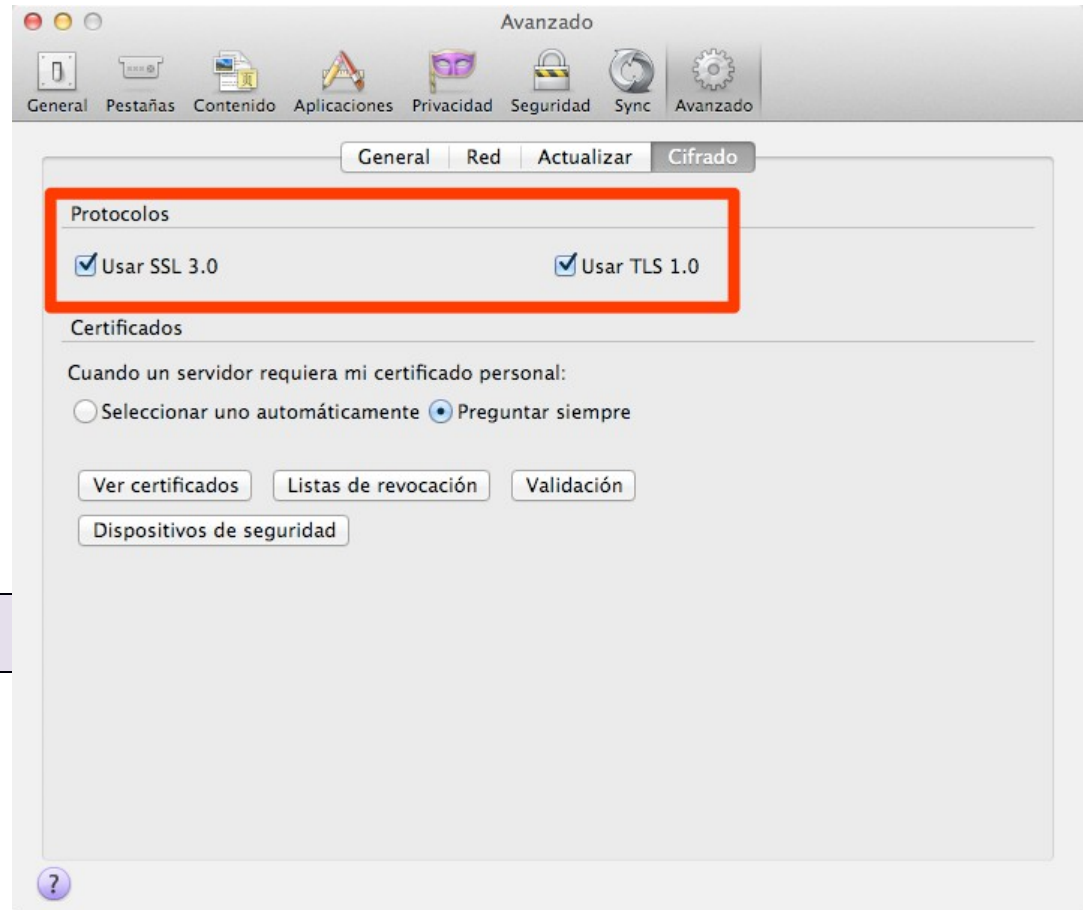
Como la hemos comentado con antelación, HTTPS es una implementación de HTTP sobre SSL o TLS en el servidor.

Este protocolo se coloca sobre la capa de transporte donde los dos protocolos más típicos son TCP y UDP.

Es un protocolo que se utiliza para asegurar **confidencialidad**, **autenticidad**, **integridad** y **no repudio** entre el cliente y el servidor.

Investiga y comenta los cuatro conceptos del párrafo anterior.

Existen dos modos: Uno en el que solo el servidor demuestra su identidad y otro en el que tanto el cliente como el servidor usan Certificados Digitales.





Las aplicaciones del uso de SSL/TSL son múltiples e incluyen la creación de redes privadas virtuales (VPN), el uso en comercio electrónico y en el correo electrónico, etc.

## HTTPS en Apache

Apache utiliza un módulo específico basado en un proyecto que se llama OpenSSL. Para utilizar HTTPS en Apache es necesario que el módulo `mod_ssl` esté activo.

```
sudo a2enmod ssl

service apache2 restart
```

Ahora el servidor debería estar escuchando tanto el puerto 80 (http) como en el 443 (https). Si miramos el archivo de configuración de puertos

```
gedit /etc/apache2/ports.conf
```

veremos que si el módulo `mod_ssl` está activo se añade la orden

```
NameVirtualHost *:80

Listen 80

<IfModule mod_ssl.c>

    # If you add NameVirtualHost *:443 here, you will also have to change

    # the VirtualHost statement in /etc/apache2/sites-available/default-ssl
```

```
# to <VirtualHost *:443>

# Server Name Indication for SSL named virtual hosts is currently not
# supported by MSIE on Windows XP.

Listen 443

</IfModule>
```

Hemos visto que esta distribución de Apache viene con dos sitios por defecto. El que no hemos usado se llama *Default-SSL* precisamente. Si lo activamos tendríamos ya un sitio con dicha configuración que escucharía por conexión segura.

```
sudo a2ensite default-ssl

service apache2 reload
```

Ahora tendríamos disponible la posibilidad de conectarnos de forma segura a ambos servidores. Si no especificamos el protocolo o usamos http se conectará de la forma estándar. Sin embargo, si nos conectamos mediante https nos muestra una excepción de seguridad como la que vimos antes. Esto es debido a que al instalar Apache se crea un certificado autofirmado para el sitio por defecto.

Si abrimos el archivo de configuración del sitio con ssl podemos ver cómo está configurado.

```
# A self-signed (snakeoil) certificate can be created by installing
# the ssl-cert package. See
# /usr/share/doc/apache2.2-common/README.Debian.gz for more info.
# If both key and certificate are stored in the same file, only the
```

```
# SSLCertificateFile directive is needed.  
  
SSLCertificateFile      /etc/ssl/certs/ssl-cert-snakeoil.pem  
  
SSLCertificateKeyFile   /etc/ssl/private/ssl-cert-snakeoil.key
```

Tras realizar los pasos anteriores, prueba a interceptar tramas de conexiones http y https. ¿Qué diferencias ves? Guarda en un archivo una trama de cada tipo para su discusión en clase.

Si te fijas, los dos sitios por defecto tienen configurado el mismo directorio como raíz para los documentos. Qué pasaría si cambias el directorio raíz de uno de los dos. Pruébalo.

Es una mala opción dejar abierta la posibilidad de acceder al mismo sitio mediante una conexión segura y una insegura. Por ello si habilitamos un sitio con HTTPS no deberíamos tener activo el equivalente con HTTP como sucede con los dos por defecto. **Lo correcto sería deshabilitar el default.**

Si deshabilitamos el sitio por defecto podemos ver que ahora, si no especificamos el protocolo, utiliza HTTPS por defecto.

```
sudo a2dissite default  
  
service apache2 reload
```

Estudia el resto del archivo de configuración del sitio por defecto con ssl y explica qué hacen el resto de directivas. Los enlaces a continuación contienen toda la información necesaria.

[https://httpd.apache.org/docs/2.2/mod/mod\\_ssl.html](https://httpd.apache.org/docs/2.2/mod/mod_ssl.html)

<https://httpd.apache.org/docs/2.2/ssl/>

## Creando un sitio virtual con HTTPS

Ahora que hemos probado el sitio seguro por defecto, vamos a crear uno desde cero. Voy a dejar habilitado el sitio por defecto para hacer un ejemplo con un sitio con HTTP y otro con HTTPS.

```
sudo a2dissite default-ssl  
  
sudo a2ensite default  
  
service apache2 reload
```

Como ya hemos visto antes, para poder usar SSL en Apache es necesario tener un certificado. El que se instala para el sitio de ejemplo ya no es válido y tenemos que conseguir uno. Podemos adquirir uno de una CA o crear uno autofirmado. Por motivos evidentes nosotros usaremos esta última opción.

Para obtener un certificado es necesario generar una clave privada y para ello necesitamos un nombre de dominio así que lo primero será **configurar el DNS** correctamente. En mi caso voy a llamarlo [www.con-ssl.es](http://www.con-ssl.es) y habrá que configurarlo como un host virtual por nombre.

Para generar la clave se utiliza el [comando genrsa](#)

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

Se puede generar una contraseña para la clave, pero si se va a utilizar para crear un certificado no es buena idea porque cada vez que el servidor web necesite acceder a la clave habrá que introducir la contraseña. Si no nos importa introducir la clave cada vez que se reinicie el servidor, usaremos la opción

```
openssl genrsa -des3 -out clavepru.key 2048
```

Actualmente se recomiendan longitudes mínimas de clave de 2048 bits.

Lo siguiente será generar una petición para nuestro certificado. Esta petición es la que deberíamos enviar a la CA para obtener un certificado firmado por ellos. Luego esperaríamos a que lo firmaran y nos enviaran el certificado para instalarlo. Nosotros usaremos uno autofirmado.

```
openssl req -new -key clavepru.key -out peticionpru.csr
```

Cuando ejecutamos este comando nos va pidiendo información de la empresa para la que sea el certificado. La vamos rellenando hasta terminar. Los campos que terminan en [ ] no son obligatorios. Si hubiéramos generado la clave con contraseña nos la pediría antes de rellenar la información.

```
You are about to be asked to enter information that will be incorporated  
into your certificate request.
```

```
What you are about to enter is what is called a Distinguished Name or a DN.
```

```
There are quite a few fields but you can leave some blank
```

```
For some fields there will be a default value,
```

```
If you enter '.', the field will be left blank.
```

```
-----
```

```
Country Name (2 letter code) [AU]:ES
```

```
State or Province Name (full name) [Some-State]:Madrid
```

```
Locality Name (eg, city) []:Madrid
```

Organization Name (eg, company) [Internet Widgits Pty Ltd]:SergioCuesta

Organizational Unit Name (eg, section) []:SC

Common Name (e.g. server FQDN or YOUR name) []:Sergio Cuesta

Email Address []:una@con-ssl.es

Please enter the following 'extra' attributes

to be sent with your certificate request

A challenge password []:

An optional company name []:

Para obtener un certificado autofirmado usaremos el comando

```
openssl x509 -req -days 365 -in petitionpru.csr -signkey clavepru.key -out certificadopru.crt
```

Lo que indica que usará el formato X.509 y tendrá una validez de un año. Si todo ha ido bien deberíamos ver algo como lo siguiente

Signature ok

subject=/C=ES/ST=Madrid/L=Madrid/O=SergioCuesta/OU=SC/CN=Sergio Cuesta/emailAddress=una@con-ssl.es

Getting Private key

En muchos sitios verás que en lugar de *crt* se crean archivos con la extensión *pem*. En teoría la diferencia es que *crt* solo contiene el certificado mientras que *pem* contiene tanto el certificado como la clave pero en la práctica esto se ignora y da igual usar una que otra.

Ten en cuenta que todos los archivos que se han creado en los pasos anteriores se han generado en el directorio en el que nos encontramos por lo que hay que moverlos a los sitios adecuados. La petición no es necesaria.

```
sudo mv clavepru.key /etc/ssl/private/  
  
sudo mv certificadopru.crt /etc/ssl/certs/
```

Crearemos un directorio para el contenido del sitio seguro

```
mkdir /var/www/con-ssl/
```

Procedemos a crear el sitio virtual por nombre

```
<IfModule mod_ssl.c>  
  
NameVirtualHost 192.168.1.36:443  
  
<VirtualHost 192.168.1.36:443>  
  
    ServerName con-ssl.es  
  
    ServerAlias www.con-ssl.es
```



```
ServerAdmin alguien@con-ssl.es

DocumentRoot /var/www/con-ssl


<Directory /var/www/con-ssl>

    DirectoryIndex index.html

    Options -Indexes

    AllowOverride None

    Order allow,deny

    allow from all

</Directory>


ErrorLog ${APACHE_LOG_DIR}/error_con_ssl.log

LogLevel warn

CustomLog ${APACHE_LOG_DIR}/con_ssl_access.log combined
```

```
# Esta es la parte de SSL

SSLEngine on

SSLCertificateFile /etc/ssl/certs/certificadopru.crt

SSLCertificateKeyFile /etc/ssl/private/clavepru.key

# Recuerda que lo siguiente es para mantener la compatibilidad con ciertas
# versiones de Microsoft Internet Explorer

BrowserMatch "MSIE [2-6]" \

    nokeepalive ssl-unclean-shutdown \

    downgrade-1.0 force-response-1.0

BrowserMatch "MSIE [17-9]" ssl-unclean-shutdown

</VirtualHost>
```

```
</IfModule>
```

Habilitamos el sitio y recargamos Apache

```
sudo a2ensite con-ssl.es  
  
service apache2 reload
```

Recuerda que el nombre que tenemos asociado en el DNS va a la dirección IP de la máquina así que **tendremos que usar** <https://www.con.ssl.es> para acceder al sitio seguro. Si usamos HTTP va al sitio por defecto. Para evitar esto deberíamos añadir otra entrada al DNS para que vaya al sitio sin SSL (puerto 80) y modificar la configuración del host virtual para que responda a las peticiones a ese otro nombre de dominio en lugar de a todo (\*)

Si los hemos adquirido los certificados a través de una CA de confianza dejaría de aparecer el aviso cuando un cliente se conecta. Evidentemente para un sitio profesional es más que recomendable.

Para casos en los que estemos involucrados en el despliegue en una empresa muy grande o por ejemplo una universidad, puede interesarnos crear **nuestra propia CA** para uso propio. También es posible crear un sitio en el que **los clientes tengan que acceder mediante un certificado propio** como por ejemplo en la Agencia Tributaria. Ambas cosas se pueden hacer usando OpenSSL pero escapa totalmente al contenido del curso.

Crea dos sitios diferentes, uno con HTTPS y el otro sin él. Configura todo correctamente para que al ir a un sitio o a otro use el protocolo adecuado sin necesidad de que el usuario lo especifique en la barra de direcciones del navegador web.

## Despliegue de aplicaciones sobre servidores web y Empaquetado de aplicaciones web.

Estos dos conceptos no los vamos a estudiar aquí. Las aplicaciones en servidores web suelen utilizar al menos una base de datos y las que se despliegan en Apache se están utilizando constantemente en el módulo de “Desarrollo Web en Entorno Servidor”. El empaquetado de aplicaciones es un concepto que está más relacionado con Tomcat por lo que lo veremos en el tema siguiente.

En los casos en los que no instalemos nuestro propio servidor sino que contratemos un hosting, debemos considerar que tenga las características que necesitemos (por ejemplo que permita o proporcione el uso de MySQL y PHP). Generalmente accedemos a nuestra estructura de carpetas mediante FTP y a la base de datos mediante algún cliente.

## Tema 3: Administración de servidores de aplicaciones

Una aplicación web es aquella que funciona a través de una red. Amazon, Google, Facebook o Twitter son ejemplos de aplicaciones web. Una aplicación web funciona en HTTP sobre TCP/IP y los clientes acceden a ella mediante un navegador web. Generalmente es una aplicación que se ajusta al modelo cliente servidor y consta de tres capas ya que hace uso de una base de datos.

El término Servidor de Aplicaciones está asociado históricamente a la plataforma Java Enterprise Edition aunque en la actualidad engloba un conjunto de lenguajes y características mucho más amplio.

Un servidor de aplicaciones es una plataforma que provee aplicaciones software proporcionando servicios como seguridad, transacciones, acceso a datos, etc.

Existen muchos servidores de aplicaciones, siendo los más frecuentes *WebLogic Application Server* de Oracle, *WebSphere Application Server* de IBM, *Tomcat* de Apache. Muy relacionado con éste último está *GlassFish*, que fue desarrollado por Sun Microsystems y actualmente pertenece a Oracle.

Nosotros veremos *Tomcat* aunque *GlassFish* es más completo. ¿Por qué existen dos basados en lo mismo? *Tomcat* solo soporta los **contenedores web** de la plataforma J2EE mientras que *GlassFish* soporta todos los tipos por lo es realmente un servidor de aplicaciones en un sentido más completo.

Vamos a ver *Tomcat* por su simplicidad, facilidad de uso y configuración, bajo consumo de recursos y alto rendimiento y por que es muy utilizado, pero en este caso no existe un paralelismo con lo que veíamos con Apache y el mercado está mucho más fragmento. Es muy probable que *Tomcat* vaya dejando paso a *GlassFish*. Para pequeñas aplicaciones se sigue recomendando *Tomcat*, pero si necesitamos J2EE al completo tendremos que buscar otra opción.

Con el tiempo Java ha ido perdiendo pujanza como lenguaje de programación para la web. Últimamente se ha ido quedando para aplicaciones grandes por características como el balanceo de carga, fácil escalabilidad o las posibilidades de comunicación entre sistemas de backend. Sin embargo se considera muy pesada para las aplicaciones más pequeñas y está siendo rápidamente sustituido por lenguajes como PHP, Ruby o Python. Un ejemplo de esto es Twitter que fue desarrollado con Ruby on Rails pero cuando vieron las dificultades que tenían para escalar la aplicación la migraron a Java.

Por si fuera poco los últimos problemas de seguridad de Java han llevado a las grandes compañías de Sistemas Operativos y de desarrollo de navegadores web a recomendar deshabilitar Java de los navegadores. Incluso “Homeland Security” un organismo gubernamental americano ha hecho esta recomendación sugiriendo que se deshabilite Java de todos los navegadores web excepto donde sea necesario. (Diciembre 2012 - Enero 2013). Oracle está intentando reaccionar sacando parches, pero por ejemplo el último (14/01/2013) lo único que hace es aumentar el nivel de seguridad por defecto y dejar en manos del usuario la decisión de ejecutar o no código no firmado.

¿Puede ser esto el fin de Java en la web? Difícilmente. Antes o después Oracle reaccionará y hay demasiadas aplicaciones enormes realizadas en Java para que desaparezca del mapa en los próximos años. Lo que sí es posible es que deje de usarse como lenguaje de desarrollo de aplicaciones web de tamaño pequeño o medio.

En mi opinión Java quedará más para el *backend* de las aplicaciones mientras que el *frontend* será desarrollado en lenguajes más ligeros. La integración de Tomcat con Apache va en esta línea.

## Arquitectura

Existen varias versiones de Apache Tomcat. La diferencia entre unas versiones y otras es la versión de Servlets y JSPs que pueden manejar. En [este enlace se pueden consultar las diferentes versiones](#), para qué especificaciones sirven y qué versión de Java es necesaria para su uso.

Apache Tomcat utiliza una estructura en la que varios componentes se organizan de manera jerárquica. Estos componentes son *Server*, *Service*, *Engine*, *Host*, *Connector* y *Context*.

Lee la página de la [documentación de Tomcat que explica estos conceptos](#). Entiende muy someramente qué hace cada uno (ya que luego los veremos en más profundidad) y cuál es la relación entre unos y otros. ¿Cuáles tendremos que modificar durante nuestro trabajo? Realizar un dibujo puede ayudarte en la tarea. Posteriormente discute en clase los conceptos aprendidos.

Cuando hayamos instalado Tomcat podremos ver ejemplos de todos estos elementos. Excepto Context, pueden consultarse todos en

```
sudo gedit /etc/tomcat7/server.xml
```

Echa un vistazo al archivo `/etc/tomcat7/server.xml`

## La estructura de directorios de Tomcat

Los directorios que se incluyen en una instalación de Tomcat se ubican generalmente en el directorio donde hemos descomprimido Tomcat. Sin embargo, en esta instalación están definidos en dos ubicaciones: `/usr/share/tomcat7` y `/var/lib/tomcat7`. Sin embargo hay otro directorio afectado en la instalación `/etc/tomcat7`

Los directorios comunes son:

- **bin:** contiene los binarios y scripts de inicio de Tomcat.
- **conf:** la configuración global de Tomcat. En esta instalación es un enlace simbólico a `/etc/tomcat7`. Tiene algunos archivos que merece la pena destacar.
  - **catalina.policy:** Este archivo contiene la política de seguridad relacionada con Java e impide que los Servlets o JSPs la sobrescriban por motivos de seguridad. En esta instalación se encuentra en `policy.d/03catalina.policy`
  - **catalina.properties:** Contiene los archivos .JAR que no pueden sobrescribirse por motivos de seguridad y otros de uso común.
  - **context.xml:** El archivo de contexto común a todas las aplicaciones. Se utiliza principalmente para informar de dónde se puede encontrar el archivo `web.xml` de las propias aplicaciones. Contiene la configuración que será común a todos los elementos Context.
  - **logging.properties:** Establece las políticas generales para el registro de actividad del servidor, aplicaciones o paquetes.
  - **server.xml:** ya hemos visto que es el fichero principal de configuración de Tomcat y que tiene mucho que ver con su arquitectura.
  - **tomcat-users.xml:** Contiene los usuarios, contraseñas y roles usados para el control de acceso. Es el archivo donde se encuentra la información de seguridad para las aplicaciones de administración de Tomcat. Todos los valores son por defecto así que deben cambiarse en caso de “descomentar” las líneas.
  - **web.xml:** Un descriptor de despliegue por defecto con la configuración compartida por todas las aplicaciones. Es un archivo con directivas de funcionamiento de las aplicaciones.
  - Además de todos estos archivos, existe un subdirectorío para cada motor con un subdirectorío `localhost` donde irá otro archivo de contexto específico para cada aplicación. Este archivo tiene la forma `nombre-de-aplicación.xml`. En este caso el motor es Catalina así que están en `Catalina/localhost`. Puedes ver que hay uno por cada paquete adicional que instalamos, documentación, ejemplos y aplicaciones de administración.
- **lib:** continen todos los .JAR comunes a todas las aplicaciones. Aquí van archivos de Tomcat, APIs de JSPs, etc y en él podemos ubicar archivos comunes a las diferentes aplicaciones como MySQL JDBC.
- **logs:** aquí van los archivos de registro. En esta instalación es un enlace simbólico a `/var/log/tomcat7`
- **temp:** este directorio es opcional. En esta instalación no está creado ni activado por defecto. Se usa para los archivos temporales que necesita Tomcat durante su ejecución.
- **webapps:** aquí se ubican las aplicaciones propiamente dichas. Ahora solo hay una que se denomina ROOT. Podría haber más pero como hemos instalado la documentación, ejemplos y administradores aparte se encuentran en su propio directorio en `/usr/share/tomcat7-admin`, `tomcat7-docs` y `tomcat7-examples`.



- **work:** es un directorio para los archivos en uso, cuando se compilan los JSPs, etc.

Busca estos directorios en la instalación de Tomcat que realizamos en el tema 1.

## Un vistazo más profundo a la arquitectura de Tomcat

El término arquitectura de Tomcat también se refiere a la estructura XML que sigue su organización. Se compone de un conjunto de elementos representados por etiquetas. En la imagen puede verse cómo se relacionan unos con otros. Son los elementos que visteis antes, y vamos a repasarlos.

### Server

Es el primer elemento superior y representa una instancia de Tomcat. Es equivalente al servidor en sí con un puerto asociado. Pueden existir varios en diferentes puertos y a veces se hace para que si una aplicación falla arrastrando al servidor esto no afecte a otras aplicaciones.

Contiene varios *Listeners* que escuchan y responden a eventos.

También usa *GlobalNamingResources* que sirven para permitir a clientes software hechos en Java encontrar objetos y datos mediante su nombre. Por ejemplo aquí se podría indicar un recurso global como una base de datos MySQL.

### Service

El servicio agrupa un contenedor de tipo *Engine* con un conjunto de conectores. El motor suele ser Catalina y los conectores por defecto HTTP y AJP.

### Connector

Los conectores sirven para comunicar las aplicaciones con clientes (por ejemplo un navegador web u otros servidores). Representan el punto donde se reciben las peticiones y se les asigna un puerto IP en el servidor.

## Containers

Tomcat se refiere a Engine, Host, Context, y Cluster como contenedores. El de nivel más alto es Engine y el más bajo Context. Algunos componentes como Realm o Valve pueden ubicarse dentro de un contenedor.

## Engine

El motor procesa las peticiones y es un componente que representa el motor de Servlets Catalina. Examina las cabeceras (por ejemplo de las tramas HTTP) para determinar a host (o virtual host) o context se le debe pasar cada petición.

Cuando Tomcat se utiliza como servidor autónomo se usa el motor por defecto. Cuando Tomcat se usa dando soporte a un servidor web se sobrescribe porque el servidor web ya ha determinado el destino correcto para las peticiones.

Un motor puede contener Hosts que representan un grupo de aplicaciones web o Context que representa a una única aplicación.

Tomcat se puede configurar con un único host o múltiples hosts virtuales como Apache.

## Host

Define un host por defecto o múltiples hosts virtuales en Tomcat. En Tomcat los hosts virtuales se diferencian por nombres de dominio distintos, por ejemplo [www.aplicacion1.es](http://www.aplicacion1.es) y [www.aplicacion2.es](http://www.aplicacion2.es). Cada uno soporta varios Context.

## Context

Este elemento es equivalente a una aplicación web. Hay que informar al motor y al host de la localización de la carpeta raíz de aplicación. También se puede habilitar la recarga dinámica (dynamic reload) para que al modificar alguna clase de la aplicación se modifique en la ejecución. Esta opción carga mucho el servidor por lo que se recomienda para pruebas pero no en producción.

En un contexto también se pueden establecer páginas de error específicas para armonizarlas con la apariencia de la aplicación.

Puede contener parámetros de inicio para establecer control de acceso en la aplicación.

## Cluster

En caso de que tengamos más de un servidor Tomcat atendiendo las peticiones, este elemento nos permite configurarlo. Es capaz de replicar las sesiones y los parámetros de cada Context. Queda muy por encima del contenido del curso.

## Realm

Se puede aplicar al nivel de Engine, Host o Context. Se utiliza para autenticación y autorización de usuarios o grupos. Pueden usarse con archivos de texto, servidores LDAP o bases de datos por ejemplo.

¿Qué es un servidor LDAP?

Si lo aplicamos a nivel de motor, los usuario o grupos tendrán idénticos permisos en todas las aplicaciones a la hora de acceder a objetos o recursos. Si lo aplicamos en un Host todas las aplicaciones los permisos serán para todas las aplicaciones de dicho entorno mientras que si lo aplicamos en un Context se establecerán por cada aplicación. Sin embargo, aunque un usuario tenga los mismos permisos para todas las aplicaciones debe seguir autenticándose en cada una de ellas por separado.

Los permisos se heredan y por eso se da la situación descrita en el párrafo anterior, pero si los permisos de un usuario o grupo se sobrescriben en un entorno de rango inferior los últimos serán los que se apliquen. Es similar a lo que veíamos con ciertas directivas (por ejemplo de directorios) en Apache.

## Valve

Se usa para interceptar peticiones antes de pasárselas a las aplicaciones. Esto nos permite pre-procesar las peticiones para bloquear algunas, registrar accesos, registrar detalles de la conexión (en archivos log), o establecer un único punto de acceso para todas las aplicaciones de un host o para todos los hosts de un servidor. Afectan al tiempo de respuesta a la petición.

Puede establecerse para cualquier contenedor Engine, Host, and Context, y/o Cluster.

Hay un concepto similar en los servlets que se denomina Filtros (Filters).

Aunque en la sección siguiente veremos los archivos de configuración en detalle, echa un vistazo al archivo en `/etc/tomcat/server.xml` para afianzar estos conceptos.

Compara este archivo con el de la instalación de Tomcat que hicimos en el tema 1.

## Configuración básica del servidor de aplicaciones

La configuración de Tomcat se realiza a través de uno o más ficheros XML. Hemos visto por encima los ficheros de configuración en los apartados anteriores. Para este propósito, los principales son: server.xml, context.xml y web.xml.

Tomcat busca estos archivos en el directorio especificado por \$CATALINA\_BASE, en un subdirectorio /conf. En nuestro caso establecimos

```
CATALINA_BASE=/var/lib/tomcat7
```

Pero si visitamos este directorio veremos que el subdirectorio /conf es un enlace simbólico a /etc/tomcat/ que es donde realmente se encuentran los archivos de configuración.

En el caso de que no se especifique \$CATALINA\_BASE se usa \$CATALINA\_HOME que es obligatoria para el arranque de Tomcat.

La configuración de Tomcat incluye infinidad de parámetros y no es el objetivo de este curso ser un experto administrador de él. Vamos a ver por encima los archivos de configuración principales y luego estudiaremos las configuraciones particulares más importantes.

La documentación de Tomcat incluye muchísima más información de todos estos elementos y sus posibles configuraciones.

### server.xml

En el archivo de configuración por defecto podemos ver que se establece un único servicio y una única instancia del servidor. El elemento <Server> tiene la siguiente forma:

```
<Server port="8005" shutdown="SHUTDOWN">
```

Puede sorprendernos que el puerto de Tomcat sea 8080 y sin embargo esté configurado en el puerto 8005. Este puerto es en el que se inicia una instancia del servidor (JVM) y en el que escucha por si llegan señales de apagado (shutdown). Esta señal no se puede mandar desde otra máquina por motivos de seguridad, pero evidentemente se puede ejecutar el comando de apagado desde otra máquina y la señal ya llegará desde la misma en la que esté el servidor.

### <Server>

El elemento <Server> puede contener otros tres:

- <Service>: Un grupo de conectores asociados con un motor. Es necesario al menos uno.
- <Listener>: Clases que tienen que escuchan y manejan eventos que tienen que ver con el ciclo de vida del servidor, por ejemplo después de arrancar, etc.
- <GlobalNamingResources>: Recursos globales que pueden ser usados en esta instancia del servidor por los componentes que los necesiten, por ejemplo una base de datos.

```
<GlobalNamingResources>

  <!-- Editable user database that can also be used by

        UserDatabaseRealm to authenticate users

  -->

  <Resource name="UserDatabase" auth="Container"

        type="org.apache.catalina.UserDatabase"

        description="User database that can be updated and saved"

        factory="org.apache.catalina.users.MemoryUserDatabaseFactory"
```

```
        pathname="conf/tomcat-users.xml" />  
  
</GlobalNamingResources>
```

### <Service>

El propósito de un servicio es agrupar un motor que procese las peticiones con uno o más conectores que gestionen los protocolos de comunicación. El servicio por defecto es el motor Catalina.

```
<Service name="Catalina">
```

Aquí se le ha dado al servicio el nombre del motor, pero no es necesario.

Un <Service> contiene al menos un <Connector> y solo un <Engine> que es obligatorio.

### <Connector>

Este elemento tiene mucho que ver con los dos modos de funcionamiento de Tomcat:

- Como servidor único: En el que Tomcat realiza las funciones de servidor de aplicaciones y de servidor web.
- Como servidor de aplicaciones: En el que Tomcat colabora con un servidor web que hace de *frontend*. El servidor web dirige todas las peticiones de JSPs y Servlets a Tomcat.

En un entorno con acceso público se suele usar la segunda configuración ya que el servidor web está mucho más preparado en términos de seguridad y privacidad.

Los dos conectores más comunes son HTTP y AJP. El segundo es usado para conectar con los servidores en el modo colaborativo (Apache u otros). Ambos pueden funcionar con SSL para mejorar la seguridad.

El puerto por defecto para HTTP es 8080. Se puede cambiar y si por ejemplo Tomcat va a estar en producción como un servidor en solitario podríamos modificarlo al puerto estándar de HTTP, 80.

El conector HTTP está habilitado:

```
<Connector port="8080" protocol="HTTP/1.1"
           connectionTimeout="20000"
           URIEncoding="UTF-8"
           redirectPort="8443" />
```

mientras que el AJP no:

```
<Connector port="8009" protocol="AJP/1.3" redirectPort="8443" />
```

Ninguno de los dos usa SSL por defecto.

### <Engine>

El motor es quien procesa las peticiones realmente. El nombre es el que le damos a la instancia y defaultHost indica a qué host virtual se le pasará una petición en caso de que no se especifique ninguno ya que el mismo motor puede procesar peticiones dirigidas a múltiples hosts virtuales de los especificados en este archivo.

```
<Engine name="Catalina" defaultHost="localhost">
```

Un motor contiene uno o más <Host>, uno o ningún <Context>, uno o ningún <Realm>, multiples <Valve> y <Listener> aunque puede no tener ninguno.

### <Realm>

Es un mecanismo de seguridad que sirve para autenticar usuarios y establecer seguridad a nivel de contenedor (recuerda qué elementos consideraba Tomcat [contenedores](#) según vimos en el apartado arquitectura). La configuración por defecto hace que Tomcat lea los usuarios del archivo tomcat-users.xml pero obviamente sería mejor configurarlo para que se usara una base de datos o servidor LDAP.

```
<!-- Use the LockOutRealm to prevent attempts to guess user passwords
```



```
via a brute-force attack -->

<Realm className="org.apache.catalina.realm.LockOutRealm">

    <!-- This Realm uses the UserDatabase configured in the global JNDI
         resources under the key "UserDatabase". Any edits
         that are performed against this UserDatabase are immediately
         available for use by the Realm. -->

    <Realm className="org.apache.catalina.realm.UserDatabaseRealm"

        resourceName="UserDatabase"/>

</Realm>
```

### <Host>

Este elemento representa un host virtual en Tomcat. La configuración por defecto solo define *localhost*. Si tenemos configurado el servidor DNS con un nombre para nuestro servidor usaremos éste.

El atributo *appBase* establece el directorio raíz de las aplicaciones. Se establece a partir de `<CATALINA_BASE>` si no se indica lo contrario. Por defecto la URL de cada aplicación es la resultante de añadir su directorio raíz a la del servidor. En nuestro ejemplo hemos instalado cuatro aplicaciones docs, examples, host-manager y manager. También se establece *ROOT* que indica la aplicación por defecto si no añadimos otras cada una en su directorio.

El atributo *unpackWARs* indica si este tipo de archivos debe ser descomprimido o no. En caso de no descomprimirlos, la ejecución será un poco más lenta.

*autoDeploy* indica si el despliegue de una aplicación que situemos en el directorio debe ser automático o no.

```
<Host name="localhost" appBase="webapps" unpackWARs="true" autoDeploy="true">
```

Además contiene un <Valve> para registrar los accesos.

```
<Valve className="org.apache.catalina.valves.AccessLogValve" directory="logs"
      prefix="localhost_access_log." suffix=".txt"
      pattern="%h %l %u %t &quot;%r&quot; %s %b" />
```

Investiga y discute en clase qué son los archivos WAR y su relación con los archivos JAR.

Configura un nombre en el DNS para tu servidor y cambia el nombre en <Host>. Prueba que puedes acceder con el nombre a todo.

---

## context.xml

El contexto en Tomcat se puede establecer a mucho niveles y se aplicará el más específico en cada aplicación.

Los descriptores de contexto de administración de cada aplicación se encuentran en \$CATALINA\_BASE/conf/nombre\_motor/nombre\_host

En nuestro caso están en /etc/tomcat/Catalina/localhost

Por ejemplo el de host-manager

```
<Context path="/host-manager"
      docBase="/usr/share/tomcat7-admin/host-manager"
```

```
antiResourceLocking="false" privileged="true" />
```

El *path* indica cómo se accederá a la aplicación en nuestro servidor. El problema para cambiarlo es que habría que modificar todos los archivos xml relacionados para que Tomcat siga encontrando la aplicación.

docBase es el directorio de despliegue de la aplicación.

Los descriptores de contexto específicos de cada aplicación web están en el directorio de cada aplicación en /META-INF

El de la misma aplicación está en /usr/share/tomcat7-admin/host-manager/META-INF

```
<Context antiResourceLocking="false" privileged="true" >

  <!--

    Remove the comment markers from around the Valve below to limit access to

    the host-manager application to clients connecting from localhost

  -->

  <!--

  <Valve className="org.apache.catalina.valves.RemoteAddrValve"

    allow="127\.\d+\.\d+\.\d+|::1|0:0:0:0:0:0:0:1" />

  -->

</Context>
```

Entonces ¿para qué sirve context.xml? contiene los parámetros que vayan a ser comunes a todas las aplicaciones. Si no se sobrescriben en algún contexto más concreto se aplicarán éstos.

```
<Context>

    <WatchedResource>WEB-INF/web.xml</WatchedResource>

</Context>
```

En este caso se especifica qué elemento de cada aplicación debe ser vigilado por si cambia. En ese caso se vuelve a desplegar la aplicación automáticamente.

## web.xml

Cualquier aplicación web en Java debe tener un descriptor de despliegue. Como sucedía con el contexto, puede establecerse varios y se aplicarán las directivas más concretas.

Cada aplicación tiene su propio descriptor y va en /WEB-INF

Por ejemplo /usr/share/tomcat7-admin/host-manager/WEB-INF

Existe uno común a todas las aplicaciones en /etc/tomcat/web.xml

Como el año pasado vimos xml es muy fácil entender un descriptor de despliegue. Abre el primero y estudia su contenido.

El segundo archivo es muchísimo más extenso pero no por eso vamos a dejar de echarle un vistazo. ¿Qué tipo de parámetros se incluyen en este archivo? ¿Crees que es adecuada esta división?

En ambos casos, si no entiendes para qué sirve un parámetro, internet puede ayudarte.

En el punto de [despliegue de aplicaciones](#) veremos algo más de este tema.

## Administrar aplicaciones web

Vamos a administrar las aplicaciones de Tomcat a través del Web Manager. En las dos instalaciones que hemos hecho incluimos este paquete. En este caso cuando [instalamos los paquetes adicionales](#).

Podemos acceder al administrador a través del enlace de la página de inicio o en la dirección <http://localhost:8080/manager/html>

Es en si misma una aplicación (con todas las características que estamos viendo) y de hecho puede administrarse desde la propia aplicación.

El acceso por defecto está deshabilitado, pero ya vimos cómo [modificar el archivo tomcat-users.xml](#) para poder entrar. Existen varios roles en caso de que queramos distribuir el trabajo, pero nos asignamos todos los permisos.

Nada más entrar en el manager vemos una lista de las aplicaciones desplegadas en este servidor.



### Gestor de Aplicaciones Web de Tomcat

Mensaje:	OK
----------	----

Gestor					
<a href="#">Listar Aplicaciones</a>	<a href="#">Ayuda HTML de Gestor</a>	<a href="#">Ayuda de Gestor</a>	<a href="#">Estado de Servidor</a>		

Aplicaciones					
Trayectoria	Versión	Nombre a Mostrar	Ejecutándose	Sesiones	Comandos
/	Ninguno especificado		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

<a href="#">/host-manager</a>	Ninguno especificado	Tomcat Host Manager Application	true	<u>1</u>	<div> Arrancar Parar Recargar Replegar </div> <div> Expirar sesiones sin trabajar ≥ 30 minutos </div>
-------------------------------	----------------------	---------------------------------	------	----------	---

Vemos que aparece un enlace a la aplicación en sí, si se ha especificado una versión, el nombre que se mostrará para la aplicación, si está en ejecución, el número de sesiones que se han establecido (si la aplicación usa sesiones) y unos botones para arrancar, parar, recargar (la aplicación en caso de modificaciones) o replegar que lo que hace es eliminarla del servidor (no estará desplegada).

El apartado de las [sesiones](#) se verá más adelante.

Accede al web manager y prueba a parar los ejemplos y la documentación. ¿Puedes conectarte a ellos?

La parte de desplegar las aplicaciones la veremos en el [apartado correspondiente](#).

Lo siguiente que nos encontramos es

## Diagnósticos

Revisa a ver si una aplicación web ha causado fallos de memoria al parar, recargar o replegarse.

Halla fallos de memoria

Este chequeo de diagnóstico disparará una colección completa de basura. Utilízalo con extremo cuidado en sistemas en producción.

## Información de Servidor

Versión de Tomcat	Versión JVM	Vendedor JVM	Nombre de SO	Versión de SO	Arquitectura de SO	NombreDeMáquina	Dirección IP
Apache Tomcat/7.0.26	1.7.0_09-b30	Oracle Corporation	Linux	3.2.0-31-generic-pae	i386	sergio-VirtualBox	127.0.1.1

Los diagnósticos sirven para ver si una aplicación está fallando, por ahora solo en el temas de uso de memoria. Si pinchamos se recarga la página y en la parte superior nos indica el resultado de la prueba.

También podemos ver un cuadro resumen con información del servidor. Esta información puede ampliarse al pinchar en “Estado Completo del Servidor” en la parte superior de la página.

Por supuesto esta administración puede ser mucho más profunda a través de los archivos de configuración vistos en este tema.

## Estado de Servidor

Gestor			
<a href="#">Listar Aplicaciones</a>	<a href="#">Ayuda HTML de Gestor</a>	<a href="#">Ayuda de Gestor</a>	<a href="#">Estado Completo de Servidor</a>

Información de Servidor							
Versión de Tomcat	Versión JVM	Vendedor JVM	Nombre de SO	Versión de SO	Arquitectura de SO	NombreDeMáquina	Dirección IP
Apache Tomcat/7.0.26	1.7.0_09-b30	Oracle Corporation	Linux	3.2.0-31-generic-pae	i386	sergio-VirtualBox	127.0.1.1

### JVM

Free memory: 12.98 MB Total memory: 24.23 MB Max memory: 123.75 MB

### "http-bio-8080"

Max threads: 200 Current thread count: 10 Current thread busy: 1  
Max processing time: 1154 ms Processing time: 3.876 s Request count: 71 Error count: 8 Bytes received: 0.00 MB Bytes sent: 0.44 MB

Stage	Time	B Sent	B Recv	Client	VHost	Request
R	?	?	?	?	?	?
S	13 ms	0 KB	0 KB	127.0.0.1	localhost	GET /manager/status?org.apache.catalina.filters.CSRF_NONCE=08754385D8DF3136E2F479C673CE02F4 HTTP/1.1

P: Parse and prepare request S: Service F: Finishing R: Ready K: Keepalive



Existen otros dos métodos de administración de Tomcat que dejamos como temas de ampliación por parte del propio alumno en caso de que lo crea conveniente: Ant y administración mediante peticiones HTTP.

---

### La estructura de archivos y directorios de una Aplicación Web

---

Una aplicación web típicamente va dentro de un directorio (puede ser un WAR como veremos más adelante) y tiene asociada una estructura de directorios común a todas:

- Raíz del directorio de la aplicación y directorios no especificados en la siguiente lista: Todo lo que no esté en los directorios WEB-INF Y META-INF son recursos públicos a los que se puede acceder a partir de la URL adecuada.
- **WEB-INF:** a parte del descriptor de despliegue (web.xml) contiene:
- **classes:** donde se ubican los archivos .class y otros recursos.
- **lib:** donde se colocan las librerías .jar específicas de esta aplicación.
- **META-INF:** contiene el archivo de contexto context.xml. También puede hacer aquí un archivo MANIFEST.MF que lista las bibliotecas JAR que deben estar disponibles para el funcionamiento de la aplicación.

## Despliegue de aplicaciones en el servidor de aplicaciones

Aunque en este módulo no se incluye cómo desarrollar un Servlet o un JSP, voy a dar unas pequeñas indicaciones. En este caso usaré Netbeans aunque Eclipse incluye las mismas funcionalidades y nos permite integrar desarrollo en muchos lenguajes.

Lo primero será instalar [Netbeans desde la página oficial](#). Vemos que tenemos una opción en la que incluso están incluidos Tomcat y GlassFish. Esta es la recomendada para poder ir haciendo las pruebas. Ten en cuenta que Tomcat no se instala por defecto así que habrá que pulsar en personalizar durante la instalación (como indica la imagen) y seleccionarlo. Incluso aunque lo hagamos de prueba (por supuesto en un caso real la máquina de desarrollo y el servidor nunca deben ser el mismo) es preferible instalar Netbeans en la máquina anfitrión o en otra virtual ya que estamos activando otro servidor Tomcat.

Si ya tenemos Netbeans instalado (y un servidor de aplicaciones) y no queremos la versión de arriba, seguimos estas [instrucciones para añadir y configurar el framework de red](#).

El tipo de proyecto que se debe crear es del grupo Java Web. Por defecto ya nos crea un JSP de tipo “Hola Mundo”. Un Servlet es una clase de java preparada para funcionar en la web. En este enlace explican brevemente [cómo programar Servlets](#). Además recuerda que hemos instalado ejemplos en los que puedes consultar el código para ver qué hacen y cómo lo hacen.



Después de ejecutar o limpiar y contruir (Clean & Build) podremos ir a la carpeta del proyecto y en un subdirectorio “dist” habrá un archivo WAR que nos servirá para el despliegue.

En el site hay un **ejemplo 0310\_JSPUno.war** que uso para estos ejemplos y puedes usar para tus prácticas. Cuando lo descargues recuerda que un WAR es un archivo empaquetado y que se puede descomprimir para obtener los archivos y carpetas originales.

Aunque no es el objetivo del curso vamos a ver el ejemplo del site y algunos de los instalados previamente. Los iremos estudiando **uno a uno** según diga el profesor y comentando en clase. Solo veremos los más sencillos. Se deja como ampliación para el alumno que lo desee ver el resto.

---

## Despliegue manual

Existen dos formas de llevar a cabo el despliegue manual de una aplicación en nuestro servidor de aplicaciones, pero son bastante similares. La primera es usando la carpeta con todos los elementos del proyecto y la segunda es con el archivo WAR.

Para **desplegar la carpeta** solo hay que copiarla en el directorio webapps de la ubicación \$CATALINA\_BASE/webapps que en nuestro caso es /var/lib/tomcat7/webapps

En el ejemplo he descomprimido los contenidos del WAR en una carpeta “Prueba” que es la que copio a webapps.

Recuerda que en nuestro fichero server.xml aparecía

```
<Host name="localhost" appBase="webapps" unpackWARs="true" autoDeploy="true">
```

Si la opción autoDeploy estuviera a false habría que reiniciar el servidor, pero en nuestro caso no hace falta así que podemos ir al navegador web y acceder a nuestra aplicación en la dirección <http://localhost:8080/Prueba/>

**Desplegar el archivo WAR** es igual pero ponemos el archivo directamente en la carpeta webapps. Ten en cuenta que ahora la ruta de la aplicación la determina el nombre del archivo WAR así que será [http://localhost:8080/0310\\_JSPUno/](http://localhost:8080/0310_JSPUno/)

---

## Estableciendo nuestra aplicación como la principal para el servidor

---

La solución anterior está bien para aplicaciones secundarias, pero ahora queremos que sea la principal en lugar de la página de inicio de Tomcat. Esto también se puede hacer de varias formas aunque lo primero sería eliminar el contenido de la carpeta ROOT de webapps. Si vas a hacerlo, ten en cuenta que en nuestro caso es mejor cambiar el nombre o mover el contenido a otro sitio por si luego queremos recuperar el estado original.

Luego las opciones serían:

1. Copiar el contenido de la carpeta del proyecto (no el proyecto en sí) a ROOT.
2. Cambiar el nombre al archivo WAR por ROOT.war y colocarlo en webapps para que lo despliegue Tomcat automáticamente en la carpeta ROOT
3. Crear un descriptor de contexto de la aplicación en la carpeta \$CATALINA\_BASE/conf/Catalina/localhost. Recuerda que esta ruta puede variar dependiendo de tu configuración, motor y host. Este descriptor se llamará ROOT.xml y en él habrá que escribir

```
<?xml version="1.0" encoding="UTF-8"?>

<Context docBase="/usr/share/Prueba" />
```

Suponiendo que hubiéramos ubicado nuestra aplicación en el directorio /usr/share/Prueba

Ten en cuenta que no hay que especificar path porque queremos que sea el que se cargue al acceder directamente a la dirección del servidor.

Si usamos esta opción reiniciamos Tomcat para que vuelva a cargar los archivos de configuración.

---

## Despliegue con Tomcat Web Manager

---

Recuerda que cuando vimos el Web Manager nos saltamos una parte que permitía desplegar aplicaciones.

Desplegar	
<b>Desplegar directorio o archivo WAR localizado en servidor</b>	
Trayectoria de Contexto (opcional):	<input type="text"/>
URL de archivo de Configuración XML:	<input type="text"/>
URL de WAR o Directorio:	<input type="text"/>
<input type="button" value="Desplegar"/>	
<b>Archivo WAR a desplegar</b>	
Seleccione archivo WAR a cargar	<input type="text"/>
<input type="button" value="Examinar..."/>	
<input type="button" value="Desplegar"/>	

La primera opción es para desplegar una aplicación que está en otro servidor.

Solo vamos a ver la opción de abajo. Esta opción es equivalente colocar un WAR en webapps. Al pinchar el botón examinar tendremos que ubicar el WAR deseado y pulsamos sobre desplegar para que active la aplicación. Nos aparecerá en la lista de aplicaciones y podremos gestionarla como a cualquier otra. Si pulsamos en “Replegar” se eliminará.

## Autenticación de usuarios

Los métodos para autenticar usuarios en Tomcat son los mismos que en Apache: Basic, Digest, Formularios HTML y Certificados Digitales. Los dos primeros ya vimos que tenían un problema si se aplicaban sobre conexiones no seguras. En el caso de formularios se establece una página (por ejemplo un JSP) que sirva de elemento donde se realizará la autenticación. En los ejemplos que hemos instalado se utiliza así que **después de ver este apartado** estarás preparado para entender cómo funciona.

Por ejemplo el descriptor de despliegue del web manager utiliza autenticación Basic y se le asigna un nombre al dominio.

```
<!-- Define the Login Configuration for this Application -->

<login-config>

    <auth-method>BASIC</auth-method>

    <realm-name>Tomcat Manager Application</realm-name>

</login-config>
```

En Tomcat cada usuario tiene asociado uno o más roles. Son estos roles los que se comprueban para conectarse y no el usuario en si. En el mismo archivo podemos ver la configuración de autenticación completa con todos los roles definidos. He eliminado gran parte del código dejando solo un ejemplo de cada tipo.

En el siguiente ejemplo es importante destacar que estamos declarando:

- <Security-constraint> una restricción de seguridad en la que se especifica un recurso o colección de recursos. En este caso todo lo que haya en el directorio *html/* y se le asigna un nombre. Además lleva una restricción de autorización donde se establecen uno o más roles que podrán acceder.

- <login-config> donde se especifica qué método se utilizará para llevar a cabo el control de acceso. Es el trozo que vimos antes.
- <security-role> por último se especifican los roles que se nombran en la seguridad de esta aplicación. Date cuenta de que deben estar definidos con el mismo nombre en el archivo, base de datos, etc que utilizemos para la autenticación.

```
<!-- Define a Security Constraint on this Application -->

<!-- NOTE:  None of these roles are present in the default users file -->

<security-constraint>

    <web-resource-collection>

        <web-resource-name>HTML Manager interface (for humans)</web-resource-name>

        <url-pattern>/html/*</url-pattern>

    </web-resource-collection>

    <auth-constraint>

        <role-name>manager-gui</role-name>

    </auth-constraint>

</security-constraint>

<!-- Define the Login Configuration for this Application -->
```

```
<login-config>

    <auth-method>BASIC</auth-method>

    <realm-name>Tomcat Manager Application</realm-name>

</login-config>

<!-- Security roles referenced by this web application -->

<security-role>

    <description>

        The role that is required to access the HTML Manager pages

    </description>

    <role-name>manager-gui</role-name>

</security-role>
```

Sin embargo si nos vamos a ver la estructura de carpetas de la aplicación “manager” vemos que no hay un subdirectorio *html/* por ningún lado. Para entender esto hay que tener en cuenta otras dos partes del archivo. Nuevamente me centro en un ejemplo porque hay varios como este.

```
<servlet-mapping>

    <servlet-name>HTMLManager</servlet-name>
```



```
<url-pattern>/html/*</url-pattern>  
  
</servlet-mapping>
```

Aquí ya vemos el mismo patrón que estamos buscando. Un mapeado ya suena a una asociación. Se asocia un nombre con un patrón URL. Si nos conectamos al manager con el navegador web veremos que se añade este *html/* a las direcciones. Si te das cuenta, al pinchar en el enlace “Estado del Servidor” se cambia *html* por *status*. También encontramos un mapeo a este añadido a las URL y si nos damos cuenta tiene su propia configuración de seguridad y roles asociados.

```
<servlet-mapping>  
  
  <servlet-name>Status</servlet-name>  
  
  <url-pattern>/status/*</url-pattern>  
  
</servlet-mapping>
```

¿Qué roles de seguridad tienen acceso a los recursos de status?

Pero todavía nos falta ver qué es ese nombre que se asocia a un patrón URL. Si miramos más arriba en el archivo

```
<servlet>  
  
  <servlet-name>HTMLManager</servlet-name>  
  
  <servlet-class>org.apache.catalina.manager.HTMLManagerServlet</servlet-class>  
  
  <init-param>
```

```
<param-name>debug</param-name>

<param-value>2</param-value>

</init-param>

<!-- Uncomment this to show proxy sessions from the Backup manager in the
      sessions list for an application

<init-param>

  <param-name>showProxySessions</param-name>

  <param-value>true</param-value>

</init-param>

-->

<multipart-config>

  <!-- 50MB max -->

  <max-file-size>52428800</max-file-size>

  <max-request-size>52428800</max-request-size>

  <file-size-threshold>0</file-size-threshold>
```

```
</multipart-config>

</servlet>
```

Donde lo más importante por ahora son las primeras líneas que asocian un nombre a una clase de Java. Podemos ver otros dos factores (sin entrar mucho más en detalle porque no es el objetivo de este punto) que nos ayudarán a comprender esto. En `/usr/share/tomcat7-admin/manager/index.jsp` podemos ver lo siguiente:

```
<% response.sendRedirect(response.encodeRedirectURL(request.getContextPath() + "/html")); %>
```

Que incluso con los conocimientos que tenemos vemos que redirecciona las peticiones al mismo path pero con `/html` al final. La última duda es ¿dónde está el servlet al que se asocia el nombre `HTMLManager`? No aparece por la estructura de carpetas de la aplicación. Debemos recordar que hay un directorio de bibliotecas compartidas: `/usr/share/tomcat7/lib` y en él hay un archivo que se llama `catalina.jar` (como parte del nombre completo de la clase). Realmente es un enlace simbólico al archivo `../java/tomcat-catalina-7.0.26.jar` o lo que es lo mismo `/usr/share/./java/tomcat-catalina-7.0.26.jar`

Si examinamos este archivo encontraremos el dichoso servlet.

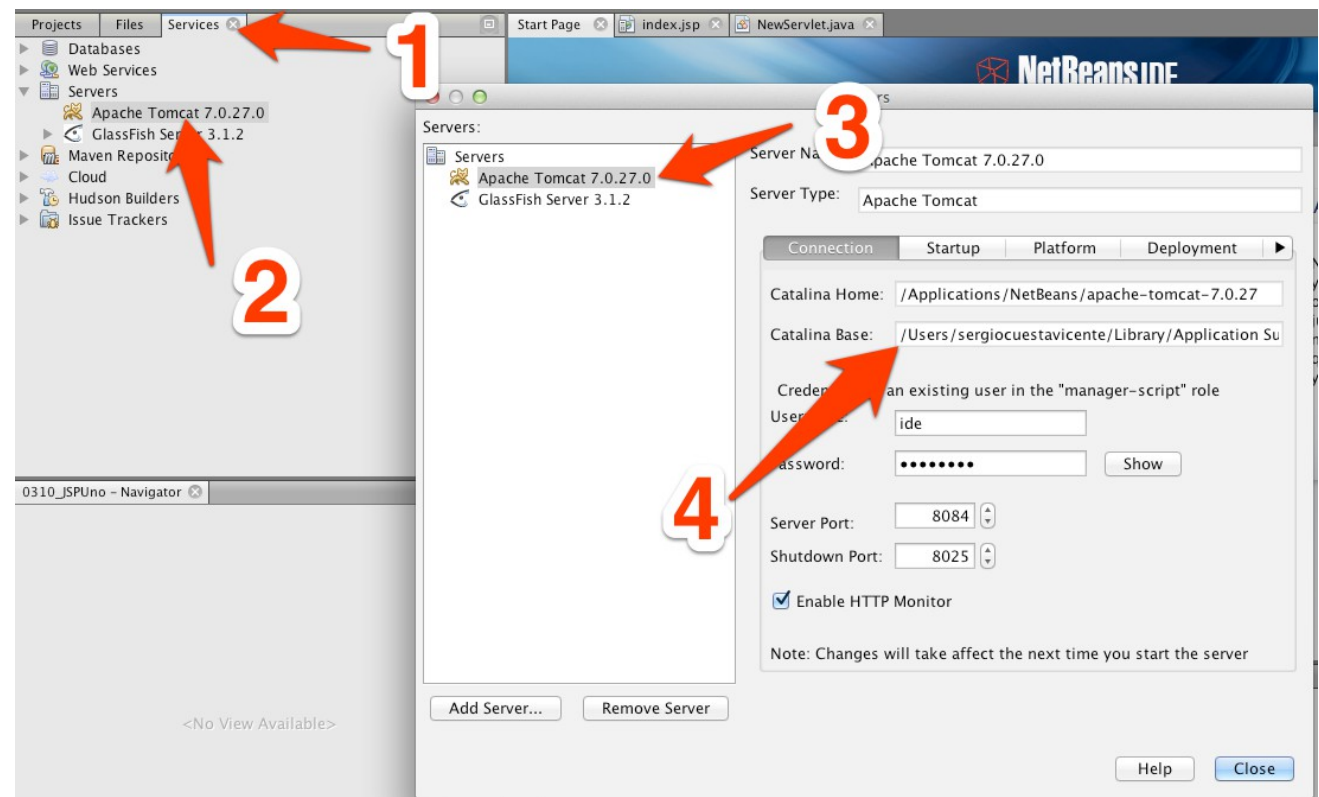
Todo este funcionamiento se basa en el elemento `<Realm>` del que ya hemos hablado. Recuerda que podía ir dentro de los elementos `<Engine>`, `<Host>`, `<Context>` o `<Cluster>` y como en otros casos esto indicará a qué se aplica. Posteriormente [veremos como configurar diferentes Realms](#), pero por ahora nos vale con saber que en el caso por defecto se basa en el archivo `tomcat-users.xml`

## Ejemplos de autenticación

Vamos a ver algunos ejemplos de autenticación de los métodos mencionados. El caso de certificados digitales evidentemente no tendrá sentido hasta que veamos la integración con SSL. Ya conocemos los problemas de seguridad de HTTP.

Ya hemos visto como desplegar aplicaciones hechas con Netbeans (con Eclipse es muy similar), por lo que voy a realizar y probar los ejemplos directamente en Netbeans. Para modificar el archivo tomcat-users.xml hay que buscarlo primero:

1. Selecciona la pestaña “Services”
2. En “Servers” hac clic con el botón derecho sobre “Properties”.
3. Comprueba que es Tomcat y no GlassFish es que está seleccionado.
4. Copia la ruta de CATALINA\_BASE
5. Cierra la ventana.
6. Haz clic en los menús en File > Open File y vete a la ruta de CATALINA\_BASE y al directorio “conf” que hay dentro. Selecciona el archivo tomcat-users.xml
7. Ahora puedes modificar el archivo que se usará. Recuerda que esto tienes que volver a hacerlo en el archivo tomcat-users.xml en el servidor donde despliegues las aplicaciones posteriormente. Este servidor es solo para pruebas.
8. En mi caso voy a dejarlo como se ve a continuación.



```
<user username="ide" password="Ay02ybUt" roles="manager-script,admin"/>

<role rolename="pruebas"/>

<role rolename="otro"/>

<user username="sergio" password="sergio" roles="pruebas"/>

<user username="cuesta" password="cuesta" roles="pruebas,otro"/>

<user username="vicente" password="vicente" roles="pruebas"/>

</tomcat-users>
```

Ten en cuenta que si el servidor ya está iniciado debes reiniciarlo para que cargue la configuración. Para esto haz clic con el botón derecho sobre el servidor Tomcat y pulsa “Restart” o la opción que necesites en cada caso.

## BASIC

Primero vamos a crear el servlet, se hace como cualquier otra clase de Java pero seleccionando Servlet como tipo y luego tienes que marcar la casilla que añade información al web.xml

```
<?xml version="1.0" encoding="UTF-8"?>

<web-app version="3.0" xmlns="http://java.sun.com/xml/ns/javaee"

    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-
app_3_0.xsd">
```

```
<servlet>

    <servlet-name>BasicServlet</servlet-name>

    <servlet-class>basic.BasicServlet</servlet-class>

</servlet>

<servlet-mapping>

    <servlet-name>BasicServlet</servlet-name>

    <url-pattern>/BasicServlet</url-pattern>

</servlet-mapping>

<session-config>

    <session-timeout>

        30

    </session-timeout>

</session-config>

</web-app>
```

Debemos modificar esto para que se corresponda con lo que pone a continuación:

```
<?xml version="1.0" encoding="UTF-8"?>

<web-app version="3.0" xmlns="http://java.sun.com/xml/ns/javaee"

    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-
app_3_0.xsd">

    <servlet>

        <servlet-name>BasicServlet</servlet-name>

        <servlet-class>basic.BasicServlet</servlet-class>

    </servlet>

    <servlet-mapping>

        <servlet-name>BasicServlet</servlet-name>

        <url-pattern>/BasicServlet</url-pattern>

    </servlet-mapping>

    <!-- esto es lo que añadimos para la seguridad básica -->

    <!-- Primero una restricción de seguridad -->
```

```
<security-constraint>

    <web-resource-collection>

        <web-resource-name>El * significa que pedimos autenticación para toda la
aplicación</web-resource-name>

        <url-pattern>/*</url-pattern>

        <http-method>GET</http-method>

        <http-method>POST</http-method>

    </web-resource-collection>

    <auth-constraint>

        <role-name>pruebas</role-name>

    </auth-constraint>

    <user-data-constraint>

        <!-- hay tres tipos CONFIDENTIAL, INTEGRAL y NONE -->

        <transport-guarantee>NONE</transport-guarantee>

    </user-data-constraint>
```



```
</security-constraint>
```

```
<login-config>
```

```
    <auth-method>BASIC</auth-method>
```

```
</login-config>
```

```
<!-- hasta aquí -->
```

```
<session-config>
```

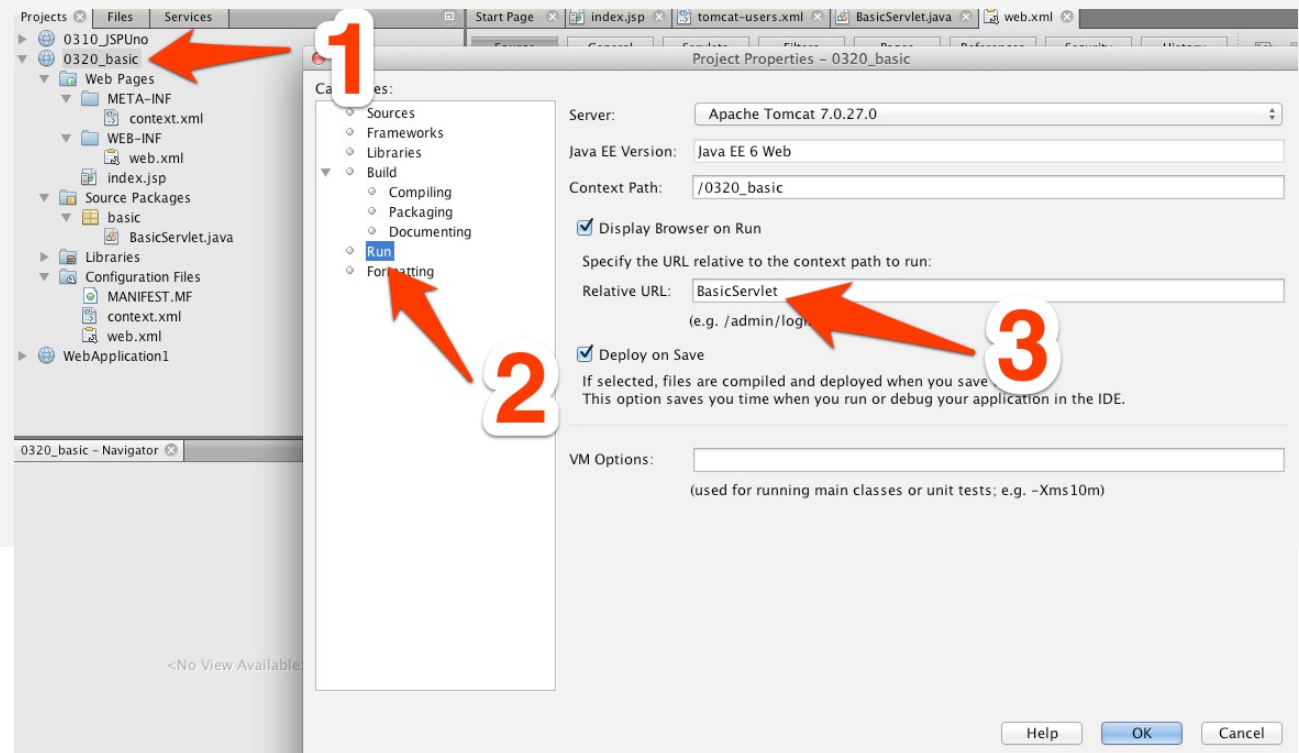
```
    <session-timeout>
```

```
        30
```

```
    </session-timeout>
```

```
</session-config>
```

```
</web-app>
```



Con esto (ten en cuenta que nos ha creado un servlet) ya tendríamos la autenticación básica funcionando, pero podemos modificar el servlet para que nos proporcione cierta información.

<transport-guarantee> tiene que ser NONE, pero cuando habilitemos SSL deberíamos cambiarlo a CONFIDENTIAL para que nos redirija las peticiones HTTP a HTTPS y se transmita la información de esa manera.

Si queremos ejecutar este servlet por defecto en Netbeans debemos hacer:

1. Clic con el botón derecho sobre el proyecto y seleccionar “Properties”
2. Seleccionar “Run”
3. Escribir el nombre del Servlet (sin extensión) en “Relative URL”

Ten en cuenta que si lo desplegamos en nuestro servidor Tomcat de producción la URL sería la correspondiente a la aplicación más el nombre del servlet.

p.ej: <http://localhost:8080/nom-aplicación/nom-servlet>

Puedes ver todo esto y el Servlet modificado que proporciona información sobre la conexión en el **ejemplo 0320\_basic.war** del site.

## DIGEST

La única diferencia entre el uso de BASIC y DIGEST consiste en cambiar esa palabra en web.xml. La única diferencia en el funcionamiento es que la contraseña se codifica, pero no encripta por lo que si alguien la intercepta es prácticamente igual de vulnerable.

Modifica el ejemplo 0320\_basic.war para que use DIGEST y comprueba la diferencia en el resultado.

## FORM

Este método se basa en formularios HTML.

Lo primero que cambia es en el archivo web.xml el elemento <login-config>

```
<login-config>

    <auth-method>FORM</auth-method>

    <!-- CUIDADO, los archivos siguientes deben ir en un subdirectorio para que funcione-->

    <form-login-config>

        <form-login-page>/protegido/login.jsp</form-login-page>

        <form-error-page>/protegido/login-failed.html</form-error-page>

    </form-login-config>

</login-config>
```

Los dos archivos HTML indican en qué página se hará el log-in y a cuál nos redirigirá si falla el intento.

Todo esto también se puede configurar en Netbeans pinchando en la pestaña “Security” cuando estamos viendo el archivo web.xml

Este método se basa en el uso de tres elementos:

1. `j_security_check`: que procesa el formulario.
2. `j_username`: como elemento donde se enviará el nombre de usuario.
3. `j_password`: para enviar la contraseña.

Puedes estudiar el **ejemplo 0330\_form** del site para ver todos estos elementos.

Source General Servlets Filters Pages References Security History Security Roles

### Login Configuration

☒ None  
☐ Digest  
☐ Client Certificate  
☐ Basic  
☐ Form

Form Login Page:  Browse...

Form Error Page:  Browse...

Realm Name:

### Security Roles

Role Name	Description
-----------	-------------

Add... Edit... Remove

### Security Constraints

Add Security Constraint

Para probar debes intentar acceder al elemento FormServlet, a través de la ruta si lo tienes en un Tomcat o configurando Netbeans si lo estás probando con el servidor integrado.

El ejemplo usa sesiones además por lo que puedes ver un uso totalmente básico de ello.

## Dominios de seguridad para la autenticación

Un dominio (Realm) es algún tipo de asociación entre usuarios, contraseñas y los roles asociados. Los roles son algo similar a los grupos de Linux y en Tomcat el acceso no se proporciona a nivel de usuario si no de rol. Un usuario puede tener asociado un número ilimitado de roles.

Existen seis tipos básicos en Tomcat aunque hay otros extendidos y cualquiera puede programar uno que extienda de estos:

1. **JDBCRealm:** La información de usuarios, contraseñas y roles se almacena en una base de datos relacional y se accede mediante un driver JDBC.
2. **DataSourceRealm:** La información también se guarda en una base de datos relacional, pero se accede a ella mediante una fuente con nombre de tipo JNDI JDBC DataSource. JNDI (Java Naming and Directory Interface) es un servicio de Java que permite buscar objetos y datos por nombre.
3. **JNDIRealm:** La información se almacena en un servidor LDAP y se accede con JNDI.
4. **UserDatabaseRealm:** La información se almacena en lo que se conoce como una fuente JNDI de Base de Datos de Usuarios. Lo más habitual es que se apoye en un archivo xml. Por ejemplo de `conf/tomcat-users.xml` aunque no es esta la opción que se usa por defecto sino la siguiente. La información puede actualizarse dinámicamente, pero no se recomienda para grandes sistemas.
5. **MemoryRealm:** La información está almacenada en un archivo XML que se carga en memoria al iniciar Tomcat. Si se modifica el archivo la información no se usa hasta que se reinicie Tomcat. Es el caso que venimos usando con `conf/tomcat-users.xml`. Por supuesto no se recomienda en producción.
6. **JAASRealm:** Se utiliza el framework Java Authentication & Authorization Service (JAAS). Es el más sofisticado de todos.

Nosotros solo veremos algunos. Ya he comentado que hay otros que extienden de estos. Toda esta información y mucha más se puede consultar en [la página de la documentación de Tomcat](#).

**Solo puede haber un Realm funcionando a la vez**, y por lo tanto si pruebas los ejemplos anteriores tras configurar Realms diferentes verás que usa para conectarse el Realm activo.

### MemoryRealm

Es el caso que hemos estado usando por defecto al conectarnos por ejemplo al manager de aplicaciones web. Ya hemos visto que la información se almacena en un fichero XML y que por defecto es `tomcat-users.xml`

Al iniciar el servidor se carga la información de usuarios del archivo en memoria y no se vuelve a actualizar hasta que se reinicie el servidor. Por este motivo y por la baja seguridad que supone tener los usuarios y contraseñas en un fichero de texto plano.

Es el caso que vimos como ejemplo en el punto anterior.

Esto se concreta en los puntos:

1. Gestionar los usuarios en el archivo XML correspondiente.
2. Crear el dominio. Para ello en el archivo de contexto que determinemos (por ejemplo en el que se encuentre en el directorio META-INF específico de cada aplicación) escribiremos lo siguiente:

```
<Context>

  <Realm className="org.apache.catalina.realm.MemoryRealm" />

</Context>
```

He dejado las etiquetas del contexto para evidenciar que va dentro.

3. Crear las restricciones de seguridad del recurso en el descriptor de despliegue correspondiente. Esto se hace con el elemento `<security-constraint>`
4. Determinar el tipo de acceso al recurso. `<login-config>`
5. Establecer uno o más roles que puedan acceder al elemento. `<security-role>`

### UserDatabaseRealm

Este tipo es una pequeña modificación del anterior que nos permite simular que estamos realizando la autenticación con una base de datos. La única diferencia con el anterior es que se realiza a través de un recurso JNDI.

**IMPORTANTE:** Realmente “menti” en el punto anterior porque el caso que se utiliza en el servidor de Tomcat según lo hemos instalado es éste. Pero al ser una extensión del anterior he preferido separar la explicación en dos partes.

Para verlo tenemos que ir al archivo server.xml

```
<!-- Global JNDI resources

    Documentation at /docs/jndi-resources-howto.html

-->

<GlobalNamingResources>

    <!-- Editable user database that can also be used by

        UserDatabaseRealm to authenticate users

    -->

    <Resource name="UserDatabase" auth="Container"

        type="org.apache.catalina.UserDatabase"

        description="User database that can be updated and saved"

        factory="org.apache.catalina.users.MemoryUserDatabaseFactory"

        pathname="conf/tomcat-users.xml" />

</GlobalNamingResources>
```

Si nos damos cuenta se define un recurso global de tipo *UserDatabase* que se asocia al archivo *tomcat-users.xml*

Además un poco más abajo

```
<!-- Use the LockOutRealm to prevent attempts to guess user passwords
      via a brute-force attack -->

<Realm className="org.apache.catalina.realm.LockOutRealm">

  <!-- This Realm uses the UserDatabase configured in the global JNDI
        resources under the key "UserDatabase". Any edits
        that are performed against this UserDatabase are immediately
        available for use by the Realm. -->

  <Realm className="org.apache.catalina.realm.UserDatabaseRealm"

    resourceName="UserDatabase"/>

</Realm>
```

Vemos que se define el *Realm* (mira el interno primero) que es justo del tipo que estamos comentando en este punto. El *Realm* exterior *LockOutRealm* parece interesante según lo que explican los comentarios ¿no? Es un contenedor para asegurar el *Realm* que utilizemos y se puede ver en [la página de la documentación de Tomcat](#).

Por lo tanto a los puntos de *MemoryRealm* habría que añadir (el primero se añade y el segundo es una modificación):

1. Configurar el recurso JDNI
2. Crear el <Realm> para que sea de tipo UserDatabaseRealm en lugar de MemoryRealm

Utilizando MemoryRealm, establece un tipo de control de acceso básico a una aplicación que despliegues en tu servidor puedes usar el ejemplo del site. Establece un rol cuyo identificador sea tu apellido y al menos un usuario con tu nombre. Prueba el acceso.

Realmente solo puede haber un <Realm> actuando en una aplicación en un momento dado. Esta afirmación implica (debes elegir una de las dos respuestas después de probar):

1. Se utiliza el Realm más concreto como con las directivas de Apache.
- 2 .Hay que eliminar el Realm de server.xml porque si no...

Elimina el control de acceso anterior y realiza lo mismo con UserDatabaseRealm ¿Puedes establecer el control de acceso solo para alguna parte de la documentación? ¿y para un archivo en particular? Al ver la estructura de carpetas de una Aplicación Web decíamos que todo lo que no se ubicara en WEB-INF o en META-INF era público. ¿Afecta a esto?

Crea otro rol en el archivo de usuarios pero no le des acceso a la aplicación de documentación. Comprueba que no tiene acceso pero el otro rol sí.

Recuerda que debes reiniciar el servidor (Tomcat) tras cada cambio.



## JDBCRealm

Este caso necesita una base de datos para funcionar por lo que empezaremos [instalando MySQL](#). Los usuarios ahora se ubicarán en una base de datos con tablas de usuarios, roles y la relación entre ellos. Los nombres de estos elementos se pueden elegir y configurar pero hay unos prácticamente estándar que serán los que usemos.

Tomcat se conecta a bases de datos mediante una tecnología llamada *Java Database Connectivity* (JDBC). JDBC funciona como una capa de abstracción entre las bases de datos y los elementos que la vayan a usar. De esta forma Tomcat se comunica con la API de JDBC en lugar de con la base de datos directamente. Así se consigue no depender de las peculiaridades de cada base de datos. JDBC se comunica con la base de datos a través de un driver que convierte las órdenes directamente al formato específico de cada base de datos.

Ya tenemos instalado MySQL y JDBC que se incluye en Java por lo que el siguiente paso será instalar el driver. Para ello podemos descargar el driver [desde su web](#) y colocarlo en \$CATALINA\_HOME/lib que en nuestro caso está en /usr/share/tomcat7/lib lo que lo hará disponible para todas las aplicaciones de Tomcat o colocarlo en el directorio WEB-INF/lib de una aplicación específica para que esté disponible en ella.

Como alternativa podemos instalarlo para que esté disponible para cualquier aplicación que use la JVM de Java:

```
sudo apt-get install libmysql-java
```

y establecemos el CLASSPATH

```
CLASSPATH=".:/usr/share/java/mysql.jar"
```

**Este último método da problemas porque el uso de classpath ya no está recomendado.**

**Nosotros usaremos el método de colocarlo en las librerías comunes de Tomcat.**

Tras lo que habrá que reiniciar para que se cargue.

Ahora, tendremos que crear la configuración correcta en MySQL para la gestión de usuarios y roles. Para ello usaremos el siguiente script de ejemplo

```
DROP DATABASE IF EXISTS tomcat_realm;
```

```
CREATE DATABASE tomcat_realm;

USE tomcat_realm;

CREATE TABLE tomcat_users (

    user_name varchar(20) NOT NULL PRIMARY KEY,

    password varchar(32) NOT NULL

);

CREATE TABLE tomcat_roles (

    role_name varchar(20) NOT NULL PRIMARY KEY

);

CREATE TABLE tomcat_users_roles (

    user_name varchar(20) NOT NULL,

    role_name varchar(20) NOT NULL,

    PRIMARY KEY (user_name, role_name),

    CONSTRAINT tomcat_users_roles_foreign_key_1 FOREIGN KEY (user_name) REFERENCES tomcat_users (user_name),
```

```
CONSTRAINT tomcat_users_roles_foreign_key_2 FOREIGN KEY (role_name) REFERENCES tomcat_roles
(role_name)

);

INSERT INTO tomcat_users (user_name, password) VALUES ('sergio', 'sergio');

INSERT INTO tomcat_users (user_name, password) VALUES ('maria', 'maria');

INSERT INTO tomcat_roles (role_name) VALUES ('pruebas');

INSERT INTO tomcat_roles (role_name) VALUES ('otro');

INSERT INTO tomcat_users_roles (user_name, role_name) VALUES ('sergio', 'pruebas');

INSERT INTO tomcat_users_roles (user_name, role_name) VALUES ('sergio', 'otro');

INSERT INTO tomcat_users_roles (user_name, role_name) VALUES ('maria', 'pruebas');

COMMIT;
```

La tabla que lista los roles no es necesaria y las claves ajenas tampoco, pero las hemos creado por mantener la corrección.

En mi caso voy a crear este script en un directorio para tenerlo accesible en todo momento. Llamo al script *tomcatusers.sql*

```
mkdir /usr/db-scripts

cd /usr/db-scripts/

gedit tomcatusers.sql
```

Accedemos a MySQL

```
mysql -u root -p
```

dentro de MySQL ejecutamos el script

```
mysql> source /usr/db-scripts/tomcatusers.sql
```

y comprobamos que todo haya ido bien.

```
mysql> select * from tomcat_users;  
  
mysql> select * from tomcat_users_roles;  
  
quit
```

Ahora creamos un usuario para que se conecte el Realm que creamos en Tomcat. Creo un script que se llame *usuarioRealm.sql*

```
USE mysql;  
  
CREATE USER 'accesoRealm'@'localhost' IDENTIFIED BY 'pwdRealm';  
  
GRANT SELECT ON tomcat_realms.* TO accesoRealm@localhost;
```

Y lo ejecuto

```
mysql> source /usr/db-scripts/usuarioRealm.sql
```

En el archivo server.xml modifíco la sección del Realm comentando el que hay e introduciendo el de la base de datos.

```
<!-- Use the LockOutRealm to prevent attempts to guess user passwords
      via a brute-force attack -->

<Realm className="org.apache.catalina.realm.LockOutRealm">

  <!-- This Realm uses the UserDatabase configured in the global JNDI
        resources under the key "UserDatabase". Any edits
        that are performed against this UserDatabase are immediately
        available for use by the Realm. -->

<!--

  <Realm className="org.apache.catalina.realm.UserDatabaseRealm"

    resourceName="UserDatabase"/>

-->

<Realm  className="org.apache.catalina.realm.JDBCRealm"

  driverName="com.mysql.jdbc.Driver"

  connectionURL="jdbc:mysql://localhost:3306/tomcat_realm"

  connectionName="accesoRealm" connectionPassword="pwdRealm"
```

```
        userTable="tomcat_users" userNameCol="user_name" userCredCol="password"

        userRoleTable="tomcat_users_roles" roleNameCol="role_name" />

</Realm>
```

Estudia los parámetros de configuración del Realm y discútelos en clase.

Tras esto reiniciamos Tomcat y si no recibimos ningún error hemos configurado el Realm bien.

```
/etc/init.d/tomcat7 restart
```

Ten en cuenta que el error puede estar en los ficheros de log de Tomcat. Aunque luego veremos más sobre esto, tienes que mirar en el directorio `/var/lib/tomcat7/logs` y buscar el archivo con la fecha de hoy de los que tengan un nombre como `catalina.2013-01-26.log` ten en cuenta que puedes borrar el contenido del log para encontrar más fácilmente los nuevos errores.

El error más habitual es que no encuentre el driver JDBC

```
ene 26, 2013 8:47:38 AM org.apache.catalina.realm.JDBCRealm authenticate SEVERE: Excepción al realizar la
autenticación java.sql.SQLException: com.mysql.jdbc.Driver      at
org.apache.catalina.realm.JDBCRealm.open(JDBCRealm.java:701)      at
org.apache.catalina.realm.JDBCRealm.authenticate(JDBCRealm.java:352)  at
org.apache.catalina.realm.CombinedRealm.authenticate(CombinedRealm.java:146)  at
org.apache.catalina.realm.LockOutRealm.authenticate(LockOutRealm.java:180)  at
org.apache.catalina.authenticator.FormAuthenticator.authenticate(FormAuthenticator.java:295)  at
org.apache.catalina.authenticator.AuthenticatorBase.invoke(AuthenticatorBase.java:450)  at
org.apache.catalina.core.StandardHostValve.invoke(StandardHostValve.java:168)  at
org.apache.catalina.valves.ErrorReportValve.invoke(ErrorReportValve.java:98)  at
```

```
org.apache.catalina.valves.AccessLogValve.invoke (AccessLogValve.java:927) at
org.apache.catalina.core.StandardEngineValve.invoke (StandardEngineValve.java:118) at
org.apache.catalina.connector.CoyoteAdapter.service (CoyoteAdapter.java:407) at
org.apache.coyote.http11.AbstractHttp11Processor.process (AbstractHttp11Processor.java:987) at
org.apache.coyote.AbstractProtocol$AbstractConnectionHandler.process (AbstractProtocol.java:579) at
org.apache.tomcat.util.net.JIoEndpoint$SocketProcessor.run (JIoEndpoint.java:309) at
java.util.concurrent.ThreadPoolExecutor.runWorker (ThreadPoolExecutor.java:1110) at
java.util.concurrent.ThreadPoolExecutor$Worker.run (ThreadPoolExecutor.java:603) at
java.lang.Thread.run (Thread.java:722)
```

**Caused by: java.lang.ClassNotFoundException: com.mysql.jdbc.Driver at**

```
java.net.URLClassLoader$1.run (URLClassLoader.java:366) at
java.net.URLClassLoader$1.run (URLClassLoader.java:355) at
java.security.AccessController.doPrivileged (Native Method) at
java.net.URLClassLoader.findClass (URLClassLoader.java:354) at
java.lang.ClassLoader.loadClass (ClassLoader.java:423) at
java.lang.ClassLoader.loadClass (ClassLoader.java:356) at java.lang.Class.forName0 (Native Method)
at java.lang.Class.forName (Class.java:186) at
org.apache.catalina.realm.JDBCRealm.open (JDBCRealm.java:697) ... 16 more
```

Que se produce porque no has colocado bien el archivo JAR con el driver o no está funcionando bien el CLASSPATH.

Prueba a acceder al web-manager. No puedes porque en la base de datos no hemos asignado el rol correcto al usuario. Para ello he creado otro script *manager.sql*

```
USE tomcat_realm;

INSERT INTO tomcat_roles (role_name) VALUES ('manager-gui');

INSERT INTO tomcat_users_roles (user_name, role_name) VALUES ('sergio', 'manager-gui');
```

```
COMMIT;
```

Prueba a desplegar aplicaciones con control de acceso en el servidor (te pueden servir algunos de los ejemplos del site) y prueba a conectarte con UserDatabaseRealm. Después configura JDBCRealm y vuelve a probar.



## Administración de sesiones. Sesiones persistentes

Tomcat utiliza un Persistent Session Manager para realizar copias de las sesiones de los usuarios en el disco. Esta gestión de sesiones es básica para el funcionamiento de las aplicaciones. Permite realizar varias mejoras sobre las sesiones siendo las principales, el establecimiento de un tiempo máximo de vida para las sesiones inactivas y guardando las sesiones en disco cuando se apaga Tomcat, lo que permite que las sesiones se mantengan durante varias ejecuciones del servidor.

Hay definida una interfaz `org.apache.catalina.Manager` para gestionar las sesiones. Se configura mediante el elemento `<Manager>` en alguno de los archivos de contexto (el global `context.xml` o el mismo en el directorio META-INF de alguna aplicación). Tomcat incorpora dos implementaciones de esta interfaz:

- `org.apache.catalina.session.StandardManager`: que es la implementación que se usa por defecto si no se especifica otra.
- `org.apache.catalina.session.PersistenManager`: que permite guardar las sesiones en una base de datos aparte de en un disco.

No vamos a entrar en muchos detalles sobre estas clases pero se pueden [consultar en la documentación de Tomcat](#).

Si vamos al archivo de contexto

```
gedit /etc/tomcat7/context.xml
```

podemos ver que por defecto viene activado

```
<!-- Uncomment this to disable session persistence across Tomcat restarts -->

<!--

<Manager pathname="" />

-->
```

Vemos que si lo descomentamos NO tendríamos sesiones persistentes aunque reiniciáramos Tomcat.

Podemos conectarnos al web manager de Tomcat <http://localhost:8080/manager/html> y probar a acceder al [http://localhost:8080/0330\\_form/FormServlet](http://localhost:8080/0330_form/FormServlet)

Podemos ver que si abrimos y cerramos el navegador varias veces se reinicia la sesión pero si tenemos abierto el navegador se mantiene para cuando volvamos. Cada vez que cerramos el navegador se suma una sesión al número de sesiones de la lista.

<u>/0330_form</u>	<i>Ninguno especificado</i>		true	2
-------------------	-----------------------------	--	------	---

Si hacemos clic sobre el número de sesiones podremos ir a la pantalla de gestión

## Sessions Administration for /0330\_form

Tips:

- Click on a column to sort.
- To view a session details and/or remove a session attributes, click on its id.

Active HttpSessions informations

[Refresh Sessions list](#) 2 active Sessions

Session Id	Type	Guessed Locale	Guessed User name	Creation Time	Last Accessed Time	Used Time	Inactive Time	TTL
<input type="checkbox"/> <a href="#">646311B5739B2615370DBBAFDE71EC4D</a>	Primary		sergio	2013-01-26 10:14:27	2013-01-26 10:14:31	00:00:04	00:05:39	00:24:20
<input type="checkbox"/> <a href="#">A207B5B451DBC3DF3FB7794089432E64</a>	Primary		maria	2013-01-26 10:14:02	2013-01-26 10:14:05	00:00:03	00:06:05	00:23:54

[Invalidate selected Sessions](#)

[Return to main page](#)

Donde tenemos varias opciones:

- Pinchar en el nombre de una columna para que se ordenen las sesiones por ese criterio.
- Refrescar la lista. Mientras estamos gestionando esto pueden estar cambiando las sesiones por muchos motivos, por ejemplo un usuario se conecta o desconecta.
- Seleccionar una o más sesiones para invalidarlas posteriormente con el botón correspondiente. Esto terminará la sesión y el usuario tendrá que volver a crearla (por ejemplo autenticándose en la aplicación).
- Pinchar en el identificador de la sesión para ver sus detalles.

## Details for Session A207B5B451DBC3DF3FB7794089432E64

Session Id	A207B5B451DBC3DF3FB7794089432E64
Guessed Locale	
Guessed User	maria
Creation Time	2013-01-26 10:14:02
Last Accessed Time	2013-01-26 10:14:05
Session Max Inactive Interval	00:30:00
Used Time	00:00:03
Inactive Time	00:13:32
TTL	00:16:27

[Refresh](#)

0 ATTRIBUTES

Remove Attribute	Attribute name	Attribute value
------------------	----------------	-----------------

[Return to session list](#)

Podemos comprobar que aunque reiniciemos Tomcat o incluso la máquina virtual las sesiones se mantienen. Si miramos el descriptor de despliegue de una aplicación podemos ver que es posible configurar el tiempo que se mantendrán las sesiones inactivas. Si tuviéramos aplicaciones que requieren mucha seguridad conviene bajarlo para que cuando el usuario se ausente se termine la conexión.

```
<session-config>

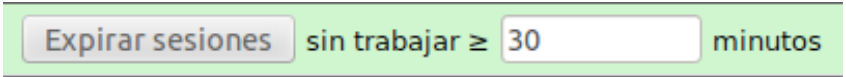
    <session-timeout>

        30

    </session-timeout>
```

```
</session-config>
```

Si miramos la lista de aplicaciones del manager vemos que podemos cerrar las sesiones que lleven inactivas más de un tiempo que pongamos. Para ello se escribe el tiempo y se pulsa el botón. Por supuesto solo tiene sentido en caso de que el tiempo que escribamos sea menor que el establecido por defecto para la aplicación.



Por defecto se utiliza el *StandardManager* que guarda las sesiones en un archivo *SESSIONS.ser* que se ubica en el directorio *work* de cada aplicación. Estos archivos se crean al detener Tomcat y se destruyen cuando se reinicia por lo que si quieres verlos tienes que parar Tomcat. Ten en cuenta que si Tomcat no se apaga correctamente (por ejemplo por un fallo en la corriente) no se guardan las sesiones.

Para configurar más aspectos de las sesiones usaríamos el elemento `<Manager>` de los archivos de contexto

**CUIDADO:** en el siguiente ejemplo, la clase indicada en realidad es la interfaz que tienen que implementar las clases para poder usarse en el elemento manager. Debería ser una clase concreta, por ejemplo una de las dos que pone al principio del punto. Ten en cuenta que el paquete de la clase tampoco es el mismo para la interfaz y las clases concretas.

```
<?xml version="1.0" encoding="UTF-8"?>

<Context antiJARLocking="true" path="/0330_form">

    <Manager

        className = "org.apache.catalina.Manager"

        distributable = "true"

        maxActiveSessions = "25"

        maxInactiveInterval = "1200"
```

```
        sessionIdLength = "16"  
  
    />  
  
</Context>
```

Estudia los atributos comunes a todas las implementaciones de Manager en [la documentación](#). Coméntalos en clase.

¿Qué hace el atributo path de *StandardManager*? ¿Qué valor le darías para deshabilitar las sesiones persistentes?

## Archivos de registro de acceso y filtro de solicitudes

Como cualquier servidor, en Tomcat es muy importante registrar algunos tipos de actividad. En el directorio `/etc/tomcat7` podemos ver dos archivos que hacen referencia a estos registros. El primero es `logging.properties` y el segundo `server.xml`.

Ambos hacen referencia al directorio `$CATALINA_BASE/logs` que en nuestra instalación se encuentra como enlace simbólico en `/var/lib/tomcat7/logs`

Es una buena práctica revisar y borrar los archivos de registro periódicamente ya que pueden crecer mucho y acabar afectando al rendimiento del servidor.

En ese directorio podemos ver cuatro tipos de archivos de registro:

1. `catalina.fecha.log`: Estos archivos guardan la actividad del motor durante un día determinado.
2. `catalina.out`: es un compendio de los anteriores.
3. `localhost.fecha.log`: guardan la actividad del sitio.
4. `localhost_access_log.fecha.txt`: son registros de acceso a las aplicaciones del servidor durante un día. Son los que trataremos aquí.

Este último tipo se aparece configurado en el archivo `server.xml` dentro del elemento `<Host>`

```
<!-- Access log processes all example.

Documentation at: /docs/config/valve.html

Note: The pattern used is equivalent to using pattern="common" -->

<Valve className="org.apache.catalina.valves.AccessLogValve" directory="logs"

        prefix="localhost_access_log." suffix=".txt"
```

```
pattern="%h %l %u %t &quot;%r&quot; %s %b" />
```

y como podemos ver nos refiere a una página de la documentación que hemos instalado pero que también podemos ver [en la página de Apache](#).

Tenemos que darnos cuenta de que realmente va asociado a un elemento <Valve> y muy relacionado con él hay otro que se llama <Filter>

Estas dos tecnologías sirven para interceptar las peticiones y respuestas de HTTP y pre procesarlas para realizar algún tipo de acción antes de que sigan su camino. Una gran ventaja es que no dependen de cada aplicación sino que pueden implementarse para todo el sitio. Una gran diferencia entre ellas es que <Valve> es un desarrollo asociado a Tomcat mientras que <Filter> pertenece a la API de Servlets.

Como ya vimos se pueden colocar en <Engine>, <Host> o <Context> afectando al entorno concreto según corresponda.

## Válvulas

Se puede consultar su uso en la [página correspondiente de la configuración](#), pero se puede apreciar que es muy extensa.

Uno puede programar su propia válvula o filtro partiendo de la [interfaz Valve](#) o de alguna de las clases [del paquete valves](#). ValveBase está espacialmente indicada para ello. De hecho muchas de las otras clases del paquete son *final* por lo que no podremos heredar de ellas. Esto queda totalmente fuera del contenido del curso por lo que nos centraremos en ver algunas de las válvulas incluidas con Tomcat.

Vamos a empezar volviendo a la que está habilitada en server.xml

```
<Valve className="org.apache.catalina.valves.AccessLogValve" directory="logs"
    prefix="localhost_access_log." suffix=".txt"
    pattern="%h %l %u %t &quot;%r&quot; %s %b" />
```

Podemos ver que usa *AccessLogValve* que como su nombre indica registra el acceso. Como está incluida en el elemento <Host> registra todos los accesos a localhost.



Consulta la documentación para hasta entender que hace esta configuración de la válvula.

Estudia el resto de válvulas (las que no acaban en Filter) solo para poder tener una idea de qué hacen. No es necesario que consultes los atributos ni la configuración.

Tras ver las válvulas incluidas, la utilidad de ellas es evidente. Una de ellas es especialmente curiosa y útil en determinados entornos. Si por ejemplo estamos en una intranet de una empresa donde hay varias aplicaciones web disponibles para los empleados, [SingleSignOn](#) nos permite que solo tengan que autenticarse en la primera que se conecten y se mantendrán esas credenciales a través de todas las aplicaciones.

Prueba la válvula SingleSignOn en tu sitio.

Crea un usuario con un rol que tenga acceso a unas aplicaciones pero no a otras. Conéctate con dicho usuario a una aplicación para la que sí tenga acceso. Luego ve a una en la que no tenga acceso. ¿Qué sucede?

## Filtros

Los filtros son una [interfaz de los Servlets](#). Por lo tanto no son exclusivos de Tomcat. Tienen un comportamiento muy similar a las válvulas pero se configuran en el descriptor de despliegue de cada aplicación. Aunque podemos implementar nuestros propios filtros que implementen la interfaz, Tomcat incluye algunos que pueden consultarse en la [página de la documentación](#) correspondiente.

Remote Address Filter, Remote Host Filter y Remote IP Filter/Valve pueden usarse tanto como válvulas como filtros. Comprueba que la documentación en ambos apartados es prácticamente igual. Echa un vistazo para entender qué hace cada uno.

Si miras (por encima) la documentación de los demás filtros verás que tienen usos mucho más concretos que las válvulas, muchos de ellos relacionados con ataques o situaciones muy específicas.

## Configurar el servidor de aplicaciones para cooperar con servidores web

Aunque Tomcat puede gestionar peticiones HTTP y servir archivos HTML esto lo hace mejor Apache (u otro servidor web). Además Apache incluye muchas más opciones de configuración y dispone de muchísimos módulos para realizar diferentes tareas. Otros motivos pueden ser el rendimiento, la distribución en varias máquinas, etc.

En este punto vamos a aprender a configurar un servidor web que reciba las peticiones y se las reenvíe a Tomcat si es necesario. El contenido estático (o derivado de otros lenguajes como PHP si tenemos Apache configurado para ello) lo servirá Apache y las peticiones relacionadas con JSPs o Servlets se las redirigirá a Tomcat.

Como ya hemos comentado la conexión con el exterior se realiza [a través de conectores](#). Los dos principales son el HTTP y el AJP; ambos soportan SSL.

El conector HTTP viene activado por defecto y es el que permite a Tomcat procesar peticiones HTTP y funcionar como un servidor web independiente. Si consultamos el archivo `server.xml` veremos.

```
<Connector port="8080" protocol="HTTP/1.1"
           connectionTimeout="20000"
           URIEncoding="UTF-8"
           redirectPort="8443" />
```

Podemos ver que las opciones de configuración son muchas más en la [ayuda de configuración](#) de Tomcat. No vamos a entrar en detalles pero es importante ver que hay un apartado donde indica cómo usarlo con SSL. De hecho, aunque no esté activo porque aparece comentado, se puede observar que viene preparado para usarlo.

```
<Connector port="8443" protocol="HTTP/1.1" SSLEnabled="true"
```

```
maxThreads="150" scheme="https" secure="true"  
  
clientAuth="false" sslProtocol="TLS" />
```

Sin embargo ahora queremos configurar Tomcat para funcionar como apoyo a un servidor web (en nuestro caso Apache). Para ello necesitamos el [conector Apache JServ Protocol \(AJP\)](#). Esto es desde el lado de Tomcat, pero ¿qué sucede con Apache? En su lado podemos usar dos módulos para realizar esta conexión: `mod_jk` o `mod_proxy`. Incluso de este último existen dos versiones: `mod_proxy_ajp` y `mod_proxy_http`. Aunque no es fácil y ni siquiera hay unanimidad sobre estas cuestiones, vamos a intentar aclarar un poco cual usar:

- **mod\_jk:** Es el más maduro y probado de todos. Realmente es el más potente y se desarrolla por la comunidad Tomcat y no como módulo de Apache. Por ello no viene incluido en la distribución y Apache y hay que añadirlo a mano. En general si no se necesita alguna de sus funcionalidades específicas no parece recomendable afrontar el resto de inconvenientes.
- **mod\_proxy\_ajp:** se distribuye con Apache desde las versiones 2.2 y usa el protocolo AJP que tiene mejor rendimiento que HTTP comunicando servidores.
- **mod\_proxy\_http:** Es similar pero usando HTTP en lugar de AJP. Es una manera fácil y rápida de configurarlo y ponerlo en funcionamiento pero que en general tiene varios inconvenientes que desaconsejan su uso en entornos de producción si no es necesario.

De cualquier forma el primer paso que debemos dar en [instalar Apache](#) y Tomcat. Nosotros ya tenemos el segundo por lo que añadiremos Apache. Aunque nosotros lo vamos a configurar en la misma máquina, puede estar en dos diferentes y de hecho esto mejorará el rendimiento reduciendo la carga. Ten en cuenta que vamos a realizar dos configuraciones diferentes por lo que es buena idea clonar la máquina virtual en este punto.

## mod\_jk

Este módulo no viene con la distribución estándar de apache, por lo que habrá que descargarlo aunque deberíamos comprobar que no esté ya descargado en `/etc/apache2/mods-available`

```
sudo apt-get update  
  
sudo apt-get install libapache2-mod-jk
```

Lo que produce una salida que es interesante leer porque nos indica que ya se activa el módulo al instalarlo (sin necesidad que activarlos manualmente) y de que tenemos que reiniciar apache para que los cambios tengan efecto.

```
Leyendo lista de paquetes... Hecho

Creando árbol de dependencias

Leyendo la información de estado... Hecho

Paquetes sugeridos:

    tomcat6 libapache-mod-jk-doc

Se instalarán los siguientes paquetes NUEVOS:

    libapache2-mod-jk

0 actualizados, 1 se instalarán, 0 para eliminar y 282 no actualizados.

Necesito descargar 144 kB de archivos.

Se utilizarán 532 kB de espacio de disco adicional después de esta operación.

Des:1 http://es.archive.ubuntu.com/ubuntu/ precise/universe libapache2-mod-jk i386 1:1.2.32-1 [144
kB]

Descargados 144 kB en 0seg. (179 kB/s)

Seleccionando paquete libapache2-mod-jk previamente no seleccionado
```

```
(Leyendo la base de datos ... 174672 ficheros o directorios instalados actualmente.)  
  
Desempaquetando libapache2-mod-jk (de .../libapache2-mod-jk_1%3a1.2.32-1_i386.deb) ...  
  
Configurando libapache2-mod-jk (1:1.2.32-1) ...  
  
Enabling module jk.  
  
To activate the new configuration, you need to run:  
  
    service apache2 restart
```

Podemos reiniciar Apache de cualquier forma.

```
/etc/init.d/apache2 restart
```

Deberíamos comprobar que esté activado consultando `mods_enabled` o

```
apachectl -M
```

Si miramos la configuración del módulo

```
gedit /etc/apache2/mods-available/jk.conf
```

Podemos observar un montón de parámetros de configuración, pero nos basta por ahora con fijarnos en la configuración de trabajadores.

```
# We need a workers file exactly once  
  
# and in the global server
```

```
JkWorkersFile /etc/libapache2-mod-jk/workers.properties
```

Un trabajador (*worker*) en Tomcat es una instancia que procesa las peticiones. En nuestra configuración nos bastará con uno solo pero múltiples trabajadores pueden configurarse si es necesario repartir la carga.

La directiva que hemos visto antes indica en qué archivo se configurarán los *workers*. Podemos editar el archivo para configurar estos trabajadores:

```
sudo gedit /etc/libapache2-mod-jk/workers.properties
```

Vemos que viene configurado para Tomcat 6 pero cambiarlo es sencillo

```
#  
  
# workers.tomcat_home should point to the location where you  
  
# installed tomcat. This is where you have your conf, webapps and lib  
  
# directories.  
  
#  
  
workers.tomcat_home=/usr/share/tomcat7
```

Lo siguiente a configurar es la lista de trabajadores que estarán disponibles. Solo vamos a tener uno pero se pueden añadir más separando los nombres con comas (,)

```
#  
  
#----- worker list -----
```

```
#-----  
  
#  
  
#  
  
# The workers that your plugins should create and work with  
  
#  
  
worker.list=ajp13_worker
```

Por ejemplo si tuviéramos dos

```
worker.list = workerprueba1, workerprueba2
```

Por último hay que configurar cada worker

```
#  
  
#----- ajp13_worker WORKER DEFINITION -----  
  
#-----  
  
#  
  
  
#
```



```
# Defining a worker named ajp13_worker and of type ajp13

# Note that the name and the type do not have to match.

#

worker.ajp13_worker.port=8009

worker.ajp13_worker.host=localhost

worker.ajp13_worker.type=ajp13
```

Esto se podría repetir para diferentes workers. Mucho más sobre estas configuraciones se puede [consultar en la ayuda](#).

Como hemos cambiado la configuración del módulo hay que reiniciar Apache.

```
/etc/init.d/apache2 restart
```

Cuando hemos visto el archivo `server.xml` de Tomcat observamos que el conector AJP viene configurado pero comentado. Si lo descomentamos está preparado para funcionar.

```
<!-- Define an AJP 1.3 Connector on port 8009 -->

<Connector port="8009" protocol="AJP/1.3" redirectPort="8443" />
```

Reinicia Tomcat para que los cambios surtan efecto

```
/etc/init.d/tomcat7 restart
```

Todavía nos falta indicar a Apache qué peticiones debe redirigir a Tomcat. Esto se puede hacer en varios puntos dependiendo de cómo tengamos configurado el servidor web, pero en nuestro caso lo vamos a realizar en el sitio web por defecto.

```
gedit /etc/apache2/sites-available/default
```

Donde montamos los directorios e indicamos el trabajador que atenderá las peticiones

```
<VirtualHost *:80>

    ServerAdmin webmaster@localhost


    DocumentRoot /var/www


    JkMount /0330_form/ ajp13_worker

    JkMount /0330_form/* ajp13_worker
```

Tras reiniciar Apache

```
/etc/init.d/apache2 restart
```

podemos acceder a la aplicación a través de Apache, observa que no indicamos el puerto: [http://localhost/0330\\_form/](http://localhost/0330_form/)

Configura Apache y Tomcat para que colaboren y que Apache redirija las peticiones de los ejemplos de Tomcat.

## mod\_proxy

**No se debe usar este módulo con el anterior**, por lo que empezaré con otra máquina virtual. Este módulo es la solución para usar AJP que viene integrada con Apache desde las versiones 2.2. Como se puede observar en la [introducción de la documentación](#) del módulo existen varios derivados del módulo que permiten añadir funcionalidad. Estos derivados requieren que el módulo principal esté presente para poder funcionar.

Lo primero que hay que hacer es activar el módulo de proxy y el que habilita el protocolo HTTP.

```
sudo a2enmod proxy  
  
sudo a2enmod proxy_http
```

Editamos la configuración del módulo

```
gedit /etc/apache2/mods-available/proxy.conf
```

Ten en cuenta que para nuestro objetivo no es necesario descomentar la primera línea como explican en los comentarios que están justo encima. La directiva *ProxyPreserveHost* indica que se le debe pasar el nombre del host que viene en la petición al servidor de aplicaciones en lugar de otro que indiquemos nosotros. Esto es importante para el virtual hosting en el servidor de aplicaciones.

```
# If you only want to use apache2 as a reverse proxy/gateway in  
  
# front of some web application server, you DON'T need  
  
# 'ProxyRequests On'.  
  
ProxyRequests Off
```

```
ProxyPreserveHost On

<Proxy *>

    AddDefaultCharset off

    Order deny,allow

    Allow from all

</Proxy>
```

Posteriormente editamos el archivo del host virtual que estemos configurando

```
gedit /etc/apache2/sites-available/default
```

Y configuramos las redirecciones. Ten en cuenta que el archivo está cortado y que las tres primeras líneas ya estaban así.

```
<VirtualHost *:80>

    ServerAdmin webmaster@localhost


    DocumentRoot /var/www


    ProxyPass /0330_form http://localhost:8080/0330_form
```

```
ProxyPassReverse /0330_form http://localhost:8080/0330_form
```

```
<Location /0330_form >
```

```
    Order allow,deny
```

```
    Allow from all
```

```
</Location>
```

y reiniciamos Apache antes de ponernos a configurar Tomcat.

```
/etc/init.d/apache2 restart
```

Solo tenemos que realizar un pequeño ajuste en el conector de server.xml

```
gedit /etc/tomcat7/server.xml
```

añadimos una línea con el puerto del que recibiremos las redirecciones.

```
<Connector port="8080" protocol="HTTP/1.1"
```

```
    connectionTimeout="20000"
```

```
    URIEncoding="UTF-8"
```

```
    redirectPort="8443"
```

```
proxyPort ="80" />
```

y reiniciamos Tomcat

```
/etc/init.d/tomcat7 restart
```

Ahora, al igual que en el caso anterior podemos acceder a la aplicación a través del servidor web [http://localhost/0330\\_form](http://localhost/0330_form)

### mod\_proxy\_ajp

Pero, ¿y el protocolo AJP?

Se configura prácticamente igual. Además pueden coexistir HTTP (y HTTPS) y AJP. En este apartado solo voy a destacar los puntos donde la configuración es diferente por lo que si quieres configurar solo AJP deberías leer antes el punto anterior.

Lo primero sería activar el módulo

```
sudo a2enmod proxy_ajp
```

Posteriormente editamos el archivo del host virtual que estemos configurando

```
gedit /etc/apache2/sites-available/default
```

Y configuramos las redirecciones. Mantengo la configuración del caso anterior para que se vea que son compatibles.

```
<VirtualHost *:80>

    ServerAdmin webmaster@localhost
```

```
DocumentRoot /var/www
```

```
ProxyPass /0330_form http://localhost:8080/0330_form
```

```
ProxyPassReverse /0330_form http://localhost:8080/0330_form
```

```
<Location /0330_form >
```

```
    Order allow,deny
```

```
    Allow from all
```

```
</Location>
```

```
ProxyPass /examples/jsp ajp://localhost:8009/examples/jsp
```

```
ProxyPassReverse /examples/jsp ajp://localhost:8009/examples/jsp
```

```
<Location /examples/jsp >
```

```
    Order allow,deny
```

```
    Allow from all

</Location>
```

y reiniciamos Apache antes de ponernos a configurar Tomcat.

```
/etc/init.d/apache2 restart
```

Solo tenemos que realizar un pequeño ajuste en el conector de server.xml

```
gedit /etc/tomcat7/server.xml
```

descomentamos el conector AJP

```
<!-- Define an AJP 1.3 Connector on port 8009 -->

<Connector port="8009" protocol="AJP/1.3" redirectPort="8443" />
```

y reiniciamos Tomcat

```
/etc/init.d/tomcat7 restart
```

Ahora, al igual que en el caso anterior podemos acceder a la aplicación a través del servidor web [http://localhost/0330\\_form](http://localhost/0330_form) (con HTTP) y a <http://localhost/examples/jsp/> (con AJP)

Configura Apache y Tomcat para que todo que colaboren mostrando la documentación de Tomcat en <http://localhost/tom-doc/>

Pruébalo con HTTP y luego con AJP.



## Seguridad en el servidor de aplicaciones. Configurar el servidor de aplicaciones con soporte SSL/TSL.

Ya [vimos que HTTPS](#) no era más que HTTP con una capa intermedia. Los mismos principios se aplican en Tomcat. En este caso también usaremos certificados digitales.

Es importante destacar que aunque en Tomcat se puede usar un conector que usa OpenSSL nosotros vamos a ver los que se basan en Java SSL. Desde JDK 1.4 se incluye JSSE en las distribuciones, por lo que tenemos a nuestra disposición todo lo necesario. Nosotros solo vamos a generar un certificado autofirmado para Tomcat, pero [en esta página](#) se pueden ver los pasos para crear uno y pedir que nos firme el certificado una CA.

Vamos a crear un almacén de certificados. La opción `-genkey` especifica que se cree un par clave pública/privada que se almacenan en un certificado autofirmado. “almacen” es el nombre del archivo que se va a crear.

```
sudo keytool -genkey -alias tomcat -keyalg RSA -keystore /var/lib/tomcat7/almacen
```

Lo que nos va pidiendo los datos necesarios.

```
Introduzca la contraseña del almacén de claves:
```

```
Volver a escribir la contraseña nueva:
```

```
¿Cuáles son su nombre y su apellido?
```

```
[Unknown]: Sergio Cuesta
```

```
¿Cuál es el nombre de su unidad de organización?
```

[Unknown]: Consejería de Educación

¿Cuál es el nombre de su organización?

[Unknown]: Comunidad de Madrid

¿Cuál es el nombre de su ciudad o localidad?

[Unknown]: Madrid

¿Cuál es el nombre de su estado o provincia?

[Unknown]: Madrid

¿Cuál es el código de país de dos letras de la unidad?

[Unknown]: ES

¿Es correcto CN=Sergio Cuesta, OU=Consejería de Educación, O=Comunidad de Madrid, L=Madrid, ST=Madrid, C=ES?

[no]: si

Introduzca la contraseña de clave para <tomcat>

(INTRO si es la misma contraseña que la del almacén de claves):

Durante esta configuración nos pide que definamos las contraseñas tanto del almacén como del archivo de claves. En mi caso ambas son “sergio”.

Lo siguiente sería habilitar el conector HTTPS en server.xml

```
<!-- Define a SSL HTTP/1.1 Connector on port 8443

      This connector uses the JSSE configuration, when using APR, the
      connector should be using the OpenSSL style configuration
      described in the APR documentation -->

<Connector port="8443" protocol="HTTP/1.1" SSLEnabled="true"
           maxThreads="150" scheme="https" secure="true"
           clientAuth="false" sslProtocol="TLS"

           keystoreFile="/var/lib/tomcat7/almacen" keystorePass="sergio"

           keyAlias="tomcat" keyPass="sergio" />
```

Puedes ver que he añadido las dos últimas líneas para indicar la configuración de seguridad. Tras reiniciar Tomcat podemos conectarnos al servidor de aplicaciones con conexiones seguras <https://localhost:8443/>

El problema es que aunque ahora puedo conectarme a las aplicaciones mediante HTTPS sigo teniendo la opción de hacerlo mediante HTTP [http://localhost:8080/0330\\_form/](http://localhost:8080/0330_form/)

Para obligar a que a una aplicación haya que conectarse mediante HTTPS hay que modificar su descriptor de despliegue. Para hacer la prueba modifico el descriptor de despliegue de 0320\_basic

```
gedit /var/lib/tomcat7/webapps/0320_basic/WEB-INF/web.xml
```

Hay que cambiar el tipo de transporte que se obliga de NONE a CONFIDENTIAL

```
<user-data-constraint>

    <transport-guarantee>CONFIDENTIAL</transport-guarantee>

</user-data-constraint>
```

Podemos ver que ahora si probamos a conectarnos a [http://localhost:8080/0320\\_basic/](http://localhost:8080/0320_basic/) nos redirige automáticamente a [https://localhost:8443/0320\\_basic/](https://localhost:8443/0320_basic/)

Ya sabemos configurar Apache y Tomcat con SSL pero **¿qué sucede con la conexión entre ellos?**

Es muy raro que esta conexión se realice a través de una red en la que no confiemos. Incluso en el caso de que se encuentren en máquinas diferentes suele usarse algún tipo de conexión privada entre ellos para que los mensajes que se envían no se distribuyan por redes más o menos públicas. Si necesitáramos configurarlo de todas formas deberíamos prestar atención a las directivas que comienzan por SSLProxy del [módulo mod\\_ssl](#) de Apache.

De todas formas para asegurar la conexión Apache – Tomcat (con HTTPS) tendríamos que usar un conector HTTPS, lo que hago un poco más arriba y cambiar las [cadenas de redirección](#) en la configuración del host virtual para que utilicen HTTPS y el puerto 8443.

## Prácticas finales

En este punto y con los conocimientos de hosts virtuales que aprendimos en Apache, estamos preparados para aprender a crear hosts virtuales en Tomcat por nosotros mismos. Por ello debes aprender a configurar un host virtual.

Te ayudarán mucho, la aplicación de host manager que instalamos con los paquetes adicionales y repasar todo este tema para ver dónde se configuraba un elemento <Host> y qué implicaciones tenía. Ten en cuenta que al intentarte conectar al host manager, si no puedes te dice el rol que tienes que tener para poder conectarte. También hay un [pequeño tutorial](#) en la documentación de Tomcat. También es importante recordar que el host manager no configura el elemento <Host> de server.xml, pero internet está lleno de tutoriales para realizar esto.

Tu objetivo final debe ser poder configurar un nuevo host virtual en Tomcat a mano y ser capaz de desplegar una de las aplicaciones de ejemplo (mejor con autenticación) sobre él. Las siguientes líneas te pueden dar una idea de cómo hacerlo.

```
<Host name="www.sergio.es" appBase="webapps_sergio.es"
  unpackWARs="true" autoDeploy="true" >
  <Alias>sergio.es</Alias>
  <Valve className="org.apache.catalina.valves.AccessLogValve"
    directory="logs"
    prefix="sergio.es_access_log." suffix=".log"
    pattern="%h %l %u %t &quot;%r&quot; %s %b"
    resolveHosts="false" />
</Host>
```

---

## Práctica a entregar

---

El último ejercicio que vamos a realizar implica usar casi todo lo que hemos visto en Apache y Tomcat. Debes configurar un sistema que gestiona peticiones HTTPS exclusivamente (no HTTP) por lo que tanto Tomcat como Apache deben usar SSL. El sitio debe tener tu nombre como nombre de dominio y tendrá archivos HTML que servirá Apache y parte con Servlets y JSPs que servirá Tomcat a través de Apache. El control de acceso se realizará con un JDBCRealm. Puedes usar los ejemplos del site para la parte de aplicaciones en el servidor. Si decides desarrollar las tuyas propias se tendrá en cuenta para la nota, pero no es necesario para que la práctica esté bien. Habrá enlaces en la parte HTML a los Servlets/JSPs. Un usuario introducirá únicamente sus credenciales una vez para todas las aplicaciones.

Pon tu nombre en comentarios tanto al principio como al final de cada archivo de configuración o de contenido que edites.

Evita que se pueda acceder a las aplicaciones de Tomcat sin pasar por Apache.

## Tema 4: Instalación y administración de servidores de transferencia de archivos

## Configuración del servicio de transferencia de archivos

El protocolo clásico para la transferencia de archivos en Internet se denomina FTP (*File Transfer Protocol*). Con el estado actual de Internet y las múltiples opciones de transferencia de archivos en la web puede parecer algo innecesario pero sigue siendo una opción sencilla y específica por lo que en ámbitos profesionales continúa gozando de buenísima salud. Por ejemplo sigue siendo el método más habitual para subir archivos, actualizaciones o modificaciones de contenido a un servidor web, especialmente en el modo de hosting.

FTP se ajusta a una arquitectura cliente/servidor como todo lo que hemos visto hasta ahora. Primero instalaremos el servidor, para luego ver diferentes clientes; en este último apartado existen alternativas muy cercanas a nosotros, pero a diferencia de los servidores web o de aplicaciones en los que el cliente lo usamos a diario, en este caso [lo mejor será algo más específico](#).

En Linux existen muchos servidores FTP diferentes, no hay más que echar un vistazo al [paquete virtual](#) de Ubuntu. Los dos más populares actualmente son **ProFTPD** y **vsFTPD**. El primero es más sencillo de utilizar y sus archivos de configuración y estructura similar hacen que se parezca mucho a Apache. Sin embargo el segundo es el servidor FTP por defecto en las principales distribuciones de Linux lo que hace que sea más sencillo de instalar y además se considera más seguro.

Al final es cuestión de preferencias personales pero nosotros trabajaremos con vsFTPD (Very Secure FTP Daemon).

Si necesitas opciones para otros sistemas puedes consultar algunas alternativas en la [comparativa de la Wikipedia](#).

En Linux, un usuario FTP tiene su propia carpeta de usuario asociada (en /home/nombre-usuario-ftp/) por lo que la integración es muy alta y FTP se beneficia de la completa y potente gestión de usuarios de Linux.

### Instalando el servidor

Desde Linux lo primero es actualizar los repositorios para luego instalar el servidor.

```
sudo apt-get update
```



```
sudo apt-get install vsftpd
```

Podemos ver que se ha añadido un script de autoarranque

```
sudo gedit /etc/init.d/vsftpd
```

por lo que los métodos habituales de gestión, mediante el script o como servicio están disponibles

```
sudo /etc/init.d/vsftpd start
```

```
sudo service vsftpd start
```

con los comandos start, stop, restart y reload.

Podemos probar que todo haya ido bien mediante el cliente en modo texto

```
ftp localhost
```

lo que nos pedirá unas credenciales para acceder. Por defecto nos conectamos como el usuario *anonymous* y sin contraseña.

```
Connected to localhost.
```

```
220 (vsFTPd 2.3.5)
```

```
Name (localhost:root): anonymous
```

```
331 Please specify the password.
```

```
Password:
```

```
230 Login successful.  
  
Remote system type is UNIX.  
  
Using binary mode to transfer files.  
  
ftp> quit  
  
221 Goodbye.
```

Observa que usamos *quit* para desconectarnos.

---

## Configurando el servidor

El archivo de configuración de vsFTPD se puede consultar como viene siendo habitual.

```
sudo gedit /etc/vsftpd.conf
```

Instala el servidor vsFTPd

Como el archivo está muy comentado y ya conocemos la mayoría de los conceptos implicados podemos discutir en clase las directivas que aparecen. En caso de no conocer algo utiliza internet.

Debemos darnos cuenta de que la combinación del uso de los directorios de usuarios locales (en el servidor Linux) con su gestión en Apache para que su directorio personal incluya una carpeta para su sitio web, las posibilidades de DNS y las opciones que acabamos de comentar en FTP permiten configurarse un servicio de *hosting* propio.

Tras cada cambio en la configuración es necesario reiniciar el servidor. Para la mayoría de los casos es suficiente con recargar la configuración (reload).

```
service vsftpd reload
```

Todas las opciones de configuración se pueden consultar [en esta página](#) de Ubuntu. También está bien [explicadas aquí](#).

## Tipos de usuarios y accesos al servicio

En FTP existen dos tipos básicos de usuario, los corrientes y los anónimos. Realmente definen el tipo de acceso porque indican si te estás autenticando con un usuario concreto o estás utilizando una cuenta anónima que generalmente no requiere autenticación. Un servidor FTP puede servir ambos tipos simultáneamente.

- **FTP anónimo:** Este modo se utiliza generalmente cuando el servidor FTP se usa para distribuir cualquier tipo de archivo o archivos a un número muy elevado de usuarios en una situación en la que la identificación no es muy importante. Si por ejemplo hemos realizado una aplicación de software libre y queremos distribuirla es una buena opción. En este tipo de conexión solo se le pide al cliente un nombre de usuario anónimo (generalmente y por defecto es *anonymous*) y si acaso (no siempre) una contraseña que se refiere a cualquier dirección de correo electrónico válida. Una vez nos hemos conectado al servidor tendremos acceso al directorio anónimo y sus subdirectorios.
- **FTP corriente:** En este caso los usuarios de FTP son los que existen en la máquina en la que instalamos el servidor. Estos usuarios podrán leer de y copiar a su directorio personal archivos remotamente. Las mismas credenciales que tienen en la máquina serán las que necesiten para conectarse a mediante FTP. vsFTPD permite que este tipo de conexión se restrinja a los usuarios de un grupo determinado. El uso de este tipo de conexión es muy habitual en los hostings web, aunque no es práctico cuando tenemos muchísimos usuarios como puede ser el caso. Para ello se utilizan los denominados **usuarios virtuales** que tendrán credenciales FTP pero no cuenta en el servidor Linux.

Se pueden configurar estas opciones en el archivo

```
sudo gedit /etc/vsftpd.conf
```

Por defecto nuestra instalación solo permite conexión anónima.

```
# Allow anonymous FTP? (Beware - allowed by default if you comment this out).  
  
anonymous_enable=YES  
  
#
```

```
# Uncomment this to allow local users to log in.  
  
#local_enable=YES
```

Pero vemos que activar o desactivar otros tipos de usuarios es cuestión de comentar o descomentar una línea.

Prueba a permitir que los usuarios con cuenta en el servidor se conecten con sus propias credenciales. Debes tener en cuenta que esto no activa la conexión de root por lo que será necesario que utilices otro usuario para conectarte.

Deshabilita la conexión anónima y prueba ambos casos. Vuelve a habilitar la conexión anónima.

El directorio por defecto para el usuario anónimo es `/srv/ftp/` y para los usuarios regulares su carpeta personal.

```
#ftp localhost  
  
Connected to localhost.  
  
220 (vsFTPd 2.3.5)  
  
Name (localhost:root): sergio  
  
331 Please specify the password.  
  
Password:  
  
230 Login successful.  
  
Remote system type is UNIX.
```

```
Using binary mode to transfer files.
```

```
ftp> ls
```

```
200 PORT command successful. Consider using PASV.
```

```
150 Here comes the directory listing.
```

```
drwxr-xr-x    2 1000    1000          4096 Sep 26 10:42 Descargas
drwxr-xr-x    2 1000    1000          4096 Sep 26 10:42 Documentos
drwxr-xr-x    2 1000    1000          4096 Sep 26 10:42 Escritorio
drwxr-xr-x    2 1000    1000          4096 Sep 26 10:42 Im??genes
drwxr-xr-x    2 1000    1000          4096 Sep 26 10:42 M??sica
drwxr-xr-x    2 1000    1000          4096 Sep 26 10:42 Plantillas
drwxr-xr-x    2 1000    1000          4096 Sep 26 10:42 P??blico
drwxr-xr-x    2 1000    1000          4096 Sep 26 10:42 V??deos
-rw-r--r--    1 1000    1000          8445 Sep 26 10:15 examples.desktop
```

```
226 Directory send OK.
```

```
ftp> quit
```

221 Goodbye.

Prueba a situar algún archivo en la carpeta del usuario anónimo. Conéctate mediante ftp para comprobar que existe el archivo.

Podemos modificar este directorio

```
mkdir /var/ftpcomp/pub  
  
sudo gedit /etc/vsftpd.conf
```

y escribimos la directiva correcta

```
anon_root=/var/ftpcomp/pub
```

tras reiniciar el servidor podemos ver que la carpeta raíz de los usuarios anónimos ha cambiado. Puedes crear un archivo dentro para comprobarlo.

Aunque generalmente no es nada recomendable (puede suponer un gran agujero de seguridad por el que se cuecen muchísimos ataques a nuestro servidor), a veces es necesario que los usuarios anónimos puedan subir archivos al servidor. Los usuarios ftp no tienen permiso para escribir en su directorio raíz por lo que debemos crear otro directorio dentro en el que permitamos subir archivos.

Creemos la carpeta y nos aseguramos de que tenga el propietario y los permisos adecuados.

```
mkdir /var/ftpcomp/pub/upload  
  
chown root.root /var/ftpcomp/pub/upload/  
  
chmod 757 /var/ftpcomp/pub/upload/
```

Puedes consultar la lista de comandos de FTP desde consola en muchas páginas de Internet, [por ejemplo aquí](#).

Cuando intentamos conectarnos puede darnos un error 500

```
# ftp localhost

Connected to localhost.

220 (vsFTPd 2.3.5)

Name (localhost:root): anonymous

331 Please specify the password.

Password:

500 OOPS: vsftpd: refusing to run with writable root inside chroot()

Login failed.

ftp> quit

421 Service not available, remote server has closed connection
```

Esto se debe a que el usuario no debe tener permisos de escritura en su carpeta raíz así que podemos quitárselos y ya está. Por ello ahora tendrá que escribir dentro de upload.

```
chmod 555 /var/ftpcomp/pub
```

También hay que modificar una directiva para que se permitan subir archivos al servidor y recargar la configuración

```
# Uncomment this to enable any form of FTP write command.
```



```
write_enable=YES
```

y un poco más abajo configuramos el que el usuario anónimos pueda subir archivos y el que pueda crear directorios.

```
# Uncomment this to allow the anonymous FTP user to upload files. This only
# has an effect if the above global write enable is activated. Also, you will
# obviously need to create a directory writable by the FTP user.

anon_upload_enable=YES

#

# Uncomment this if you want the anonymous FTP user to be able to create
# new directories.

anon_mkdir_write_enable=YES
```

recargamos la configuración

```
service vsftpd reload
```

y probamos. Observa que en el ejemplo se intenta en el directorio que hemos configurado para ello y en el que no. Ten en cuenta que el archivo a subir está en el directorio desde el que he ejecutado el cliente FTP.

```
# ftp localhost

Connected to localhost.
```

```
220 (vsFTPd 2.3.5)

Name (localhost:root): anonymous

331 Please specify the password.

Password:

230 Login successful.

Remote system type is UNIX.

Using binary mode to transfer files.

ftp> cd upload

250 Directory successfully changed.

ftp> put aSubir.txt

local: aSubir.txt remote: aSubir.txt

200 PORT command successful. Consider using PASV.

150 Ok to send data.

226 Transfer complete.

ftp> cd ..
```

```
250 Directory successfully changed.  
  
ftp> put aSubir.txt  
  
local: aSubir.txt remote: aSubir.txt  
  
200 PORT command successful. Consider using PASV.  
  
553 Could not create file.  
  
ftp> quit  
  
221 Goodbye.
```

Volvemos a desactivar la opción de que los usuarios anónimos puedan usar subir archivos simplemente volviendo a comentar las dos últimas directivas pero mantenemos la opción de que los usuarios corrientes puedan subir archivos.

```
useradd -m prueba  
  
passwd prueba
```

Un grave problema de seguridad en FTP y Linux (no solo con vsFTPD) es que un usuario tiene acceso a toda la estructura de archivos del servidor. Aunque no tenga permisos para escribir puede leer y listar...

```
ftp localhost  
  
Connected to localhost.  
  
220 (vsFTPD 2.3.5)
```

```
Name (localhost:root): prueba

331 Please specify the password.

Password:

230 Login successful.

Remote system type is UNIX.

Using binary mode to transfer files.

ftp> ls

200 PORT command successful. Consider using PASV.

150 Here comes the directory listing.

-rw-r--r--    1 1002    1003          8445 Apr 16  2012 examples.desktop

226 Directory send OK.

ftp> cd /

250 Directory successfully changed.

ftp> ls

200 PORT command successful. Consider using PASV.
```

150 Here comes the directory listing.

```

drwxr-xr-x    2 0      0          4096 Sep 27 07:58 bin
drwxr-xr-x    3 0      0          4096 Sep 27 08:02 boot
drwxr-xr-x    2 0      0          4096 Sep 26 08:28 cdrom
drwxr-xr-x   15 0      0          4080 Feb 09 06:28 dev
drwxr-xr-x  128 0      0        12288 Feb 09 10:42 etc
drwxr-xr-x    4 0      0          4096 Feb 09 10:41 home
lrwxrwxrwx    1 0      0              37 Sep 27 08:01 initrd.img -> /boot/initrd.img-3.2.0-31-
generic-pae
lrwxrwxrwx    1 0      0             36 Sep 26 10:18 initrd.img.old -> boot/initrd.img-3.2.0-29-
generic-pae
drwxr-xr-x   21 0      0          4096 Sep 26 10:19 lib
drwx-----    2 0      0        16384 Sep 26 08:24 lost+found
drwxr-xr-x    2 0      0          4096 Sep 27 08:27 media
drwxr-xr-x    2 0      0          4096 Apr 19  2012 mnt
drwxr-xr-x    3 0      0          4096 Sep 27 07:14 opt

```

```
dr-xr-xr-x 153 0      0      0 Feb 09 06:27 proc
drwx----- 20 0      0     4096 Feb 09 10:07 root
drwxr-xr-x 22 0      0      780 Feb 09 09:29 run
drwxr-xr-x  2 0      0     4096 Sep 27 08:20 sbin
drwxr-xr-x  2 0      0     4096 Mar 05 2012 selinux
drwxr-xr-x  3 0      0     4096 Feb 09 07:04 srv
drwxr-xr-x 13 0      0      0 Feb 09 06:27 sys
drwxrwxrwt  9 0      0     4096 Feb 09 10:17 tmp
drwxr-xr-x 10 0      0     4096 Feb 09 10:07 usr
drwxr-xr-x 14 0      0     4096 Feb 09 09:33 var
lrwxrwxrwx  1 0      0      33 Sep 27 08:01 vmlinuz -> boot/vmlinuz-3.2.0-31-generic-pae
lrwxrwxrwx  1 0      0      33 Sep 26 10:18 vmlinuz.old -> boot/vmlinuz-3.2.0-29-generic-
pae
226 Directory send OK.
ftp> quit
221 Goodbye.
```

La forma fácil de solucionarlo es

```
chroot_local_user=YES
```

Para la gran mayoría de los casos, a no ser que necesitemos un control excesivo sobre lo que hacen los usuarios esta opción sería más que suficiente.

### Configuración avanzada del chroot

Realmente la opción anterior tiene un pequeño agujero de seguridad como se indica en las FAQ de vsftpd

Q) Help! What are the security implications referred to in the "chroot\_local\_user" option?

A) Firstly note that other ftp daemons have the same implications. It is a generic problem. The problem isn't too severe, but it is this: Some people have FTP user accounts which are not trusted to have full shell access. If these accounts can also upload files, there is a small risk. A bad user now has control of the filesystem root, which is their home directory. The ftp daemon might cause some config file to be read - e.g. /etc/some\_file. With chroot(), this file is now under the control of the user. vsftpd is careful in this area. But, the system's libc might want to open locale config files or other settings...

Para evitar esto hay que cambiar las opciones de configuración.

```
# You may specify an explicit list of local users to chroot() to their home
# directory. If chroot_local_user is YES, then this list becomes a list of
# users to NOT chroot().

# (Warning! chroot'ing can be very dangerous. If using chroot, make sure that
# the user does not have write access to the top level directory within the
```

```
# chroot)

chroot_local_user=NO

chroot_list_enable=YES

# (default follows)

chroot_list_file=/etc/vsftpd.chroot_list
```

En el que he dejado el archivo de usuarios por defecto. Lo siguiente sería cambiar los permisos al directorio del usuario prueba para que no tenga permisos de escritura en su carpeta personal (y así nos deje conectarnos)

```
chmod 555 /home/prueba
```

El problema es que cualquier otro usuario que se conecte todavía puede acceder a todo

```
# useradd -m pru

# passwd pru

# ftp localhost

Connected to localhost.

220 (vsFTPd 2.3.5)

Name (localhost:root): pru

331 Please specify the password.
```



Password:

230 Login successful.

Remote system type is UNIX.

Using binary mode to transfer files.

ftp> ls

200 PORT command successful. Consider using PASV.

150 Here comes the directory listing.

-rw-r--r--	1	1003	1004	8445	Apr 16	2012	examples.desktop
------------	---	------	------	------	--------	------	------------------

226 Directory send OK.

ftp> cd ..

250 Directory successfully changed.

ftp> ls

200 PORT command successful. Consider using PASV.

150 Here comes the directory listing.

drwxr-xr-x	2	1003	1004	4096	Feb 09	11:02	pru
------------	---	------	------	------	--------	-------	-----

```
dr-xr-xr-x    2 1002    1003          4096 Feb 09 10:56 prueba
drwxr-xr-x   21 1000    1000          4096 Feb 09 10:15 sergio
226 Directory send OK.
ftp> cd ..
250 Directory successfully changed.
ftp> ls
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
drwxr-xr-x    2 0        0          4096 Sep 27 07:58 bin
drwxr-xr-x    3 0        0          4096 Sep 27 08:02 boot
drwxr-xr-x    2 0        0          4096 Sep 26 08:28 cdrom
drwxr-xr-x   15 0        0          4080 Feb 09 06:28 dev
drwxr-xr-x  128 0        0         12288 Feb 09 11:02 etc
drwxr-xr-x    5 0        0          4096 Feb 09 11:02 home
lrwxrwxrwx    1 0        0              37 Sep 27 08:01 initrd.img -> /boot/initrd.img-3.2.0-31-
generic-pae
```

```

lrwxrwxrwx    1 0      0          36 Sep 26 10:18 initrd.img.old -> boot/initrd.img-3.2.0-29-
generic-pae

drwxr-xr-x   21 0      0          4096 Sep 26 10:19 lib

drwx-----   2 0      0         16384 Sep 26 08:24 lost+found

drwxr-xr-x    2 0      0          4096 Sep 27 08:27 media

drwxr-xr-x    2 0      0          4096 Apr 19  2012 mnt

drwxr-xr-x    3 0      0          4096 Sep 27 07:14 opt

dr-xr-xr-x  152 0      0           0 Feb 09 06:27 proc

drwx-----  20 0      0          4096 Feb 09 10:07 root

drwxr-xr-x   22 0      0           780 Feb 09 09:29 run

drwxr-xr-x    2 0      0          4096 Sep 27 08:20 sbin

drwxr-xr-x    2 0      0          4096 Mar 05  2012 selinux

drwxr-xr-x    3 0      0          4096 Feb 09 07:04 srv

drwxr-xr-x   13 0      0           0 Feb 09 06:27 sys

drwxrwxrwt    9 0      0          4096 Feb 09 10:17 tmp

drwxr-xr-x   10 0      0          4096 Feb 09 10:07 usr

```

```
drwxr-xr-x  14 0      0      4096 Feb 09 09:33 var
lrwxrwxrwx   1 0      0      33 Sep 27 08:01 vmlinuz -> boot/vmlinuz-3.2.0-31-generic-pae
lrwxrwxrwx   1 0      0      33 Sep 26 10:18 vmlinuz.old -> boot/vmlinuz-3.2.0-29-generic-pae
226 Directory send OK.
ftp> quit
221 Goodbye.
```

Para solucionarlo siempre podemos incluir a los usuarios en el archivo de lista.

```
gedit /etc/vsftpd.chroot_list
```

y escribir el nombre de usuario uno debajo de otro

```
prueba
sergio
pru
```

En mi caso Sergio no puede conectarse al servidor FTP porque sigue teniendo permisos de escritura en su directorio. Combinando estos elementos se puede conceder acceso a los usuarios que queramos.

Para la gran mayoría de los casos, a no ser que necesitemos un control excesivo sobre lo que hacen los usuarios, la opción de

## Usuarios Virtuales

Todo este apartado lo haremos con una copia de la máquina virtual para no influir en el resto del tema.

También se pueden utilizar usuarios virtuales. Para ello hay que estar usando PAM (Pluggable Authentication Modules) de Linux. En el archivo de configuración

```
# This string is the name of the PAM service vsftpd will use.  
  
pam_service_name=vsftpd
```

Vamos a realizar un ejemplo muy útil para alojar sitios de diferentes personas que puedan actualizar sus propios archivos mediante FTP.

Instalamos el módulo

```
sudo apt-get install vsftpd libpam-pwdfile
```

Vamos a crear un nuevo archivo de configuración así que renombramos el original y creamos uno nuevo

```
sudo mv /etc/vsftpd.conf /etc/vsftpd.conf.bak  
  
sudo gedit /etc/vsftpd.conf
```

con las siguientes directivas

```
listen=YES  
  
anonymous_enable=NO  
  
local_enable=YES
```

```
write_enable=YES  
  
local_umask=022  
  
nopriv_user=vsftpd  
  
virtual_use_local_privs=YES  
  
guest_enable=YES  
  
user_sub_token=$USER  
  
local_root=/var/www/$USER  
  
chroot_local_user=YES  
  
hide_ids=YES  
  
guest_username=vsftpd
```

Investiga y comenta en clase qué hacen las directivas que desconoces.

En el siguiente paso hay que usar *htpasswd* por lo que necesitamos tener instalado Apache lo que en el ejemplo que vemos es más que necesario.

Creamos un directorio para colocar los archivos de configuración

```
sudo mkdir /etc/vsftpd
```

y creamos

```
sudo htpasswd -cd /etc/vsftpd/ftpd.passwd sergiol
```

donde -c indica que se cree el archivo y -d que se utilice MD5. Para los siguientes usuarios omitiremos la opción -c

```
sudo -d htpasswd /etc/vsftpd/ftpd.passwd maria
```

Copiamos el archivo de configuración de PAM

```
sudo mv /etc/pam.d/vsftpd /etc/pam.d/vsftpd.bak
```

y creamos otro

```
sudo gedit /etc/pam.d/vsftpd
```

con el contenido

```
auth required pam_pwdfile.so pwdfile /etc/vsftpd/ftpd.passwd  
  
account required pam_permit.so
```

¡Y creamos un usuario local sin acceso en local! Cuando el usuario se conecte a través de un cliente FTP al servidor, este usuario que creamos es el que determinará los permisos y propiedades de archivos. A él se asociarán los usuarios virtuales que se conecten a través de FTP

```
sudo useradd --home /home/vsftpd --gid nogroup -m --shell /bin/false vsftpd
```

Ya solo queda configurar los directorios

```
mkdir /var/www/sergiol
```

```
chmod -w /var/www/sergio1  
  
mkdir /var/www/sergio1/www  
  
chmod -R 755 /var/www/sergio1/www  
  
chown -R vsftpd:nogroup /var/www/sergio1
```

Ahora podemos conectarnos y subir archivos a `/var/www/sergio1/www/` pero no al raíz del usuario.

## Permisos y cuotas

Hemos visto en los puntos anteriores cómo se configuran los **permisos** generales de usuarios anónimos y corrientes. Los permisos específicos en el caso que estamos tratando se concretan con los permisos del usuario o grupo al que pertenece en Linux para las carpetas correspondientes. Es importante recordar que si al ir a conectarnos con un usuario al servidor vsFTPD nos aparece el error 500, es porque no se le pueden dar permisos de escritura al usuario en su carpeta raíz.

Establecer límites a los usuarios es de vital importancia en un sistema FTP para evitar que unos pocos consuman demasiados recursos. Aunque no es parte de este módulo, quiero destacar que es posible limitar el ancho de banda y número de conexiones simultáneas que puede usar el mismo usuario. El tema que aquí vamos a discutir es el más importante porque un solo usuario puede ocupar demasiado espacio de disco y no dejar nada para otros. Para evitar esto se establece una **cuota de espacio en disco**.

vsFTPD permite establecer cuotas para cada usuario mediante un paquete adicional. Este paquete de hecho establece cuotas para los usuarios en Linux, no solo en vsFTPD.

```
sudo apt-get install quota
```

Lo siguiente que hay que hacer es habilitar el uso de cuotas en nuestro sistema de archivos. Para ello se edita un fichero

```
sudo gedit /etc/fstab
```



y localizamos la línea del punto de montaje del sistema de archivos donde estableceremos la cuota

```
# /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>
proc          /proc                proc    nodev,noexec,nosuid 0           0

# / was on /dev/sda1 during installation

UUID=5636501c-a7be-44e1-a7d9-0c78fce32598 /          ext4    errors=remount-ro 0           1

# swap was on /dev/sda5 during installation

UUID=70f12e1a-101a-4ef7-9501-3cebfeef9408 none      swap     sw              0           0
```

En este caso viene con el identificador del disco pero puede aparecer como /dev/hda1/ pero lo que yo he buscado es que el punto de montaje fuera / (la raíz). Detrás del tipo de extensión añadimos **usrquota** y **grpquota** para cuotas de usuarios y de grupos respectivamente. Puede añadirse solo una de las dos si la otra no nos va a hacer falta.

```
# / was on /dev/sda1 during installation

UUID=5636501c-a7be-44e1-a7d9-0c78fce32598 /          ext4      usrquota,grpquota,errors=remount-ro
0          1
```

Para que se carguen los cambios que afectan al sistema de archivos es necesario reiniciar el equipo o ejecutar

```
mount -o remount /
```

Lo siguiente ya sería asignar cuotas de disco a los usuarios. Para ello hay que entender unos parámetros que se establecen para cada cuota:

- **Blocks:** Indica el número de bloques que podrá ocupar el usuario en disco. Se mide en KB
- **Soft:** Es el límite blando. Cuando el usuario lo supera se le avisa, pero dispone de una semana (por defecto, se puede modificar) para ocupar un espacio menor al que indica este parámetro. Si pasada una semana no lo ha liberado, se establecerá como un límite efectivo y no podrá ocupar nada más.
- **Hard:** El límite duro indica la cantidad que no puede superar de ninguna manera.
- **Inodes:** El número de nodos y por lo tanto ficheros que puede tener el usuario.

Primero debemos saber qué sistema de ficheros estamos usando:

```
# df -T
```

S.ficheros	Tipo	1K-bloques	Usado	Disponible	Uso%	Montado en
/dev/sda1	ext4	7223800	3150336	3706516	46%	/
udev	devtmpfs	505356	4	505352	1%	/dev
tmpfs	tmpfs	205056	792	204264	1%	/run
none	tmpfs	5120	0	5120	0%	/run/lock

```
none          tmpfs          512640        8          512632        1% /run/shm
```

Ahora ya podemos establecer una cuota.

```
setquota -u prueba 10240 20480 0 0 /dev/sda1
```

Hemos establecido una cuota blanda de 10240KB (10MB) y una dura de (20MB) y sin límites duros ni blandos de inodos. Si establecemos 0 como valor de alguno de los límites no ese límite no se aplicará.

Discute en clase las implicaciones que puede tener no establecer límites duros o blandos.

La cuota de un grupo se establecería con la opción `-g` en lugar de `-u`. Este límite es para el grupo en general, no para cada usuario del grupo.

También pueden editarse las cuotas:

```
edquota -u NombreUsuario
```

```
edquota -g NombreGrupo
```

Para ver un listado de todos los usuarios y sus cuotas usamos

```
repquota -a
```

y para consultar la de un usuario en particular

```
quota -u prueba
```

Establece una cuota blanda de 1MB para un usuario y una dura de 2MB, prueba a superar la blanda pero quedándote por debajo de la dura y luego prueba a superar la dura. Apunta lo que sucede.

## Modos de conexión del cliente

FTP es un protocolo que funciona siempre sobre TCP ya que no hay opción o versión con UDP. Típicamente FTP utiliza dos puertos, uno de control (para órdenes o comandos, el 21) y un puerto para datos (20). Si te has fijado durante los puntos anteriores, el propio servidor de FTP nos pedía que pasáramos al modo pasivo, pero... ¿qué es? Primero veremos el activo.

### Modo activo

En el modo activo las cosas funcionan como se exponen a continuación:

1. El cliente hace una petición al servidor en el puerto 21 desde lo que se conoce como un puerto alto (>1023 que llamaremos N), con comandos como por ejemplo "ls" o "get".
2. Si la petición requiere la transferencia de datos del servidor al cliente, **el servidor** abre una conexión a otro puerto alto (generalmente N+1)
3. El servidor transfiere los datos.

En los pasos anteriores se han omitido las confirmaciones, etc involucradas.

Este modo sigue la lógica de la mayoría de las conexiones TCP.

El problema con el que se encuentra este modo es que si el cliente se encuentra tras un firewall o cualquier dispositivo que haga NAT lo más probable es que el servidor no pueda conectarse con él haciendo que falle el paso 2.

### Modo pasivo

La solución a esto es el modo pasivo, en el que las cosas suceden de manera un poco diferente:

1. El cliente hace una petición al servidor en el puerto 21 desde lo que se conoce como un puerto alto (>1023 que llamaremos N), con comandos como por ejemplo "ls" o "get". Este paso es idéntico.

2. El cliente ha avisado al servidor de que él se encargará de comenzar la conexión, por lo que el servidor contesta con otro puerto alto al que el cliente debe conectarse.
3. Si la petición requiere la transferencia de datos del servidor al cliente, **el cliente** abre una conexión a ese puerto.
4. El servidor transfiere los datos.

Como puedes observar la principal diferencia es quién abre la conexión y se hace para evitar los problemas comentados en el apartado anterior.

Por defecto el cliente se conecta en modo activo, pero se puede pasar al pasivo mediante el comando adecuado.

```
#ftp localhost

Connected to localhost.

220 (vsFTPd 2.3.5)

Name (localhost:root): sergiol

331 Please specify the password.

Password:

230 Login successful.

Remote system type is UNIX.

Using binary mode to transfer files.

ftp> ls

200 PORT command successful. Consider using PASV.
```

```
150 Here comes the directory listing.

drwxr-xr-x    2 ftp      ftp          4096 Feb 09 11:19 www

226 Directory send OK.

ftp> passive

Passive mode on.

ftp> ls

227 Entering Passive Mode (127,0,0,1,74,3).

150 Here comes the directory listing.

drwxr-xr-x    2 ftp      ftp          4096 Feb 09 11:19 www

226 Directory send OK.

ftp> quit

221 Goodbye.
```

Para activar o desactivar el modo pasivo se utiliza en el archivo de configuración (está activado)

```
pasv_enable=YES
```

o lo mismo con NO

## Protocolo seguro de transferencia de archivos

El uso de SSL/TSL en FTP tiene sentido especialmente a la hora de comunicar las credenciales de autenticación así que si solo vamos a usar FTP anónimo no deberíamos preocuparnos.

Lo primero que debemos hacer es generar un certificado. vsFTPD utiliza el formato pem así que lo generamos con la siguiente orden

```
openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /etc/ssl/private/vsftpd.pem -out /etc/ssl/private/vsftpd.pem
```

Habrà que editar el archivo de configuración

```
mv /etc/vsftpd.conf /etc/vsftpd.conf.bak  
  
sudo gedit /etc/vsftpd.conf
```

y escribir lo siguiente

```
listen=YES  
  
anonymous_enable=NO  
  
local_enable=YES  
  
write_enable=YES  
  
ssl_enable=YES
```

```
listen_port=990

#The following line enables implicit mode

implicit_ssl=YES

allow_anon_ssl=NO

#This will force secure data connections, not required, but recommended

force_local_data_ssl=YES

#This will force secure logins, not strictly required, but REALLY recommended

force_local_logins_ssl=YES

ssl_tlsv1=YES

ssl_sslv2=NO

ssl_sslv3=NO

#La siguiente línea es necesaria para que Filezilla acepte la conexión

ssl_ciphers=AES128-SHA

rsa_cert_file=/etc/ssl/private/vsftpd.pem
```

Los parámetros anteriores deberían ser bastante evidentes a estas alturas. Discute en clase qué hacen.



Posteriormente tendremos que reiniciar el servicio.

```
service vsftpd restart
```

## Un cliente en modo texto para FTPS

Ahora necesitamos un cliente que soporte este tipo de conexiones, para ello instalamos *lftp*, el más extendido en modo texto para Linux.

```
sudo apt-get update
```

```
sudo apt-get install lftp
```

ahora ya podemos probar nuestra conexión.

```
lftp ftps://localhost
```

```
lftp localhost:~> user prueba
```

```
Clave:
```

```
lftp prueba@localhost:~> ls
```

```
ls: Error fatal: Certificate verification: Not trusted
```

```
lftp prueba@localhost:~> set ssl:verify-certificate no
```

```
lftp prueba@localhost:~> ls
```

```
-rw-r--r--      1 1001      1002          8445 Apr 16  2012 examples.desktop
```

```
lftp prueba@localhost:~> quit
```

En la anterior prueba se utilizan unos comandos básicos, se pueden consultar todos con el comando *help* una vez conectados a lftp.

## Utilización de herramientas gráficas

El cliente gráfico más extendido en FTP es [Filezilla](#). Después de descargarlo y descomprimirlo ejecutamos el archivo bin/filezilla.

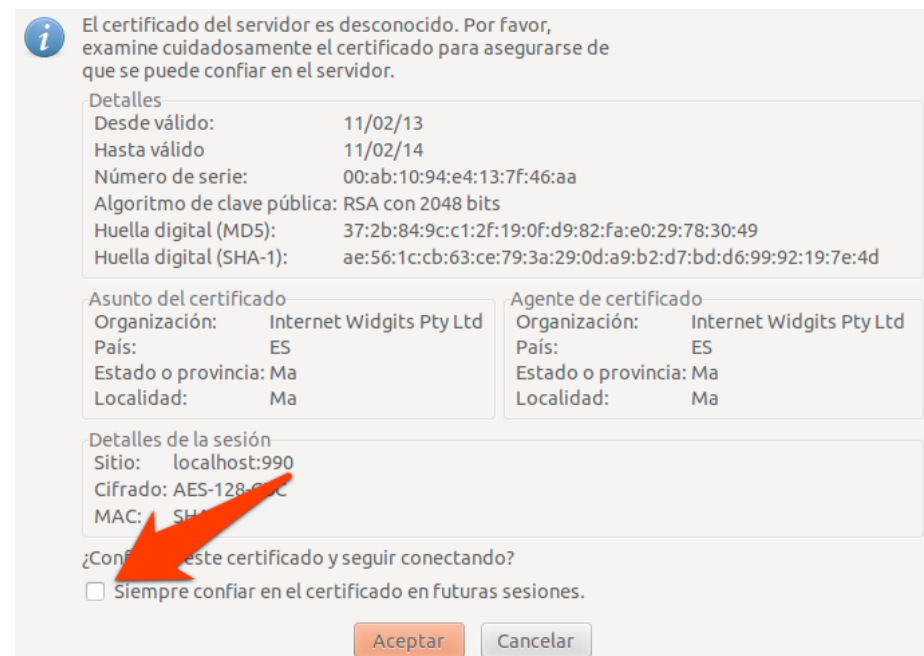


En la ventana que nos aparece lo primero es conectarnos al servidor, para lo que tenemos que establecer unos parámetros:

1. La url del servidor, con el protocolo *ftps* en caso de que queramos una conexión segura y el servidor lo permita.
2. El nombre de usuario.
3. La contraseña.
4. Un puerto en caso de que el servidor no esté configurado para recibir conexiones en los puertos estándar.
5. Pulsamos en el botón para conectarnos.

En el caso de establecer una conexión segura, si el certificado no es de confianza, nos aparecerá una ventana de aviso en la que veremos los datos del certificado y podremos marcar el certificado para que nos vuelva a preguntar si confiamos en él.

Una vez nos hemos conectado veremos que en la parte inferior tenemos una especie de explorador de archivos en el que en la parte de la izquierda aparece lo que tenemos en local y en la de la derecha las carpetas y



archivos en el servidor FTP. El funcionamiento es muy sencillo siendo la operación básica la de subir o bajar archivos para la que basta con arrastrar de un sitio a otro. Ten en cuenta que los permisos en las carpetas en ambos lados tienen que permitir que realicemos la acción que pretendemos llevar a cabo.

The screenshot shows the FileZilla interface with two panels. The left panel is labeled 'Sitio local: /home/prueba/' and shows a directory tree with folders like etc, home, prueba, sergio, and lib. The right panel is labeled 'Sitio remoto: /home/prueba' and shows a directory tree with folders like home, prueba, and subir. Below the directory trees are two tables showing file listings.

Nombre de archivo	Tamaño de archivo	Tipo de archivo	Última modificación
..		Directorio	12/02/13 09:00...
subir		Directorio	12/02/13 09:00...
.bash_logout	220	Archivo	03/04/12 17:58...
.bashrc	3.486	Archivo	03/04/12 17:58...
.profile	675	Archivo	03/04/12 17:58...
aSubir.txt	0	txt-archivo	12/02/13 09:04...
examples.d...	8.445	desktop-ar...	16/04/12 20:09...

1 archivo seleccionado. Tamaño total: 0 bytes

Nombre de archivo	Tamaño de archivo	Tipo de archivo	Última modificación	Permisos	Pro
..		Directorio	12/02/13 08...	drwxr-xr-x	0 0
subir		Directorio	12/02/13 08...	drwxr-xr-x	0 0
exam...	8.445	desktop-...	16/04/12	-rw-r-r-	100

1 archivo y 1 directorio. Tamaño total: 8.445 bytes

Instala Filezilla y prueba a conectarte mediante FTPS. Prueba a subir y bajar archivos. ¿Puedes eliminarlos o renombrarlos en el servidor? ¿Qué botón se utiliza para desconectarse?

## Utilización del servicio de transferencia de archivos desde el navegador

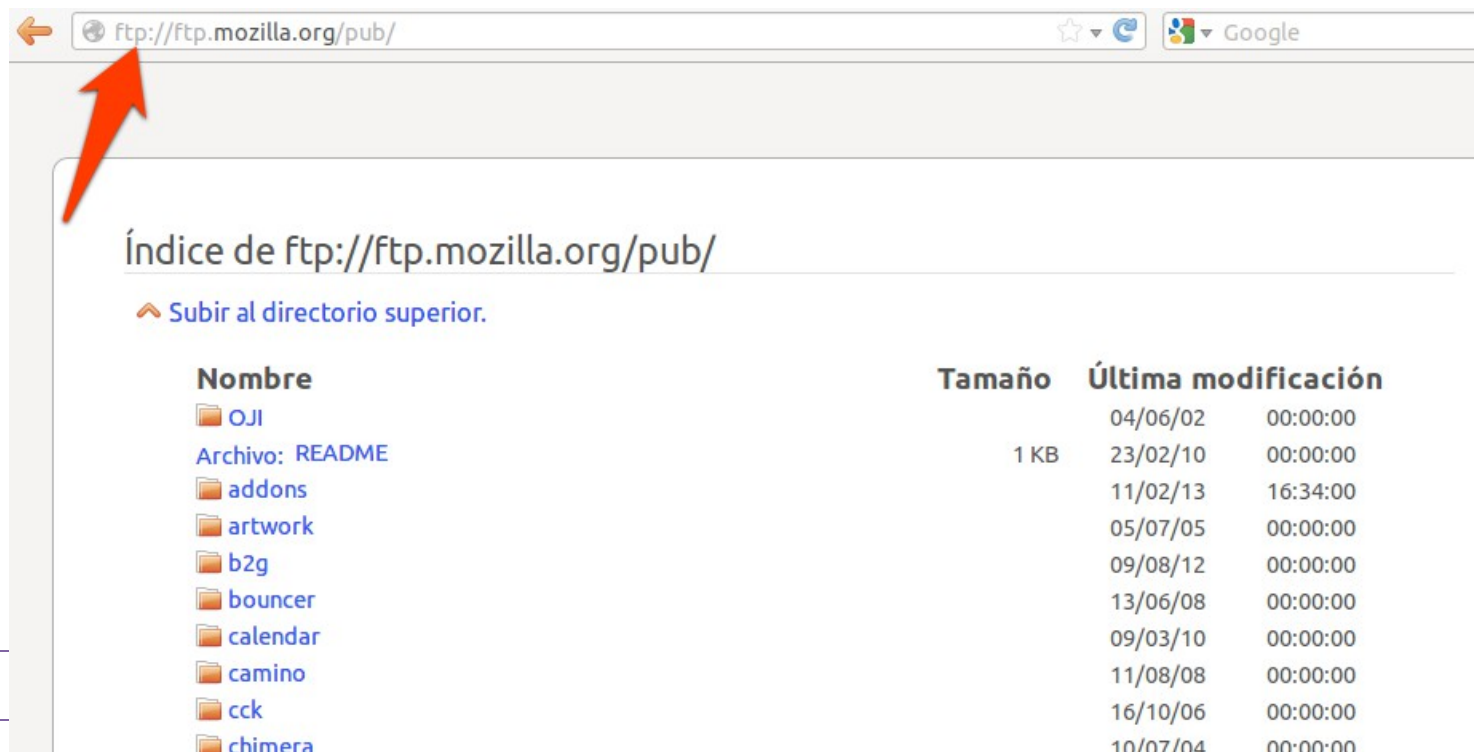
Los navegadores Web actuales soportan conexiones FTP sin embargo no suelen soportar FTPS o FTPES. Por ello revierto la configuración del servidor a FTP normal.

```
mv /etc/vsftpd.conf /etc/vsftpd.conf.ftpsh\n\nmv /etc/vsftpd.conf.bak /etc/vsftpd.conf\n\nsudo service vsftpd restart
```

Para conectarnos con una conexión anónima no hay más que poner <ftp://localhost> en la barra de direcciones, donde localhost será la dirección del sitio.

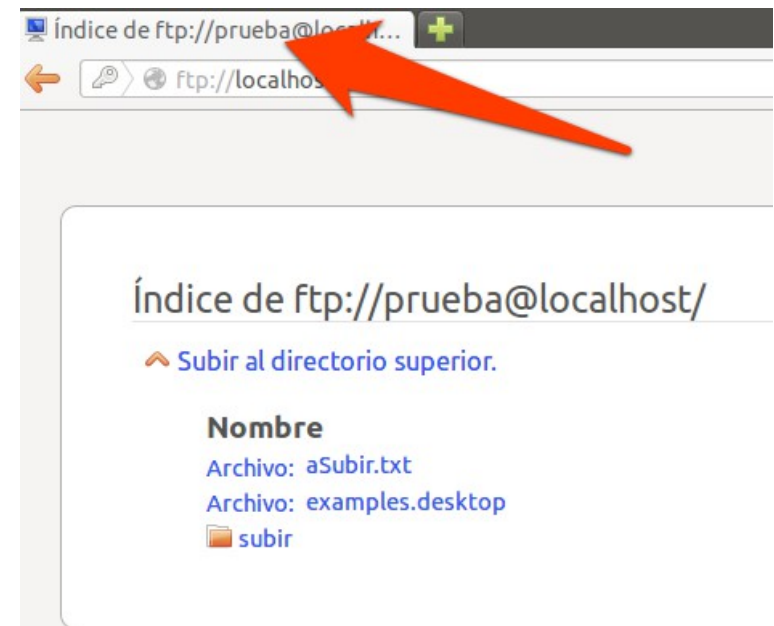
Pero para conectarte con un usuario se escribe <ftp://nombre-usr@sitio.web>

Por ejemplo si me conecto a <ftp://prueba@localhost> me pide la contraseña para el usuario y luego me muestra su carpeta, lo primero que vemos es que ya no aparece el



nombre del usuario en la barra de dirección pero sí en el título de la ventana/pestaña de navegación. Sin embargo los navegadores web no son clientes FTP y solo permiten navegar por las carpetas y descargar archivos, pero no subirlos.

Este último problema y el de las conexiones FTPS lo podemos solucionar mediante complementos como por ejemplo [FireFTP](#) para Firefox.



## Utilización del servicio de transferencia de archivos en el proceso de despliegue de la aplicación web

---

Este punto del temario ha quedado cubierto con lo visto en todos los anteriores. El despliegue de aplicaciones Web mediante FTP implica instalar un servidor FTP en la misma máquina en la que tengamos el servidor Web y/o de Aplicaciones y habilitar la carpeta de la aplicación para que se puedan conectar determinados usuarios con opciones para modificar el contenido. De esta forma podremos desarrollar toda la aplicación en una máquina de desarrollo y pruebas y subir la versión final y probada al servidor sin necesidad de acceso físico a él. Evidentemente para un desarrollador web que trabaje para muchos clientes esta opción es completamente vital.

## Desarrollo de operaciones remotas de gestión de contenidos: WebDAV

[WebDAV](#) (muchas veces se conoce como DAV a secas) es una extensión de HTTP que facilita el manejo de documentos en servidores web. Permite gestionar versiones y autoría de documentos de una forma distribuida. Como tantos proyectos desarrollados por grupos de trabajo en Internet, está en constante evolución.

Hay que tener en cuenta que WebDAV es un protocolo por lo que se puede ver como un sistema de ficheros distribuido con capacidades como por ejemplo bloquear recursos para su actualización evitando que otros puedan estar modificándolos a la vez. Al ser un protocolo permite que se creen aplicaciones sobre él para la gestión distribuida de documentos en la web. En el caso que nos importa tiene la gran utilidad de permitir que grupos de trabajo de gente en localizaciones físicas diferentes mantengan y actualicen una web o aplicación web minimizando los problemas que ello conlleva.

En el propio sitio del organismo puedes consultar una [lista de productos](#) que usan WebDAV, si le echas un vistazo te vas a sorprender.

En el fondo WebDAV no es más que una especie de conexión que te permite desarrollar en modo remoto accediendo a los recursos del servidor para modificarlos como si estuvieran en local. Para demostrar esto se puede consultar la siguiente lista de enlaces para que se vea cómo se configura la conexión al servidor en diferentes aplicaciones:

1. [Microsoft Office](#)
2. [Dreamweaver](#)
3. [Explorador de Windows](#)
4. [GNOME en Linux](#)
5. [MAC OS X](#)

En [esta otra página](#) lo explican para varias situaciones diferentes.

En la lista de productos puedes ver que hay muchas aplicaciones de edición de documentos (no necesariamente documentos Web) que usan esta tecnología e incluso es posible que la hayamos usado sin saberlo.

Apache tiene [un módulo](#) para poder habilitar WebDAV.



Instalar y configurar un sitio en Apache para que utilice WebDAV es sencillo. Es un tema perfecto para “aprender a aprender”. Aquí tienes un [pequeño tutorial](#) pero hay mucha información en Internet. Prueba a habilitar WebDAV en un sitio de Apache y a conectarte de diferentes maneras. Puedes probar a conectarte de dos formas diferentes e intentar editar el mismo archivo para ver si funciona el bloqueo de recursos.

## Tema 5: Servicios de red implicados en el despliegue de una aplicación web

## Resolutores de nombres

Internet y por extensión la gran mayoría de las redes de ordenadores en la actualidad utilizan el conjunto de protocolos TCP/IP. La capa de red de este conjunto es la que define las direcciones de los diferentes dispositivos conectados, conocidas como direcciones IP. En la versión actual (4) del protocolo IP las direcciones se componen de cuatro grupos o números de 8 bits y suelen representarse como cuatro puntos del 0 al 255 separados por puntos. No es el propósito de este módulo abundar en este tema, baste decir que lo que nosotros conocemos como “direcciones de Internet” o URLs no son más que asociaciones de un nombre más fácil de recordar o que se asocie a algo con una de estas direcciones IP.

El protocolo de este conjunto que se encarga de esta tarea es DNS (*Domain Name System*). El proceso de buscar la dirección IP asociada a un nombre de dominio determinado (por ejemplo [www.nba.com](http://www.nba.com)) se conoce como resolver una dirección.

Como todo lo que venimos viendo en este curso sigue un modelo Cliente/Servidor en el que los clientes (**resolutores** o *resolvers*) forman parte de los sistemas operativos de las máquinas y se encargan de contactar con los servidores para que les indiquen la dirección IP asociada a una URL cuando sea necesario. Hay que tener en cuenta que un servidor DNS puede ser cliente de otro servidor DNS.

Los servidores se conocen como Servidores de Nombres (*Name Servers*) y funcionan según el modelo jerárquico y distribuido:

- Jerárquico porque se organiza como un árbol invertido. En la raíz (dominio arpa que recibe su nombre del proyecto germen de Internet) estaría la autoridad principal que delega las terminaciones principales a organismos, entidades o países. Estas terminaciones incluyen elementos como .com, .org, .net, .gov, .es, .us, .uk, .fr etc. Los encargados de gestionar cada una de estas terminaciones (dominios de [nivel superior o top-level](#)) pueden a su vez subdividir y delegar el dominio. Por ejemplo en Gran Bretaña han creado uno comercial y su terminación es .co.uk y así .co sería una zona delegada dentro de .uk  
Un ejemplo completo de todo esto sería por ejemplo [www.ii.uam.es](http://www.ii.uam.es) en el que (de derecha a izquierda) se indica que es un dominio de España, de la Universidad Autónoma de Madrid, de la Escuela de Ingeniería Informática y dentro de eso que hace referencia al servidor web. En este último apartado puede haber otros como por ejemplo ftp. Las dos direcciones de ejemplo vistas arriba son *hostnames* ya que tienen una o más direcciones IP asociadas. Las partes de la derecha indican dominios o subdominios, pero no se asocian con direcciones concretas.
- Distribuido porque no es necesario que un resolutor DNS sea capaz de conocer todas las posibles asociaciones sino que puede conocer algunos y pasarle las que no conoce a otros servidores así como recibir a su vez peticiones de otros pares para resolver las que sí conoce. Esto lo veremos en más detalle en el proceso de resolución de nombres.

Como nota para el futuro, conviene decir que nos encontramos en un proceso de transición tanto en IP de la v4 a la v6 como en DNS en el que se pretende flexibilizar el conjunto de terminaciones y se están [estudiando posibles consideraciones](#) para incluir nuevos dominios de nivel superior.

---

## Proceso de resolución de un nombre de dominio

---

Vamos a ver el ejemplo con [www.ii.uam.es](http://www.ii.uam.es)

Un resolutor de nombres realiza su función mediante consultas a diferentes servidores. Siempre empieza por el nivel superior (el que está más a la derecha en la URL).

1. El **resolutor del cliente** le envía una petición a un servidor de nombres de la zona raíz (root zone) preguntando por ese subdominio de nivel superior (es). La dirección IP de al menos algunos servidores de la zona raíz está disponible mediante unos archivos que actualiza periódicamente una autoridad. Realmente el ordenador de nuestra casa necesita las direcciones IP de servidores DNS de un ISP (u otros organismos) como veremos más abajo. El servidor raíz le contesta con la dirección IP de quién gestiona el dominio de nivel superior .es (o con la información más completa que tenga que generalmente es a quién preguntar)
2. El cliente envía otra consulta a uno de los servidores de nivel superior que gestionan el subdominio .es preguntando por la dirección IP del servidor que gestiona .uam.es
3. El cliente redirige la consulta a dicho servidor preguntado por .ii.uam.es y contesta con su IP
4. Por último, se realiza una consulta de www a ii.uam.es que ya es un servicio determinado.
5. Con la IP de esta máquina la aplicación que solicitó la consulta (por ejemplo un navegador web) puede conectarse a su destino.

Este proceso se conoce como DNS estándar y saturaría Internet y los servidores de nombres por lo que en la práctica se utiliza combinado con un proceso de almacenado en **caché** de direcciones de Internet por todos los elementos implicados (desde el resolutor de nombres de nuestro ordenador) y lo que se denomina **DNS recursivo**.

El almacenamiento en caché se produce cada vez que se realiza una consulta. Los dispositivos por los que pasa la información almacenan toda la que van recibiendo (no solo la relacionada directamente con la URL o IP consultada).

El DNS recursivo básicamente permite que uno de estos elementos resuelva las IPs de dominios de otra gente. Si por ejemplo estás en el ordenador de tu casa y quieres buscar algo en google. Tecleas [www.google.es](http://www.google.es) y tu navegador primero intenta ver si él mismo sabe la IP de Google España. Si no la sabe se la pregunta al sistema operativo que si tampoco lo sabe la preguntará a los servidores de tu ISP (Internet Services Provider, por ejemplo

Telefónica o Jazztel) que si no lo saben ya recorrerán el camino contado anteriormente. La gran diferencia es que en este modelo si el servidor a quién preguntas no sabe la respuesta se encarga él de buscarla (y almacenarla en caché por si le hacen la misma consulta) haciendo de cliente que pregunta a otro servidor. Este proceso se repite hasta que se obtiene una respuesta concreta; generalmente la IP buscada pero también puede ser que la URL consultada no existe.

Este proceso no funcionaría bien sin un tiempo de vida (TTL) de las direcciones almacenadas en caché tras el que se marcarán como no válidas y se volverán a consultar con la primera petición que llegue.

Para todo este proceso se crean zonas directas que contienen registros SOA y NS y pueden contener cualquier otro menos PTR. Vemos los registros un poco más abajo.

### Reverse Lookup

También existe el proceso inverso (reverse DNS) por el que se pueden obtener los nombres de dominio asociados a una dirección IP determinada. Estas consultas siguen el mismo proceso, pero usan un dominio específico que se llama *in-addr.arpa* (*ip6.arpa* para IPv6). Las direcciones IP se representan invertidas como un nombre de dominio. Por ejemplo la dirección 123.45.67.89 se representaría como 89.67.45.123.in-addr.arpa

Primero se consultaría a quién está asociada 123 y así sucesivamente.

La zona que se debe crear en este caso se llama zona inversa y suele contener registros SOA, NS, PTR y CNAME.

El comando `nslookup` sirve para hacer consultas DNS. Funciona en Windows, MAC o Linux. Usando la propia ayuda del comando descubre cómo se realiza: una consulta directa, una inversa, una consulta a un servidor DNS concreto (no al por defecto)

## Parámetros de configuración y registros del servidor de nombres afectados en el despliegue

Como nos ha hecho falta en varias ocasiones a lo largo del curso, la instalación y configuración de un servidor web se [incluyó en los apéndices](#). Sin embargo faltaba la parte teórica que nos permitirá entender qué hacemos y por qué así como mejorar la configuración si fuera necesario.

En el apartado anterior ya hemos visto para qué se utilizan las zonas directa e inversa y por qué son necesarias. Además, podemos comprender para qué se utilizan los *forwarders* como medio para resolver las consultas que no conocemos, siendo estos *forwarders* generalmente servidores de nombres (DNS) de un ISP u otras organizaciones. Por ejemplo Google tiene servidores de nombres con las direcciones 8.8.8.8 y 8.8.4.4

Sin embargo hay una parte que no ha quedado nada clara y son los registros de los servidores DNS. Un *Resource Record* (RR) es el elemento más básico de DNS. Los registros definen las características de una zona o dominio y se usan en las consultas de nombres. Cada registro tiene un tiempo de vida (TTL *Time To Live*) que indica cuándo dejará de ser válido. En definitiva los registros se utilizan para asociar una URL a una IP.

Los registros tienen un formato estándar: *Owner TTL Class Type RDATA* donde

- **Owner:** es el nombre de la máquina o del dominio DNS al que pertenece este recurso.
- **TTL (*Time To Live*):** El tiempo en segundos que debe mantenerse este registro en caché. Si no aparece se usa el TTL mínimo del registro SOA.
- **Class:** Indica que familia de protocolos se usa. En Internet es casi exclusivamente IN.
- **Type:** El tipo del RR.
- **RDATA:** Los datos del recurso que indica el registro. Por ejemplo una dirección IP.

Hay muchos [tipos diferentes de registros](#) pero los más importantes son:

- **SOA:** Cualquier zona que se cree en un servidor DNS debe contener un registro SOA (*Start of Authority*) al inicio de la definición. Este tipo de registro contiene los parámetros:
  - Owner , TTL , Class y Type como vimos arriba.
  - Authoritative server (Servidor Autorizado): El servidor DNS primario de la zona.
  - Responsible person (persona responsable): correo electrónico de la persona responsable de la zona. Usa un punto en lugar de arroba.
  - Serial number (número de serie): muestra cuántas veces ha sido actualizada la zona.

- Refresh (refresco): indica como de frecuentemente tienen los servidores DNS secundarios de la zona que consultar si ésta ha cambiado.
- Retry (reintento): cuánto esperará un servidor secundario la respuesta antes de volver a consultar posibles cambios en la zona al servidor primario.
- Expire (expiración): cuánto tiempo desde la última actualización sigue siendo válida la información de un servidor secundario.
- Minimum TTL: El tiempo de vida que se aplica a todos los registros de la zona si no tienen uno específico. Ya hemos visto antes que cada registro puede llevar uno propio pero que es optativo.

En los apéndices se encuentran ejemplos de configuración y aquí podemos ver otro

```
prueba.ejemplo.com. IN SOA (  
  
    ns1.prueba.ejemplo.com. ; servidor autorizado para la zona  
  
    sergio.cuesta.prueba.ejemplo.com. ; e-mail del administrador de zona  
  
    5099 ; serial number  
  
    3600 ; refresh (1 hora)  
  
    600 ; retry (10 mins)
```

```
86400 ; expire (1 día)
```

```
60 ) ; minimum TTL (1 min)
```

- **NS:** Los registros NS indican los servidores de nombres autorizados para una zona. Pueden indicar servidores tanto primarios como secundarios y una zona tiene que tener al menos un registro NS con el servidor primario. En caso de haber zonas delegadas indican los servidores de dichas zonas también.

Así la siguiente línea tendría que aparecer en los servidores de nombres de ejemplo.com y prueba.ejemplo.com

```
prueba.ejemplo.com. IN NS ns1.prueba.ejemplo.com.
```

- **A:** Los registros de dirección (*Address*) asocian un nombre de dominio completo (*Fully Qualified Domain Name* o *FQDN*) con una dirección IP.

```
ns1 IN A 123.45.67.89
```

- **PTR:** Los registros puntero (*PoinTeR*) hacen lo contrario a los registros A. Asocian una dirección IP con el FQDN.

```
89.67.45.123.in-addr.arpa. IN PTR ns1.prueba.ejemplo.com.
```

```
o
```

```
89 IN PTR ns1.prueba.ejemplo.com.
```

- **CNAME:** Los registros de nombres canónicos crean un alias. Suelen utilizarse para ocultar la estructura de nuestro sitio. Por ejemplo si nosotros tenemos un servidor FTP que se llama ftp1 y luego lo cambiaremos a otro no queremos que los usuarios se enteren. También sirve para que varios FQDN se asocien a la misma máquina: varios host virtuales o varios servicios como ftp y www.



```
ftp.prueba.ejemplo.com. IN CNAME ftp1.prueba.ejemplo.com.
```

- **MX:** Este registro es específico para servidores de intercambio de correo (*Mail Exchange*). Un servidor de intercambio de correo se encarga de procesar y redirigir los correos de un dominio. Procesar el correo puede significar entregarlo a la dirección adecuada dentro de nuestro dominio o pasarlo a un tipo diferente de transporte de correo. Redirigir el correo significa enviárselo al servidor de correo de destino, directamente o a través de otros que se encuentren en el camino mediante *Simple Mail Transfer Protocol* (SMTP). Puedes tener varios servidores de correo en el mismo dominio por lo que es posible tener muchos registros MX.

```
*.prueba.ejemplo.com. IN MX 0 mailserver1.prueba.ejemplo.com.
```

```
*.prueba.ejemplo.com. IN MX 10 mailserver2.prueba.ejemplo.com.
```

```
*.prueba.ejemplo.com. IN MX 10 mailserver3.prueba.ejemplo.com.
```

El número que aparece después de MX es la prioridad. En este caso tenemos un servidor principal con la máxima prioridad y dos delegados con igual prioridad entre ellos. Si un correo llega y el primero no está activo se le pasaría al segundo o al tercero indistintamente. Si tenemos un gran tráfico de correo se pueden establecer todos con máxima prioridad para evitar saturaciones.

El comando `dig` nos muestra información de este tipo de recursos en una forma muy similar a la que usamos para configurar el servidor DNS. Investiga el comando y comenta sus opciones más útiles. Después contesta a las preguntas de más abajo.

¿Qué pasa si escribes `dig` sin más? ¿Cómo puedes indicarle a `dig` el servidor DNS que debe usar?

## Servicio de directorios: características y funcionalidad

El término “Servicios de Directorio” puede ser ambiguo y llevar a confusión. Una estructura de carpetas o volúmenes o diferentes bases de datos podrían considerarse dentro de la categoría, sin embargo un “Servicio de Directorio” en informática se considera que es algo similar a una guía telefónica o una lista de contactos donde los contactos no tienen por qué ser personas. Más concretamente se refiere a sistemas basados en X.500 o en alguna variante como LDAP.

Los servicios de directorio son sistemas que nos facilitan el acceso a recursos de nuestra red de manera universal. Son bases de datos de objetos que representan recursos tales como usuarios, grupos, carpetas o volúmenes, servidores, impresoras o aplicaciones.

Básicamente sirven para asociar el nombre de un recurso a su dirección de red, por lo que en este aspecto se parecen mucho a DNS. Un servicio de directorio define la estructura de los nombres o espacio de nombres (*namespace*) para la red: se establece un conjunto de normas que especifican cómo se compondrán los nombres y aseguran que no habrá duplicados ni repetidos. Estos nombres suelen conocerse como *Distinguished name* (DN) que en una traducción literal sería nombre distinguido. Si se cambia la dirección de un recurso esto solo implica cambiar la asociación en el objeto correspondiente y todo seguirá funcionando igual.

Sin embargo los servicios de directorio no solo incorporan esa funcionalidad. Como cada recurso está representado por un objeto, dichos objetos pueden tener atributos. Por ejemplo en el caso de usuarios se pueden incluir muchos otros datos como el nombre completo de la persona, su cargo en la organización, un teléfono de contacto, etc. Como es natural se pueden establecer permisos sobre estos objetos para asegurar que no todos los recursos o su información estarán disponibles para todo el mundo.

En definitiva un servicio de directorio nos permite unificar todos los recursos de la red sin importar de qué tipo son o los protocolos o topología específica de la red donde se encuentran. Basta con autenticarnos en la red para poder acceder a los recursos en los que tengamos permisos. Como además se pueden organizar de manera distribuida y replicada puede haber varios servidores proporcionando los mismos espacios de nombres y servir como sustitutos en caso de saturación o fallo.

En principio puede parecer que un sistema de directorio no es más que una base de datos relacional pero en un servicio de directorio se considera que la información va a leerse muchísimas más veces que escribirse ya que las actualizaciones o modificaciones serán poco frecuentes. Esto hace que determinados conceptos como las transacciones sean poco importantes y por lo tanto no se suelen incluir en el software de servicios de directorio. Por

lo tanto un servicio de directorio es una base de datos pero no en el sentido al que estamos acostumbrados de BdD relacional y está totalmente optimizada para las operaciones de lectura.

La especificación [X.500](#) define las líneas maestras de los servicios de directorio, sin embargo es tan compleja que ningún software actual cumple con todas sus condiciones. X.500 está pensando para funcionar con el modelo de pila de protocolos de red OSI de ISO, y concretó los aspectos que aquí nos interesan en un protocolo DAP (*Directory Access Protocol*). Sin embargo, aunque son relativamente similares ya sabemos que en la actualidad el conjunto de protocolos predominante en redes es TCP/IP y asociado a él se estableció un protocolo LDAP (*Lightweight Directory Access Protocol*) que como su nombre indica es más ligero y sencillo.

Cuando vimos certificados hablamos de X.509 que de hecho es parte de la familia de especificaciones de X.500

## LDAP

Mucha gente considera que X.500 es tan farragoso que nunca llegará a usarse completamente por lo que LDAP se creó con la idea de que fuera funcional. LDAP está bastante extendido y se utiliza en muchos casos así que será el que veamos.

*Lightweight Directory Access Protocol* implementa un servicio de nombres almacenando los objetos según una estructura jerárquica de árbol. La estructura que utiliza es similar a las de DNS o las carpetas y volúmenes en sistemas UNIX/Linux. Esta estructura de árbol se denomina *Data Information Tree* (DIT).

Los objetos que almacenan los datos en LDAP están compuestos de un nombre y otros atributos que contengan información asociada al objeto. Por ejemplo una persona con su nombre, teléfono, e-mail, etc. Estos objetos se denominan entradas (*Entry*). Estas entradas tienen un tipo o clase (*Class*) asociado al igual que pasa con los objetos en Programación Orientada a Objetos. Los atributos de una clase pueden ser obligatorios u optativos.

Es un protocolo de la capa de aplicación de TCP/IP al igual que HTTP, FTP, SMTP o DNS por ejemplo.

En la raíz de esta estructura de directorios se encuentra el DN (*Distinguished name*) base. Generalmente representa la organización cuyos recursos se van a gestionar mediante LDAP. Por ejemplo el nombre de una empresa u organismo. Por ejemplo en una empresa que se llamara “Ejemplo S.A.” éste podría ser el DN base o también podría usarse en nombre de dominio de la empresa: “ejemplo.com”. Ten en cuenta que puede ser un departamento de una empresa, una facultad de una universidad, etc., no es necesario que sea el organismo al completo.

Debajo del DN base aparecerán otros elementos de alguna manera organizada. Generalmente se utilizan agrupaciones lógicas. En el origen se asignaban estas agrupaciones a diferentes departamentos de una empresa (ventas, marketing, etc.) por lo que recibieron el nombre de Unidades Organizativas (*Organizational Units*, OU). En la actualidad, estas OU suelen corresponderse con tipos de recurso: usuarios, grupos, servidores, impresoras, volúmenes de red, etc.

¿Cómo es un DN? Cada DN consta de dos partes, su propio identificador y su ubicación en el árbol. Su identificador se llama *Relative Distinguished Name* (RDN) y es el nombre en el formato que queramos. Por ejemplo “Impresora a Color”. La segunda parte indica dónde se encuentra en un formato similar al que se utiliza en DNS; si mis impresoras están en una categoría “impresoras” el nombre completo será: Impresora a color – Impresoras – ejemplo.com

Como se puede ver el DN se construye con el RDN del elemento seguido del DN del padre que a su vez será su RDN seguido del DN del padre y así sucesivamente.

Esto siempre es más complejo con personas que con objetos inanimados. Por ejemplo para añadir a María Ramírez podríamos darle un RDN “María Ramírez” o su nombre de usuario “mramirez”. ¿Cuál es mejor? Depende de la situación.

La propia [página de la Wikipedia](#) sobre LDAP proporciona un poco más de información sobre las posibles operaciones a realizar en LDAP que nos pueden ayudar a comprender mejor el funcionamiento básico. Consúltalas y comenta en clase qué hacen.

Un vistazo [al software de servidor](#) que implementa LDAP nos lleva a ver que por ejemplo Active Directory de los servidores Windows los utilizan.

Como sucede con otros protocolos que hemos visto en LDAP hay varias versiones. Las predominantes en la actualidad son la 2 y 3. La versión 3 es compatible con clientes que soporten la versión 2 y es un proyecto de mejora incremental de la versión v2 al igual que está sucediendo con el desarrollo modular de HTML5 o CSS3.

LDAP es casi como la interfaz de una clase en Java, define cómo se representan los datos y qué operaciones con ellos se podrán realizar, pero no habla de la manera en la que se almacenarán los datos o de cualquier otro detalle interno de funcionamiento del servidor.

## Instalación de OpenLDAP

En la actualidad hay varios servidores gratuitos de LDAP. El que más antiguo y extendido es [OpenLDAP](#). El servidor coge su configuración inicial de la que tengamos en `/etc./hosts` así que habrá que editarlo pero antes cambiaremos el nombre del host.

```
gedit /etc/hostname
```

Y le doy el nombre “servidor”

```
gedit /etc/hosts
```

Y dejar

```
127.0.0.1    localhost
127.0.1.1    servidor.ejemplo.com servidor
```

Lo siguiente será instalar el servidor

```
sudo apt-get update
sudo apt-get install slapd ldap-utils
```

Durante la instalación se nos pide una contraseña de administrador, como siempre yo pondré “sergio”.

El paquete *slapd* es el servidor mientras que *ldap-utils* es un conjunto de aplicaciones de cliente

Un comando con el que podemos echar un vistazo general al servidor es **slapcat**

## Archivos básicos de configuración. Interpretación y uso

Podemos ver el archivo básico de la configuración del servidor

```
sudo gedit /etc/ldap/ldap.conf
```

Lo que muestra

```
#  
  
# LDAP Defaults  
  
#  
  
# See ldap.conf(5) for details  
  
# This file should be world readable but not world writable.  
  
#BASE dc=example,dc=com  
  
#URI ldap://ldap.example.com ldap://ldap-master.example.com:666  
  
#SIZELIMIT 12
```

```
#TIMELIMIT 15

#DEREF          never

# TLS certificates (needed for GnuTLS)

TLS_CACERT /etc/ssl/certs/ca-certificates.crt
```

Si te das cuenta la única línea no comentada es la última que indica el certificado que usa TLS en LDAP. Con los conocimientos adquiridos hasta la fecha debería ser transparente cómo crear otro certificado y usarlo con nuestro servidor pero no ahondaremos en el tema.

La base de datos de LDAP está configurada como un conjunto de archivos [LDIF](#) (*LDAP Data Interchange Format*) que se encuentran a partir de `/etc./ldap/slapd.d/`

Los archivos LDIF deben cumplir unas normas muy claras:

- Las líneas de comentarios empiezan por # y se ignoran al procesar el archivo
- Las líneas que empiezan por un único espacio en blanco se consideran una continuación de la anterior incluso si ésta última es un comentario
- Cada entrada se debe separar de la siguiente por una línea en blanco

Podemos consultar esta estructura mediante el protocolo LDAP con

```
sudo ldapsearch -Q -LLL -Y EXTERNAL -H ldapi:/// -b cn=config dn
```

El DN del administrador para el que introdujimos es `cn=admin,dc=ejemplo,dc=com`.

También podemos consultar el árbol de nuestro sitio

```
ldapsearch -x -LLL -H ldap:/// -b dc=ejemplo,dc=com dn
```



### Lo que muestra

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

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

Donde la primera línea es el origen del DIT y la segunda el DN del administrador como vimos un poco más arriba.

La opción `-x` sirve para usar autenticación simple, `-LLL` indica que no nos muestre detalles adicionales de LDIF, `-H` es el host y `-b` sirve para indicar a partir de qué nodo tiene que buscar.

### Así por ejemplo

```
ldapsearch -x -LLL -H ldap://servidor.ejemplo.com -b cn=admin,dc=ejemplo,dc=com dn
```

Nos mostraría solo el administrador porque no tiene hijos.

Como podemos ver esta forma de configurar OpenLDAP usa un árbol LDAP (DIT) y se maneja igual que el resto así que las operaciones que veamos para modificar el contenido de LDAP son compatibles con lo que aprendamos aquí. Además esto permite que se pueda modificar la configuración del servidor sin la necesidad de reiniciar en casi ningún caso.

Como veíamos un poco más arriba, la raíz del árbol de configuración se denomina `cn=config` que contiene los atributos de la configuración global, todo lo demás se encuentra en nodos hijo que pueden contener información de:

- Módulos cargados dinámicamente
- Definiciones de esquemas
- Configuración del back-end
- Configuración específica de la base de datos

Llegados a este punto es importante aclarar qué interpreta OpenLDAP como backend y qué como frontend:

- Frontend: La parte que se encarga del acceso a red y de procesar las órdenes relacionadas con el protocolo.

- Backend: La parte que se encarga del almacenamiento de los datos.

Sin embargo esto puede llevar a confusión ya que existe un elemento que se denomina base de datos “frontend” que sirve para aplicar configuraciones de forma global a todo el resto de las bases de datos.

El aspecto general del LDIF de configuración sería como sigue (el texto entre los símbolos <> se debería sustituir por configuración real):

```
# Configuración global

dn: cn=config

objectClass: olcGlobal

cn: config

<atributos de configuración global>


# sDefiniciones de esquemas

dn: cn=schema,cn=config

objectClass: olcSchemaConfig

cn: schema

<system schema>
```

```
dn: cn={X}core,cn=schema,cn=config
objectClass: olcSchemaConfig
cn: {X}core
<core schema>

# Más esquemas definidos por el usuario
...

# iniciones de backend
dn: olcBackend=<typeA>,cn=config
objectClass: olcBackendConfig
olcBackend: <typeA>
<configuración específica de backend>

# Definiciones de la base de datos
```

```
dn: olcDatabase={X}<typeA>,cn=config
objectClass: olcDatabaseConfig
olcDatabase: {X}<typeA>
<configuración específica de la base de datos>

# Más definiciones y configuraciones

...
```

Puedes ver que algunos atributos llevan un número {X} (donde X sería un número entero) antes del valor. Esto es debido a que en un LDAP no se especifica ningún orden así que se ha tenido que introducir una manera de especificar un orden determinado cuando es necesario que tal configuración se aplique siempre antes que tal otra.

Hasta la versión 2.2 (actualmente se usa la 2.4) OpenLDAP usaba un archivo de configuración *slapd.conf* que se ha sustituido por este otro sistema en el que las directivas de configuración con atributos de entradas en el DIT de configuración. Para facilitar la transición todas las directivas que se utilizaban en el archivo antiguo ahora llevan el prefijo “olc” en el nombre del atributo.

Para añadir, modificar o eliminar cláusulas de configuración se utilizan las mismas operaciones que veremos para usar el árbol específico que hemos creado. Son atributos y se modifican igual. En el apartado de control de acceso veremos ejemplos con directivas de configuración relacionadas con los permisos. Las cláusulas de configuración se pueden ver en la [documentación de OpenLDAP](#) pero a mi entender escapan al contenido del curso.

El servidor también contiene algunos esquemas preinstalados

```
# ls /etc/ldap/schema/

collective.ldif  cosine.schema  java.ldif  openldap.ldif
```

collective.schema	duaconf.ldif	java.schema	openldap.schema
corba.ldif	duaconf.schema	ldapns.schema	pmi.ldif
corba.schema	dyngroup.ldif	misc.ldif	pmi.schema
core.ldif	dyngroup.schema	misc.schema	ppolicy.ldif
core.schema	inetorgperson.ldif	nis.ldif	ppolicy.schema
cosine.ldif	inetorgperson.schema	nis.schema	README

Pueden consultarse estos esquemas abriendo los archivos

```
gedit /etc/ldap/schema/inetorgperson.ldif
```

Pero es más fácil yendo a la documentación de OpenLDAP. [Aquí vemos](#) el mismo que con el comando anterior y será uno de los que usaremos en los ejemplos que vayamos haciendo.

## Creando contenido en LDAP

Para poder autenticar usuarios usando LDAP primero tendremos que crearlos. Vamos a crear una estructura como se indica:

- Un nodo usuarios
- Un nodo grupos
- Un usuario
- Un grupo

Para ello creamos un archivo crea\_contenido.ldif

```
gedit crea_contenido.ldif
```

Con el contenido

```
dn: ou=Usuarios,dc=ejemplo,dc=com  
  
objectClass: organizationalUnit  
  
ou: Usuarios  
  
  
dn: ou=Grupos,dc=ejemplo,dc=com  
  
objectClass: organizationalUnit  
  
ou: Grupos
```

```
dn: cn=programadores,ou=Grupos,dc=ejemplo,dc=com
```

```
objectClass: posixGroup
```

```
cn: programadores
```

```
gidNumber: 5000
```

```
dn: uid=sergio,ou=Usuarios,dc=ejemplo,dc=com
```

```
objectClass: inetOrgPerson
```

```
objectClass: posixAccount
```

```
objectClass: shadowAccount
```

```
uid: sergio
```

```
sn: Cuesta
```

```
givenName: Sergio
```

```
cn: Sergio Cuesta
```

```
displayName: Sergio Cuesta
```

```
uidNumber: 10000

gidNumber: 5000

userPassword: sergio

gecos: Sergio Cuesta

loginShell: /bin/bash

homeDirectory: /home/scuesta

mail: sergio.cuesta@ejemplo.com

telephoneNumber: 55555555

st: Madrid
```

Ahora ejecutamos la orden para añadir el contenido creado en el archivo lo que nos pide la contraseña y nos informa de qué se ha creado

```
# ldapadd -x -D cn=admin,dc=ejemplo,dc=com -W -f crea_contenido.ldif

Enter LDAP Password:

adding new entry "ou=Usuarios,dc=ejemplo,dc=com"

adding new entry "ou=Grupos,dc=ejemplo,dc=com"
```



```
adding new entry "cn=programadores,ou=Grupos,dc=ejemplo,dc=com"
```

```
adding new entry "uid=sergio,ou=Usuarios,dc=ejemplo,dc=com"
```

Ahora que ya tenemos contenido podemos buscar datos dentro de LDAP

```
# ldapsearch -x -LLL -b dc=ejemplo,dc=com 'uid=sergio'
```

```
dn: uid=sergio,ou=Usuarios,dc=ejemplo,dc=com
```

```
objectClass: inetOrgPerson
```

```
objectClass: posixAccount
```

```
objectClass: shadowAccount
```

```
uid: sergio
```

```
sn: Cuesta
```

```
givenName: Sergio
```

```
cn: Sergio Cuesta
```

```
displayName: Sergio Cuesta
```

```
uidNumber: 10000  
gidNumber: 5000  
gecos: Sergio Cuesta  
loginShell: /bin/bash  
homeDirectory: /home/scuesta  
mail: sergio.cuesta@ejemplo.com  
telephoneNumber: 55555555  
st: Madrid
```

También podemos consultar atributos determinados

```
# ldapsearch -x -LLL -b dc=ejemplo,dc=com 'uid=sergio' uid displayName mail  
  
dn: uid=sergio,ou=Usuarios,dc=ejemplo,dc=com  
  
uid: sergio  
  
displayName: Sergio Cuesta  
  
mail: sergio.cuesta@ejemplo.com
```

O buscar el atributo “uid” de todas las entradas

```
# ldapsearch -x -LLL -b dc=ejemplo,dc=com uid  
  
dn: dc=ejemplo,dc=com  
  
dn: cn=admin,dc=ejemplo,dc=com  
  
dn: ou=Usuarios,dc=ejemplo,dc=com  
  
dn: ou=Grupos,dc=ejemplo,dc=com  
  
dn: cn=programadores,ou=Grupos,dc=ejemplo,dc=com  
  
dn: uid=sergio,ou=Usuarios,dc=ejemplo,dc=com  
  
uid: sergio
```

Instala OpenLDAP en tu servidor y añade información sobre ti como usuario. Prueba a realizar consultas de contenido. ¿Qué sucede si eliminas el parámetro -LLL en tus búsquedas? ¿Consideras más útiles las búsquedas así o con el parámetro -LLL? ¿Por qué?

Una vez hayas terminado, crea otro archivo para añadir un segundo usuario y ejecútalo. Prueba nuevas búsquedas.

Existen muchísimas opciones de búsqueda y no es el objetivo de este curso profundizar mucho en LDAP, pero para poder poner un par de ejemplos más de búsquedas voy a añadir otro usuario

```
gedit crea_maria.ldif
```

Con el contenido

```
dn: uid=maria,ou=Usuarios,dc=ejemplo,dc=com  
  
objectClass: inetOrgPerson  
  
objectClass: posixAccount  
  
objectClass: shadowAccount  
  
uid: maria  
  
sn: Ramírez  
  
givenName: María  
  
cn: María Ramírez  
  
displayName: María Ramírez  
  
uidNumber: 10001
```

```
gidNumber: 5001

userPassword: maria

gecos: Maria Ramirez

loginShell: /bin/bash

homeDirectory: /home/mramirez

mail: maria.ramirez@ejemplo.com

telephoneNumber: 66666666

st: Madrid
```

Lo añadimos

```
ldapadd -x -D cn=admin,dc=ejemplo,dc=com -W -f crea_maria.ldif
```

Y buscamos los usuarios del estado Madrid

```
# ldapsearch -x -LLL -b dc=ejemplo,dc=com "st=Madrid" displayName

dn: uid=sergio,ou=Usuarios,dc=ejemplo,dc=com

displayName: Sergio Cuesta
```

```
dn: uid=maria,ou=Usuarios,dc=ejemplo,dc=com
```

```
displayName:: TWfYw6lhIFJhbcOtcMv6
```

Si te das cuenta, no le han gustado las tildes en el nombre de María. Luego mostraremos cómo modificar atributos.

Podemos buscar todos los nodos dentro de Usuarios

```
# ldapsearch -x -LLL -b ou=Usuarios,dc=ejemplo,dc=com uid
```

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

```
dn: uid=sergio,ou=Usuarios,dc=ejemplo,dc=com
```

```
uid: sergio
```

```
dn: uid=maria,ou=Usuarios,dc=ejemplo,dc=com
```

```
uid: maria
```

O buscar los nodos con estado Madrid y uid maria

```
# ldapsearch -x -LLL -b dc=ejemplo,dc=com "(&(st=Madrid)(uid=maria))"
```

```
dn: uid=maria,ou=Usuarios,dc=ejemplo,dc=com
```

```
objectClass: inetOrgPerson
objectClass: posixAccount
objectClass: shadowAccount
uid: maria
sn:: UmFtw61yZXo=
givenName:: TWFyw61h
cn:: U2VyZ2lvIFJhbcOtcMv6
displayName:: TWFyw61hIFJhbcOtcMv6
uidNumber: 10001
gidNumber: 5001
gecos: Maria Ramirez
loginShell: /bin/bash
homeDirectory: /home/mramirez
mail: maria.ramirez@ejemplo.com
telephoneNumber: 66666666
```

```
st: Madrid
```

Por último buscaremos los que tengan uid maria o sergio

```
# ldapsearch -x -LLL -b dc=ejemplo,dc=com "(|(uid=sergio)(uid=maria))" gecos
```

```
dn: uid=sergio,ou=Usuarios,dc=ejemplo,dc=com
```

```
gecos: Sergio Cuesta
```

```
dn: uid=maria,ou=Usuarios,dc=ejemplo,dc=com
```

```
gecos: Maria Ramirez
```

---

## Modificando el contenido

Vamos a ver cómo añadir, borrar o modificar el contenido de atributos de entidades LDAP. Empezaremos creando un archivo con los cambios.

```
gedit modifica_contenido.ldif
```

Con las operaciones de modificación

```
dn: uid=sergio,ou=Usuarios,dc=ejemplo,dc=com
```

```
add: mobile
```

```
mobile: 77777777
```



```
dn: uid=sergio,ou=Usuarios,dc=ejemplo,dc=com
```

```
delete: telephoneNumber
```

```
dn: uid=maria,ou=Usuarios,dc=ejemplo,dc=com
```

```
changetype: modify
```

```
replace: sn
```

```
sn: Ramirez
```

```
dn: uid=maria,ou=Usuarios,dc=ejemplo,dc=com
```

```
changetype: modify
```

```
replace: givenName
```

```
givenName: Maria
```

```
dn: uid=maria,ou=Usuarios,dc=ejemplo,dc=com
```

```
changetype: modify  
  
replace: cn  
  
cn: Maria Ramirez  
  
dn: uid=maria,ou=Usuarios,dc=ejemplo,dc=com  
  
changetype: modify  
  
replace: displayName  
  
displayName: Maria Ramirez
```

Y lo usamos

```
ldapmodify -x -D cn=admin,dc=ejemplo,dc=com -W -f modifica_contenido.ldif
```

Podemos ver cómo se han aplicado los cambios

```
# ldapsearch -x -LLL -b ou=Usuarios,dc=ejemplo,dc=com  
  
dn: ou=Usuarios,dc=ejemplo,dc=com  
  
objectClass: organizationalUnit  
  
ou: Usuarios
```

```
dn: uid=sergio,ou=Usuarios,dc=ejemplo,dc=com
objectClass: inetOrgPerson
objectClass: posixAccount
objectClass: shadowAccount
uid: sergio
sn: Cuesta
givenName: Sergio
cn: Sergio Cuesta
displayName: Sergio Cuesta
uidNumber: 10000
gidNumber: 5000
gecos: Sergio Cuesta
loginShell: /bin/bash
homeDirectory: /home/scuesta
```

```
mail: sergio.cuesta@ejemplo.com

st: Madrid

mobile: 77777777

dn: uid=maria,ou=Usuarios,dc=ejemplo,dc=com

objectClass: inetOrgPerson

objectClass: posixAccount

objectClass: shadowAccount

uid: maria

uidNumber: 10001

gidNumber: 5001

gecos: Maria Ramirez

loginShell: /bin/bash

homeDirectory: /home/mramirez

mail: maria.ramirez@ejemplo.com
```

```
telephoneNumber: 66666666  
  
st: Madrid  
  
sn: Ramirez  
  
givenName: Maria  
  
cn: Maria Ramirez  
  
displayName: Maria Ramirez
```

Otra posible modificación sería añadir los usuarios al grupo creado. Creamos otro archivo

```
gedit aniade_grupos.ldif
```

Con el contenido

```
dn: cn=programadores,ou=Grupos,dc=ejemplo,dc=com  
  
changetype: modify  
  
add: memberuid  
  
memberuid: cn=sergio,ou=Usuarios,dc=ejemplo,dc=com  
  
dn: cn=programadores,ou=Grupos,dc=ejemplo,dc=com
```

```
changetype: modify  
  
add: memberuid  
  
memberuid: cn=maria,ou=Usuarios,dc=ejemplo,dc=com
```

Y lo usamos

```
ldapmodify -x -D cn=admin,dc=ejemplo,dc=com -W -f aniade_grupos.ldif
```

Podemos consultar la entrada para ver que todo ha ido bien

```
# ldapsearch -x -LLL -b cn=programadores,ou=Grupos,dc=ejemplo,dc=com  
  
dn: cn=programadores,ou=Grupos,dc=ejemplo,dc=com  
  
objectClass: posixGroup  
  
cn: programadores  
  
gidNumber: 5000  
  
memberUid: cn=sergio,ou=Usuarios,dc=ejemplo,dc=com  
  
memberUid: cn=maria,ou=Usuarios,dc=ejemplo,dc=com
```

## Borrando entidades

Hemos aprendido a crear y modificar entidades, solo nos falta ver cómo eliminarlas. Para ello vamos a crear un archivo que nos elimine todas las entidades que creamos con el archivo crea\_contenido.ldif. Ten en cuenta que los archivos siguen existiendo por lo que puedes volver a ejecutarlos en cualquier momento. De todas formas los usuarios que hemos creado son compatibles con los de Linux. Si buscas cómo crear usuarios en Internet verás que hay otras clases que se le pueden asignar, no es necesario que tomen las que hemos determinado nosotros.

Creamos un archivo borra\_contenido.ldif con lo siguiente

```
uid=sergio,ou=Usuarios,dc=ejemplo,dc=com  
  
uid=maria,ou=Usuarios,dc=ejemplo,dc=com  
  
cn=programadores,ou=Grupos,dc=ejemplo,dc=com  
  
ou=Usuarios,dc=ejemplo,dc=com  
  
ou=Grupos,dc=ejemplo,dc=com
```

Y lo usamos

```
ldapdelete -x -D cn=admin,dc=ejemplo,dc=com -W -f borra_contenido.ldif
```

Ahora tenemos la base de datos LDAP como al principio.

## Cientes gráficos

Existe un gran número de clientes con interfaz gráfica para LDAP. Los más conocidos son:

- [JXplorer](#): El que vamos a usar.
- [Apache Directory Studio](#): Un plugin para Eclipse. Muy utilizado entre los desarrolladores que usan este entorno.
- [phpLDAPadmin](#): Un cliente desarrollado en PHP con la ventaja de que se ejecuta en cualquier navegador web.

Para instalar JXplorer lo más fácil es hacerlo desde línea de comandos

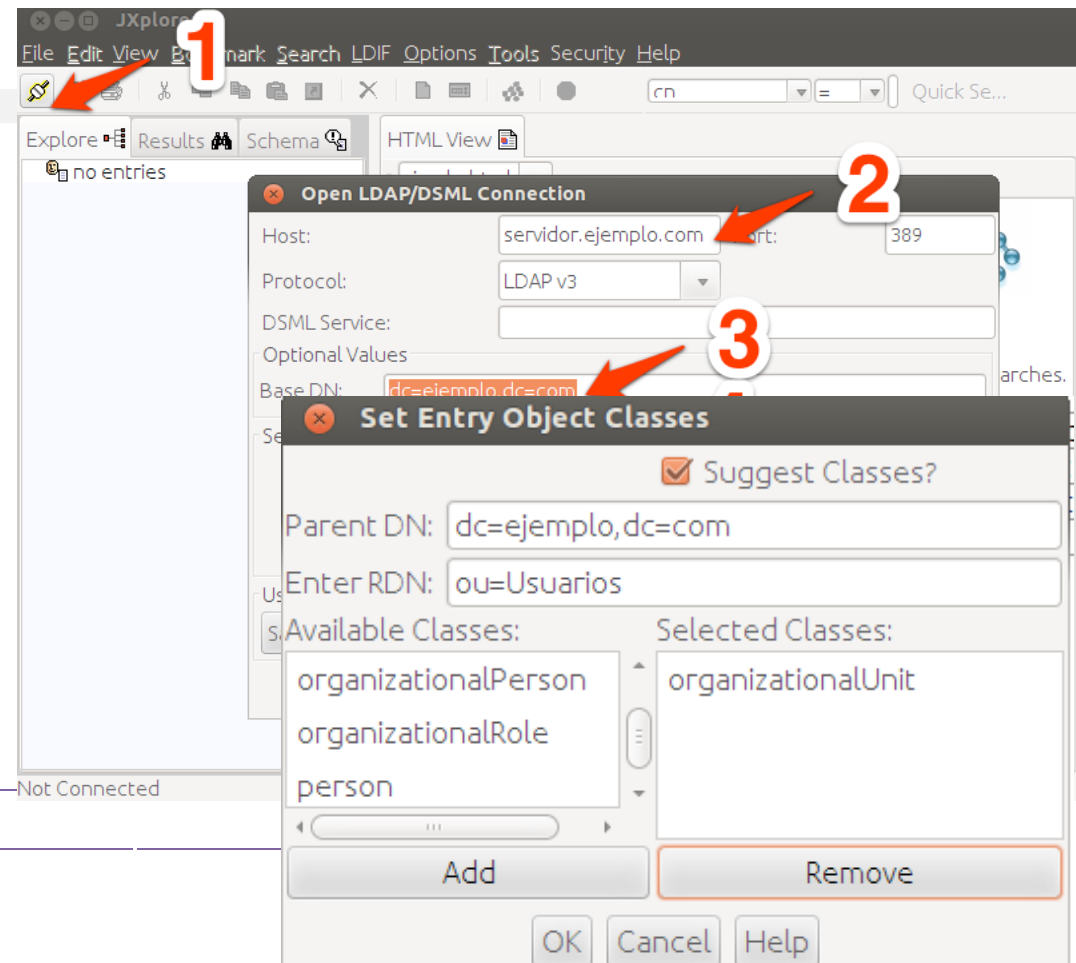
```
sudo apt-get install jxplorer
```

Y se ejecuta como cualquier otra aplicación.

Una vez abrimos la aplicación hay que conectarse:

1. Pulsamos el botón de conectar
2. Escribimos la dirección del servidor LDAP
3. Escribimos los “dc”
4. Elegir autenticación de usuario
5. Escribir los datos del usuario que se conectará
6. Ponemos la contraseña
7. Pulsamos OK

Una vez nos hemos conectado ya vemos el árbol de nuestro servidor.





Mediante *Edit* -> *New* o con el botón derecho *New* podemos crear nuevas entidades. Cuando hemos seleccionado los parámetros y pulsamos OK se nos muestra a la derecha la tabla con los atributos de la entidad pero no aparecerá en el árbol hasta que no pulsemos *Submit*.

Ayudándote de Internet aprende a crear, modificar y borrar entidades en nuestro árbol y replica la estructura que hicimos desde línea de comandos.

## Autenticación de usuarios en el servicio de directorios

El acceso al servidor LDAP y el DIT se controla mediante directivas que se representan como atributos en los propios nodos del árbol. Podemos ver un caso

```
# sudo ldapsearch -Q -LLL -Y EXTERNAL -H ldapi:/// -b \
> cn=config '(olcDatabase={0}config)' olcAccess
dn: olcDatabase={0}config,cn=config
olcAccess: {0}to * by dn.exact=gidNumber=0+uidNumber=0,cn=peercred,cn=external
,cn=auth manage by * break
```

Ten en cuenta que estamos hablando del acceso al servidor LDAP no del acceso mediante usuarios creados en LDAP a servidores web o de aplicaciones, de lo que hablaremos más tarde.

Podemos ver que para el control de acceso se utiliza el atributo `olcAccess`. En la [documentación de OpenLDAP](#) puedes ver las opciones del atributo (es bastante complejo). Debes ir al punto 8.3 ya que la primera parte habla de cómo se configuraba con el archivo de configuración (método antiguo). Nosotros vamos a centrarnos en algunos puntos clave.

Por defecto los servidores LDAP garantizan permisos de lectura a todos los nodos a cualquier usuario. Esto no es tan raro como parece: hay que tener en cuenta que se trata de un servicio de directorio cuya función es facilitar el acceso a recursos (ya sean personas, impresoras, servidores, etc.). Para entender mejor que no es raro podemos pensar en qué sucedería si para realizar operaciones de lectura en DNS fuera necesario tener un usuario y contraseña autorizados.

Como en esta sección vamos a aprender a modificar los permisos para que solo determinados usuarios pueden acceder a partes de la información, conviene aclarar aquí que igual que hemos hecho para las operaciones de modificación del contenido se puede utilizar un usuario a la hora de realizar búsquedas o consultas

```
ldapsearch -x -D cn=admin,dc=ejemplo,dc=com -W -LLL -b dc=ejemplo,dc=com
```

## El atributo olcAccess

Para usar el atributo olcAccess hay que fijarse en:

- A qué (what): Entradas (Entries) para las que estableceremos los permisos.
- A quién (who): Usuarios a los que se les aplican los permisos.
- Permisos (Access to grant): qué tipo de acceso se le va a permitir.

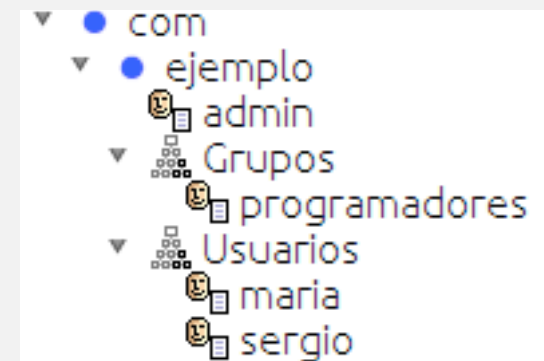
Para entender mejor los ejemplos conviene recordar que la estructura de nuestro DIT es como se ve en la imagen o lo que es lo mismo

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

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

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

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



```
dn: cn=programadores,ou=Grupos,dc=ejemplo,dc=com
```

```
dn: uid=sergio,ou=Usuarios,dc=ejemplo,dc=com
```

```
dn: uid=maria,ou=Usuarios,dc=ejemplo,dc=com
```

### A qué

Aunque existe una tercera forma solo vamos a considerar dos maneras de especificar las entradas objetivo:

- to \*
- to dn.<scope-style>=<DN>

La primera forma sirve para seleccionar todas las entradas.

La segunda forma nos permite seleccionar algunas en particular pero *scope-style* nos permite aprovechar la estructura arbórea de la base de datos LDAP. Puede ser “base” que se aplica solo a la entrada que coincida, “one” que se aplica a los hijos de la entrada que coincida, “subtree” que se aplica a todo el árbol cuya raíz fuera la entrada que coincida o “children” que se aplicaría al mismo árbol que en el caso anterior pero exceptuando la raíz.

Lo vamos a ver con ejemplos:

1. dn.base="ou=Usuarios,dc=ejemplo,dc=com" solo se aplicaría al nodo “Usuarios”;
2. dn.one="ou=Usuarios,dc=ejemplo,dc=com" se aplicaría a “sergio” y “maria”;
3. dn.subtree="ou=Usuarios,dc=ejemplo,dc=com" se aplicaría a “Usuarios”, “sergio” y “maria”
4. dn.children="ou=Usuarios,dc=ejemplo,dc=com" se aplicaría a “sergio” y “maria” pero a diferencia del ejemplo 2 si alguno tuviera hijos también se aplicaría a todos ellos al igual que sucedería en el ejemplo 3;

También se pueden usar filtros como los que hemos utilizado en búsquedas. Así

to filter=(st=Madrid) se aplicaría a “sergio” y “maria” porque ambos tienen ese valor en el atributo st.

Por último conviene destacar que se pueden seleccionar determinados atributos

to attrs=atributo1, atributo2

### A quién

Se pueden especificar varios tipos:

- \* : Cualquier usuario ya sea anónimo o autenticado
- anonymous : solo usuarios anónimos
- users: solo usuarios autenticados
- self: el usuario asociado con la entrada indicada en “a qué”
- dn.<scope-style>=<DN>: Los usuarios que cumplan con lo definido como se explicó en el punto anterior

Con los ejemplos completos del final se entenderá mejor esto.

### Permisos

Hay varios niveles. En la lista a continuación lo que aparece después del igual es el equivalente ya que cada nivel implica que se tienen los permisos del anterior. Detrás de los dos puntos va la explicación. Se incluyen todas las posibilidades pero a nosotros solo nos interesan “none”, “read”, “write” y “manage”

1. none =o : sin acceso
2. disclose =d : para poder obtener información si se produce un error
3. auth =dx : necesario para autenticarse
4. compare =cdx : necesario para comparar
5. search =scdx : necesario para aplicar filtros en las búsquedas
6. read =rscdx : para poder leer los resultados de las búsquedas
7. write =wrscdx : para poder modificar o cambiar nombres
8. manage =mwrscdx : necesario para administrar

## Orden

Las directivas se aplican en el orden en el que aparecen. Este orden se puede forzar mediante números entre llaves {X}. Es importante tener en cuenta que se sigue una política de aplicar el primero que coincida así que si por ejemplo aplicamos algo a los usuarios anónimos y luego algo a todos esta última parte no se aplicará a los anónimos porque ya han coincidido con la entrada anterior. Esto y muchas más cosas se ven más claras en los ejemplos.

Existen muchas más opciones de control de acceso como se comenta en [la documentación](#) de OpenLDAP. También hay algunos ejemplos interesantes en [esta otra página](#). Por último [aquí hay una explicación](#) muy de “andar por casa” con ejemplos de aplicación práctica directa.

Consulta y discute en clase cada ejemplo de la sección 8.3.5 del primer enlace del párrafo anterior, los más relevantes del segundo enlace y todos los del tercero.

---

## Contraseñas para los usuarios

Es necesario saber cómo añadir contraseñas a los usuarios porque hasta ahora los hemos usado como simples elementos de una base de datos pero evidentemente si se van a poder conectar a aplicaciones o al servidor LDAP necesitarán de una contraseña.

La forma más sencilla es añadirles una contraseña mediante una modificación del usuario. En un archivo pwd.ldif

```
dn: uid=sergio,ou=Usuarios,dc=ejemplo,dc=com
changetype: modify
add: userPassword
userPassword: unaPwd
```

Que usamos

```
ldapmodify -x -D cn=admin,dc=ejemplo,dc=com -W -f pwd.ldif
```

Si buscamos los datos del usuario no nos muestra la contraseña

```
# ldapsearch -x -LLL -b ou=Usuarios,dc=ejemplo,dc=com "uid=sergio"
```

```
dn: uid=sergio,ou=Usuarios,dc=ejemplo,dc=com
```

```
objectClass: inetOrgPerson
```

```
objectClass: posixAccount
```

```
objectClass: shadowAccount
```

```
uid: sergio
```

```
sn: Cuesta
```

```
givenName: Sergio
```

```
cn: Sergio Cuesta
```

```
displayName: Sergio Cuesta
```

```
uidNumber: 10000
```

```
gidNumber: 5000
```

```
gecos: Sergio Cuesta  
loginShell: /bin/bash  
homeDirectory: /home/scuesta  
mail: sergio.cuesta@ejemplo.com  
telephoneNumber: 55555555  
st: Madrid
```

Ahora podemos modificar la contraseña desde línea de comandos

```
# ldappasswd -h servidor.ejemplo.com -D "uid=sergio,ou=Usuarios,dc=ejemplo,dc=com" -W -S  
"uid=sergio,ou=Usuarios,dc=ejemplo,dc=com"  
  
New password:  
  
Re-enter new password:  
  
Enter LDAP Password:
```

Debido a la configuración que tenemos en el servidor LDAP no es posible conectarse sin contraseña por lo que si quisiéramos que maria también tuviera contraseña tendríamos que hacer lo mismo. Lo mejor sería crear los usuarios con una contraseña tonta y que fuera siempre la misma y automáticamente hacer que la cambiaran.



## Aplicando las directivas

Hasta ahora hemos visto cómo funciona el control de acceso en general y hemos hablado de que aparecen como atributos en el árbol LDAP, pero ¿cómo y a qué elementos se le aplican? Eso es lo que vamos a ver en este punto.

El objeto (o entrada) a la que se le añaden los atributos `olcAccess` es al que representa la configuración de la base de datos. Estos objetos se guardan a partir del directorio `/etc/ldap/slapd.d/` como ya vimos. Más concretamente las bases de datos cuelgan del subdirectorio `cn=config`. En él hay varios archivos que hacen referencia a `olcDatabase` pero suele ser el siguiente:

```
gedit /etc/ldap/slapd.d/cn\=config/olcDatabase\=\{1\}hdb.ldif
```

Podemos asegurarnos abriendo el archivo pero SIN HACER NINGUNA MODIFICACIÓN y buscando el parámetro `olcSuffix` para comprobar que coincide con el DN de base de nuestro árbol.

```
# AUTO-GENERATED FILE - DO NOT EDIT!! Use ldapmodify.

# CRC32 532fe58d

dn: olcDatabase={1}hdb

objectClass: olcDatabaseConfig

objectClass: olcHdbConfig

olcDatabase: {1}hdb

olcDbDirectory: /var/lib/ldap

olcSuffix: dc=ejemplo,dc=com
```

...

Una forma más segura y probablemente mejor localizar el nodo mediante una búsqueda

```
sudo ldapsearch -Q -LLL -Y EXTERNAL -H ldapi:/// -b cn=config olcSuffix
```

Lo que nos va a mostrar el resultado ya esperado

```
dn: cn=config
```

```
dn: cn=module{0},cn=config
```

```
dn: cn=schema,cn=config
```

```
dn: cn={0}core,cn=schema,cn=config
```

```
dn: cn={1}cosine,cn=schema,cn=config
```

```
dn: cn={2}nis,cn=schema,cn=config
```

```
dn: cn={3}inetorgperson,cn=schema,cn=config
```

```
dn: olcBackend={0}hdb,cn=config
```

```
dn: olcDatabase={-1}frontend,cn=config
```

```
dn: olcDatabase={0}config,cn=config
```

```
dn: olcDatabase={1}hdb,cn=config
```

```
olcSuffix: dc=ejemplo,dc=com
```

Con este DN ya sabemos a qué elemento aplicar los cambios. Lo primero que deberíamos hacer es consultar qué `olcAccess` tiene

```
# sudo ldapsearch -Q -LLL -Y EXTERNAL -H ldapi:/// -b olcDatabase={1}hdb,cn=config olcAccess
```

```
dn: olcDatabase={1}hdb,cn=config
```

```
olcAccess: {0}to attrs=userPassword,shadowLastChange
```

```
by self write

by anonymous auth

by dn="cn=admin,dc=ejemplo,dc=com" write

by * none

olcAccess: {1}to dn.base=""

by * read

olcAccess: {2}to *

by self write

by dn="cn=admin,dc=ejemplo,dc=com" write

by * read
```

He modificado un poco la salida para facilitar la lectura

Comenta en clase qué permisos hay configurados por defecto

Ahora que ya tenemos todos los factores solo hace falta recordar que se modifica como cualquier otra entrada del árbol LDAP por lo que lo primero sería crear un archivo “permisos.ldif” con las modificaciones. Yo voy a añadir un nuevo `olcAccess` pero se pueden modificar o eliminar como ya vimos.

```
dn: olcDatabase={1}hdb,cn=config
```

```
delete: olcAccess

dn: olcDatabase={1}hdb,cn=config

add: olcAccess

olcAccess: {0}to attrs=userPassword,shadowLastChange by self write by anonymous auth by
dn="cn=admin,dc=ejemplo,dc=com" write by * none


dn: olcDatabase={1}hdb,cn=config

add: olcAccess

olcAccess: {1}to dn.base="" by * read


dn: olcDatabase={1}hdb,cn=config

add: olcAccess

olcAccess: {2}to dn.children="ou=Usuarios,dc=ejemplo,dc=com" by anonymous auth by self write by *
read
```

```
dn: olcDatabase={1}hdb,cn=config

add: olcAccess

olcAccess: {3}to * by self write by dn="cn=admin,dc=ejemplo,dc=com" write by * read
```

Y lo aplicamos, debería permitir el acceso anónimos a “sergio” y “maria” solo para autenticarse, pero no por ejemplo para leer los datos. Ten en cuenta que la salida de la búsqueda un poco más arriba la formateé para facilitar la lectura. En el archivo de modificación NO introduzcas saltos de línea ni nada similar en medio de atributos o sus valores. Ten en cuenta que he tenido que modificar todos los permisos para poder cambiar el orden de la cláusula que añado (la {2}) porque si la hubiera puesto al final se aplicaría primero la que ahora está al final y nunca se aplicaría la que he añadido.

```
# ldapmodify -Y EXTERNAL -H ldapi:/// -f ./permisos.ldif

SASL/EXTERNAL authentication started

SASL username: gidNumber=0+uidNumber=0,cn=peercred,cn=external,cn=auth

SASL SSF: 0

modifying entry "olcDatabase={1}hdb,cn=config"

modifying entry "olcDatabase={1}hdb,cn=config"

modifying entry "olcDatabase={1}hdb,cn=config"
```

```
modifying entry "olcDatabase={1}hdb,cn=config"
```

```
modifying entry "olcDatabase={1}hdb,cn=config"
```

Si repetimos la consulta de los permisos

```
# sudo ldapsearch -Q -LLL -Y EXTERNAL -H ldapi:/// -b olcDatabase={1}hdb,cn=config olcAccess
dn: olcDatabase={1}hdb,cn=config
olcAccess: {0}to attrs=userPassword,shadowLastChange by self write by anonymous
s auth by dn="cn=admin,dc=ejemplo,dc=com" write by * none
olcAccess: {1}to dn.base="" by * read
olcAccess: {2}to dn.children="ou=Usuarios,dc=ejemplo,dc=com" by anonymous auth auth by self write by
* read
olcAccess: {3}to * by self write by dn="cn=admin,dc=ejemplo,dc=com" write by * read
```

Nos queda comprobar que ya no hay acceso anónimo a “sergio” ni “maria” pero sí a otros nodos

```
# ldapsearch -x -LLL -b dc=ejemplo,dc=comdn: dc=ejemplo,dc=com
objectClass: top
objectClass: dcObject
```

```
objectClass: organization
```

```
o: ejemplo.com
```

```
dc: ejemplo
```

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

```
objectClass: simpleSecurityObject
```

```
objectClass: organizationalRole
```

```
cn: admin
```

```
description: LDAP administrator
```

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

```
objectClass: organizationalUnit
```

```
ou: Usuarios
```

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



```
objectClass: organizationalUnit

ou: Grupos

dn: cn=programadores,ou=Grupos,dc=ejemplo,dc=com

objectClass: posixGroup

cn: programadores

gidNumber: 5000

memberUid: cn=sergio,ou=Usuarios,dc=ejemplo,dc=com

memberUid: cn=maria,ou=Usuarios,dc=ejemplo,dc=com
```

Ten en cuenta que las dos últimas líneas son atributos de “programadores”, una entrada para la que no hemos modificado el acceso.

La última prueba es si se pueden leer estos datos con un usuario. Podría hacerlo con “admin”, pero es mejor probar con “sergio” que tiene menos privilegios. Hago la consulta desde usuarios para que sea un poco más corto el resultado.

```
# ldapsearch -x -D uid=sergio,ou=Usuarios,dc=ejemplo,dc=com -W -LLL -b ou=Usuarios,dc=ejemplo,dc=com

Enter LDAP Password:

dn: ou=Usuarios,dc=ejemplo,dc=com

objectClass: organizationalUnit
```

```
ou: Usuarios

dn: uid=sergio,ou=Usuarios,dc=ejemplo,dc=com

objectClass: inetOrgPerson
objectClass: posixAccount
objectClass: shadowAccount

uid: sergio

sn: Cuesta

givenName: Sergio

cn: Sergio Cuesta

displayName: Sergio Cuesta

uidNumber: 10000

gidNumber: 5000

gecos: Sergio Cuesta

loginShell: /bin/bash
```

```
homeDirectory: /home/scuesta  
  
mail: sergio.cuesta@ejemplo.com  
  
telephoneNumber: 55555555  
  
st: Madrid  
  
userPassword:: e1NTSEF9TWdKdGNSWW1KMW15MzBlbkwxVFFGQi8vYXVLZ3h3V0o=  
  
  
dn: uid=maria,ou=Usuarios,dc=ejemplo,dc=com  
  
objectClass: inetOrgPerson  
  
objectClass: posixAccount  
  
objectClass: shadowAccount  
  
uid: maria  
  
sn: Ramirez  
  
givenName: Maria  
  
cn: Maria Ramirez  
  
displayName: Maria Ramirez
```

```
uidNumber: 10001  
  
gidNumber: 5001  
  
gecos: Maria Ramirez  
  
loginShell: /bin/bash  
  
homeDirectory: /home/mramirez  
  
mail: maria.ramirez@ejemplo.com  
  
telephoneNumber: 66666666  
  
st: Madrid
```

Vemos que tiene acceso no solo a él, sino también a “maria”.

Modifica el control de acceso para que cada usuario solo pueda leer sus datos, pero no los de otro.

## Control de acceso en redes de ordenadores

En el caso que nos ocupa no necesitamos establecer un complejo sistema de autenticación ya que como he escrito no vamos a usar LDAP como un sistema de control de acceso de usuarios para toda una red sino que lo aplicaremos al acceso a aplicaciones. En el caso de que desarrollemos algo para una empresa que utilice LDAP para autenticar a sus usuarios en equipos y servicios de una red, la gestión de los usuarios la llevará un administrador y nosotros solo necesitaremos conectar el servidor LDAP con nuestro servidor web o de aplicaciones. Por todo lo anterior vamos a realizar una configuración simple que impida a cualquiera consultar nuestro árbol LDAP. Si por ejemplo hemos configurado nuestro servidor LDAP en la misma

máquina que el servidor web/de aplicaciones podríamos configurar los nombres del servidor LDAP mediante el fichero host y así no habría acceso a él desde otra máquina, lo que no nos hace falta porque habríamos ubicado el resto en la misma.

Es posible utilizar estos usuarios de LDAP como usuarios de las máquinas (Linux en este caso). [Aquí se explica cómo hacerlo](#), pero en mi opinión no tiene nada que ver con el despliegue de aplicaciones por lo que no profundizaré en ello. Este es un ámbito de administradores de sistemas y no de desarrolladores web.

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

Uno de los principales usos de LDAP en las empresas u organizaciones es el tener usuarios centralizados y comunes a todos sus controles de acceso: en equipos, en aplicaciones, en servicios, en recursos...

La configuración de LDAP para estos casos es bastante compleja pero en el caso que nos toca se refiere a establecer la autenticación en servidores web y de aplicaciones contra un servidor LDAP. La configuración de OpenLDAP ya la hemos hecho así que nos falta la integración con Apache y Tomcat.

### LDAP y Apache

Para realizar estos ejemplos voy a comenzar con una instalación limpia de Apache en el servidor donde he instalado OpenLDAP, como hemos visto durante todo el curso, esto podría combinarse con cualquier opción de configuración vista hasta el momento.

Lo primero que tenemos que hacer es habilitar el módulo [mod\\_authnz\\_ldap](#).

```
sudo a2enmod authnz_ldap

sudo service apache2 restart
```

Ahora tendremos que editar la configuración del sitio por defecto para habilitar la autenticación

```
<VirtualHost *:80>

    ServerAdmin webmaster@localhost
```

```
DocumentRoot /var/www

<Directory />

    Options FollowSymLinks

    AllowOverride None

</Directory>

<Directory /var/www/>

    Options Indexes FollowSymLinks MultiViews

    AllowOverride None

    Order allow,deny

    allow from all

    AuthName "Acceso a www.servidor.ejemplo.com"

    AuthType Basic

    AuthBasicProvider ldap

    AuthzLDAPAuthoritative on
```

```
AuthLDAPUrl "ldap://servidor.ejemplo.com/dc=ejemplo,dc=com?uid?sub?(objectClass=*) "  
  
AuthLDAPBindDN "cn=admin,dc=ejemplo,dc=com"  
  
AuthLDAPBindPassword sergio  
  
Require valid-user  
  
</Directory>  
  
...
```

Tras reiniciar el servidor ya podremos conectarnos con usuarios del servidor LDAP.

Busca en la documentación del módulo las directivas en negrita y comenta en clase para qué se utilizan. ¿Te parece buena opción utilizar el usuario administrador?

Ten en cuenta que no sería necesario un usuario con contraseña si permitiéramos las consultas anónimas. No es buena idea utilizar el usuario administrador, sería mucho mejor crear uno con permisos de lectura en todo el árbol pero sin permisos de escritura ya que la contraseña hay que escribirla en el archivo.

¿Qué pasa si queremos dar acceso a un grupo de usuarios pero no a cualquier usuario? El grupo que creamos no nos facilita esta tarea así que vamos a crear un grupo de otro tipo. Creo un archivo con el nuevo grupo

```
dn: cn=usrApache,ou=Grupos,dc=ejemplo,dc=com  
  
objectClass: groupOfUniqueNames  
  
objectClass: top
```



```
cn: usrApache

uniqueMember: uid=sergio,ou=Usuarios,dc=ejemplo,dc=com

uniqueMember: uid=maria,ou=Usuarios,dc=ejemplo,dc=com
```

Y lo proceso

```
ldapadd -x -D cn=admin,dc=ejemplo,dc=com -W -f grupo.ldif
```

Edito el fichero del sitio por defecto de apache

```
<VirtualHost *:80>

    ServerAdmin webmaster@localhost


    DocumentRoot /var/www

    <Directory />

        Options FollowSymLinks

        AllowOverride None

    </Directory>

    <Directory /var/www/>
```

```
Options Indexes FollowSymLinks MultiViews

AllowOverride None

Order allow,deny

allow from all

AuthName "Acceso a www.servidor.ejemplo.com"

AuthType Basic

AuthBasicProvider ldap

AuthzLDAPAuthoritative on

AuthLDAPUrl "ldap://servidor.ejemplo.com/dc=ejemplo,dc=com?uid?sub?(objectClass=*)"

AuthLDAPBindDN "cn=admin,dc=ejemplo,dc=com"

AuthLDAPBindPassword sergio

Require ldap-group cn=usrApache,ou=Grupos,dc=ejemplo,dc=com

</Directory>

...
```

Evidentemente esta segunda forma sería mucho más práctica cuando tenemos usuarios para varios fines.

Un tercer modo sería dar permiso a algunos usuarios determinados. Para ello solo habría que cambiar la línea en negrita por

```
Require ldap-user sergio maria
```

Por último es posible pedir que un determinado atributo tenga un valor especificado.

```
Require ldap-attribute gidNumber=5023
```

Incluso se pueden combinar requisitos

```
Require ldap-group cn=usrApache,ou=Grupos,dc=ejemplo,dc=com
```

```
Require ldap-user roberto, juan
```

```
Require ldap-attribute gidNumber=5023
```

```
Satisfy any
```

Donde la última línea indica que con cumplir con una de las anteriores es suficiente.

Habilita HTTPS en el servidor Apache y crea dos directorios. En OpenLDAP configura dos grupos con diferentes usuarios cada uno. Configura el acceso a cada uno de los directorios para que solo puedan acceder los usuarios de unos de los grupos. Comprueba que has configurado todo correctamente.

## LDAP y Tomcat

En el caso de Tomcat también voy a hacer una instalación limpia con la aplicación del formulario. Evidentemente es necesario modificar el descriptor de despliegue de la aplicación para que valide con el grupo o usuarios especificados en Tomcat.

Una vez tenemos el servidor Tomcat funcionando y hemos desplegado la aplicación (en mi caso en la carpeta “form”) podemos configurar el Realm. Recuerda que se puede configurar en cualquier contexto dependiendo de a qué nivel queremos que se utilice. En este caso voy a usarlo solo para la aplicación así que edito el contexto propio

```
gedit /var/lib/tomcat7/webapps/form/META-INF/context.xml
```

Y lo dejamos

```
<?xml version="1.0" encoding="UTF-8"?>

<Context antiJARLocking="true" path="/form">

    <Realm className="org.apache.catalina.realm.JNDIRealm"

        connectionName="cn=admin,dc=ejemplo,dc=com"

        connectionPassword="sergio"

        connectionURL="ldap://servidor.ejemplo.com:389"

        userPattern="uid={0},ou=Usuarios,dc=ejemplo,dc=com"

        roleBase="ou=Grupos,dc=ejemplo,dc=com"

        roleName="cn"

        roleSearch="(uniqueMember={0})"

    />
```

```
</Context>
```

Existen muchos otros [atributos y parámetros de configuración](#) y al igual que en el caso de Apache, sería mejor haber creado un usuario con permisos de lectura solo.

Consulta los atributos configurados en el JNDI Realm anterior y comenta en clase su uso.

Ahora editamos el descriptor de despliegue

```
gedit /var/lib/tomcat7/webapps/form/WEB-INF/web.xml
```

Y lo dejo como sigue, solo he tocado las líneas en negrita

```
<?xml version="1.0" encoding="UTF-8"?>

<web-app                                version="3.0"                                xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd">

    <servlet>

        <servlet-name>FormServlet</servlet-name>

        <servlet-class>form.FormServlet</servlet-class>

    </servlet>
```

```
<servlet-mapping>

    <servlet-name>FormServlet</servlet-name>

    <url-pattern>/FormServlet</url-pattern>

</servlet-mapping>

<!-- esto es lo que añadimos para la seguridad FORM -->

<!-- Primero una restricción de seguridad -->

<security-constraint>

    <web-resource-collection>

        <web-resource-name>El * significa que pedimos autenticación para toda la aplicación</web-
resource-name>

        <url-pattern>/*</url-pattern>

        <http-method>GET</http-method>

        <http-method>PUT</http-method>

        <http-method>HEAD</http-method>

        <http-method>POST</http-method>
```

```
<http-method>OPTIONS</http-method>

<http-method>TRACE</http-method>

<http-method>DELETE</http-method>

</web-resource-collection>

<auth-constraint>

    <role-name>usrApache</role-name>

</auth-constraint>

<user-data-constraint>

    <transport-guarantee>NONE</transport-guarantee>

</user-data-constraint>

</security-constraint>

<login-config>

    <auth-method>FORM</auth-method>

    <!-- CUIDADO, los archivos siguientes deben ir en un subdirectorio para que funcione-->
```

```
<form-login-config>

    <form-login-page>/protegido/login.jsp</form-login-page>

    <form-error-page>/protegido/login-failed.html</form-error-page>

</form-login-config>

</login-config>


<session-config>

    <session-timeout>

        30

    </session-timeout>

</session-config>


<security-role>

    <description/>

    <role-name>usrApache</role-name>
```



```
</security-role>  
  
</web-app>
```

Ahora ya podemos conectarnos con los mismos usuarios del grupo que creamos para Apache en el caso anterior.

Instala Tomcat con HTTPS y dos aplicaciones. Da acceso a cada aplicación a uno de los grupos que creaste para Apache. Prueba que todo es correcto.

Con este último ejercicio ya hemos visto que es muy útil tener un servidor LDAP para los usuarios ya que podríamos centralizar los usuarios tanto para la web como para las aplicaciones facilitando así la administración y el funcionamiento a los usuarios que solo tienen que conocer un único usuario y contraseña.

## Tema 6: Documentación y sistemas de control de versiones

## Herramientas externas para la generación de documentación

Las herramientas de generación de documentación permiten crear una documentación para programadores (API) o para usuarios finales a partir de comentarios introducidos en el código fuente. Un ejemplo que todos conocemos es Javadoc de Java con el cual si comentamos nuestro código siguiendo unas pautas determinadas podemos obtener una documentación similar a la de la [API de Java](#) para nuestras propias clases.

Existen muchísimas herramientas de este tipo por lo que es difícil elegir una. Por si fuera poco, podemos [ver en la comparativa](#) que ninguna está disponible para todos los principales sistemas operativos y todos los principales lenguajes de programación.

Si vamos a programar en Java principalmenete y el uso que pretendemos dar es principalmente documentar una API de nuestras clases Javadoc es la opción obvia. Para usos más específicos deberíamos buscar una que se ajuste a nuestras necesidades. Nosotros vamos a ver [Doxygen](#) por ser una de las más extendidas, estar disponible para todos los principales sistemas operativos y poder usarse con bastantes lenguajes de programación.

Documentar con Doxygen si se conoce Javadoc es bastante directo. Las ventajas que tiene Docxygen es que nos permite documentar más lenguajes de programación y que además puede producir varios formatos de salida. Esto y algunos ejemplos sencillos pueden verse en [su página de la Wikipedia](#).

En ninguno de los puntos referentes a la generación de documentación vamos a entrar en profundidad ya que se han visto en Javadoc y además es algo que va a depender de las necesidades de cada proyecto por lo que no podemos considerar una herramienta en concreto como predominante o destacada. Si alguien necesita más información sobre [Docxygen la documentación](#) es bastante completa.

### Instalación, configuración y uso

Instalar Docxygen es muy sencillo con los binarios. En el [manual explican](#) también como compilar los fuentes para instalarlos tanto en Linux como en Windows.

En mi caso voy a usar el binario para Mac [pero aquí se puede descargar](#) cualquiera. ¿Por qué en este caso no utilizo Linux? Doxygen debe instalarse en la máquina en la que desarrollemos habitualmente.

Tras instalar el programa nos sale una pantalla donde habrá que configurar la carpeta donde estarán los fuentes del proyecto. El uso de la opción Wizard es muy sencillo.

Cómo debo documentar el [código se explica](#) en la documentación.

Investiga y discute en clase las diferentes opciones de cada pantalla: Project, Mode, Output y Diagrams

Una vez tengas claras las opciones, vamos a documentar un ejemplo y probaremos a exportarlo en chm.

Este tipo de herramientas son mucho más útiles si podemos integrarlas en el entorno de desarrollo. Doxygen tiene plugins en [Netbeans](#) y en [Eclipse](#).

## Creación y utilización de plantillas

La documentación generada puede mostrarse en diferentes formatos. Por ejemplo, en la carpeta Javadoc de proyectos de Java hay un CSS con la configuración de apariencia por defecto de la API. Editando esta CSS podemos conseguir el aspecto que queramos.

En Doxygen también existe esta posibilidad: En la opción Wizard solo podemos configurar los colores (output, HTML, change color), pero en el modo Expert, HTML podemos seleccionar una CSS propia.

Sin embargo el término plantillas se utiliza para muchos otros conceptos como podemos [ver aquí](#). En la generación automática de código se utiliza para crear plantillas de comentarios. Por ejemplo una plantilla para los comentarios de un paquete, otra para los de clase, otra para los de método, etc.

Estas plantillas nos permiten incluir campos o comentarios por defecto además de elementos como por ejemplo el autor, la fecha, etc.

Las plantillas se crean y se utilizan en el entorno de desarrollo determinado que estemos utilizando. Por ejemplo en Eclipse, en las opciones de cada lenguaje de programación suele haber una opción denominada *Code Templates* que es donde se modifica.

## Instalación, configuración y uso de sistemas de control de versiones.

<http://www.avajava.com/tutorials/categories/version-control>

### Subversion

- Operaciones avanzadas.
- Seguridad de los sistemas de control de versiones.
- Historia de un repositorio.

## Apéndices

En este apartado se incluyen herramientas y programas que no aparecen en el temario del módulo o son necesarias antes de aparecer pero que se consideran importantes para el correcto desarrollo del mismo. Por estos motivos se cubren de manera somera y se suele presentar la manera más sencilla de usarlos.

## Webmin: Una interfaz gráfica de administración

Webmin es un administrador de interfaz web para servidores en Linux.

Para instalarlo, debemos añadir fuentes de software nuevo a nuestra instalación. Lo primero es abrir el archivo de fuentes

```
sudo gedit /etc/apt/sources.list
```

y añadir las siguientes líneas

```
deb http://download.webmin.com/download/repository sarge contrib  
deb http://webmin.mirror.somersettechsolutions.co.uk/repository sarge contrib
```

Tras guardar y salir debemos importar la clave GPG

```
wget http://www.webmin.com/jcameron-key.asc  
sudo apt-key add jcameron-key.asc
```

actualizamos la lista de fuentes

```
sudo apt-get update
```

e instalamos Webmin

```
sudo apt-get install webmin
```

Podemos acceder a la consola de Webmin desde <https://serverip:10000/> cambiando *serverip* por el nombre del que nos informa cuando termina la instalación. También se puede acceder usando la IP en el mismo puerto o si estamos en la misma máquina con localhost y el puerto. En todos los casos ten en cuenta que la conexión es HTTPS. El usuario es *root* y la contraseña la que le hayamos asignado en Linux.



## Un pequeño servidor DNS

En nuestro caso vamos a instalar un pequeño DNS en nuestro servidor para poder probar la opción.

Ten en cuenta que antes de instalar algo es importante tener la máquina actualizada.

```
sudo apt-get update
```

En mi caso estoy instalando Bind en la misma máquina virtual en la que he instalado Apache. Esta máquina no es visible desde la red por lo que solo se podrá probar desde dentro de esta máquina. La dirección IP de la máquina es 10.0.2.15

Podríamos hacer que fuera visible desde la máquina anfitrión si hemos activado esa interfaz en Virtual Box y creamos todos los archivos con la dirección la interfaz de red correspondiente, pero además deberíamos cambiar las direcciones de los DNS de la máquina anfitrión para que usen el nuestro. Como no es recomendable, lo probaremos solo desde la máquina virtual.

```
sudo apt-get install bind9
```

abrimos el fichero de configuración

```
sudo gedit /etc/bind/named.conf.local
```

y escribimos lo siguiente

```
# Esta es la definición de la zona. Remplaza el dominio de ejemplo por otro

zone "ejemplo.es" {

    type master;
```

```
file "/etc/bind/zones/ejemplo.es.db";

};

# Esta es la definicion de la zona inversa para DNS. Modifica las direcciones IP para que se ajusten
a las tuyas.

zone "2.0.10.in-addr.arpa" {

type master;

file "/etc/bind/zones/rev.2.0.10.in-addr.arpa";

};
```

Podemos comprobar que la sintaxis sea correcta con la siguiente orden. Si es correcta no aparecerá ningún mensaje pero si no lo es nos mostrará el problema.

```
named-checkconf
```

Lo que debemos hacer ahora es abrir el archivo de opciones.

```
sudo gedit /etc/bind/named.conf.options
```

en él debemos quitar los comentarios de la parte de *forwarders* y dejarlo como sigue. Ten en cuenta que las direcciones de los servidores DNS de tu configuración pueden cambiar. Se debe escribir esta orden justo en el sitio que está indicado. Escribirlo por ejemplo al final del archivo haría que nuestro servidor no funcionara.

```
forwarders {  
  
    80.58.61.254;  
  
    80.58.61.250;  
  
};
```

Creamos el directorio para configurar las zonas

```
sudo mkdir /etc/bind/zones
```

y creamos el archivo de configuración de nuestra zona.

```
sudo gedit /etc/bind/zones/ejemplo.es.db
```

y escribimos lo siguiente

```
; cambia ejemplo.es por el nombre de tu dominio. No te olvides el punto despues del nombre de dominio  
  
$TTL      3600  
  
ejemplo.es. IN SOA ns.ejemplo.es. admin.ejemplo.es. (  
  
; NO modifiques los siguiente parametros  
  
2006081401  
  
28800
```

```
3600

604800

38400

)

; Modifica lo siguiente para que se ajuste a tu nombre de dominio
ejemplo.es. IN NS ns.ejemplo.es.

; Modifica las direcciones IP con las adecuadas en tu caso

@      IN      A      10.0.2.15

ns     IN      A      10.0.2.15

www    IN      A      10.0.2.15
```

Podemos comprobar la correcta sintaxis con la orden

```
named-checkzone ejemplo.es /etc/bind/zones/ejemplo.es.db
```

Donde el primer parámetro es el nombre de la zona y el segundo la ruta al archivo correspondiente.

Debe dar una salida como

```
zone ejemplo.es/IN: loaded serial 2006081401  
  
OK
```

El siguiente paso es crear el archivo de zona inversa.

```
sudo gedit /etc/bind/zones/rev.2.0.10.in-addr.arpa
```

y escribir lo siguiente

```
; modifica los parametros correctamente  
  
; el numero antes de IN PTR ejemplo.es es la dirección de la maquina donde este el servidor DNS  
  
$TTL      604800  
  
@ IN SOA ns.ejemplo.es. admin.ejemplo.es. (  
    2006081401;  
    28800;  
    604800;  
    604800;
```

```
86400
)

; 15 es el ultimo numero de la ip de la maquina en la red.

@                IN      NS      ns.ejemplo.es.

15               IN      PTR     ejemplo.es.
```

Y lo comprobamos

```
named-checkzone 2.0.10.in-addr.arpa /etc/bind/zones/rev.2.0.10.in-addr.arpa
```

En este punto nos toca reiniciar el servicio DNS

```
sudo service bind9 restart
```

En caso de que nos dé **problemas** lo mejor es abrir la interfaz de Webmin y acceder a *Servers > BIND DNS Server*. La opción *Check BIND Config* es de mucha utilidad para comprobar que no nos hemos equivocado en la configuración. El problema más habitual es que haya algún fallo en RND. La opción *Setup RND* nos puede ayudar.

**Otro problema** muy habitual es que si has copiado y pegado de este archivo a los de configuración, las comillas dobles son diferentes por lo que no funcionará. Debes sustituir todas las comillas dobles y volver a escribirlas.

Debemos probar que todo funcione correctamente. Lo primero es informar a nuestro ordenador de que debe buscar en el servidor la resolución de nombres.

```
gedit /etc/resolv.conf
```

y sustituir lo que ponga por lo siguiente

```
nameserver 10.0.2.15
```

```
domain ejemplo.es
```

Ya podemos probar el resultado

```
dig ejemplo.es
```

Lo que nos devolverá algo similar a lo siguiente:

```
; <<>> DiG 9.8.1-P1 <<>> ejemplo.es

;; global options: +cmd

;; Got answer:

;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 22708

;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 1


;; QUESTION SECTION:

;ejemplo.es.                IN      A
```

```
;; ANSWER SECTION:

ejemplo.es.      3600 IN      A      10.0.2.15

;; AUTHORITY SECTION:

ejemplo.es.      3600 IN      NS      ns.ejemplo.es.

;; ADDITIONAL SECTION:

ns.ejemplo.es.   3600 IN      A      10.0.2.15

;; Query time: 3 msec

;; SERVER: 10.0.2.15#53(10.0.2.15)

;; WHEN: Wed Aug 29 14:53:56 2012

;; MSG SIZE rcvd: 77
```

También podemos probar con ping si se encuentra



```
ping -c 4 www
```

y debemos obtener algo como

```
PING www.ejemplo.es (10.0.2.15) 56(84) bytes of data.  
  
64 bytes from ejemplo.es (10.0.2.15): icmp_req=1 ttl=64 time=0.017 ms  
  
64 bytes from ejemplo.es (10.0.2.15): icmp_req=2 ttl=64 time=0.033 ms  
  
64 bytes from ejemplo.es (10.0.2.15): icmp_req=3 ttl=64 time=0.031 ms  
  
64 bytes from ejemplo.es (10.0.2.15): icmp_req=4 ttl=64 time=0.029 ms  
  
--- www.ejemplo.es ping statistics ---  
  
4 packets transmitted, 4 received, 0% packet loss, time 2999ms  
  
rtt min/avg/max/mdev = 0.017/0.027/0.033/0.008 ms
```

Podemos probar a hacer una consulta a un sitio externo para comprobar que seguimos teniendo acceso a Internet

```
ping -c 4 www.google.es
```

```
PING www-cctld.l.google.com (173.194.34.23) 56(84) bytes of data.
```

```
64 bytes from par03s02-in-f23.1e100.net (173.194.34.23): icmp_req=1 ttl=51 time=84.4 ms
64 bytes from par03s02-in-f23.1e100.net (173.194.34.23): icmp_req=2 ttl=51 time=83.5 ms
64 bytes from par03s02-in-f23.1e100.net (173.194.34.23): icmp_req=3 ttl=51 time=74.6 ms
64 bytes from par03s02-in-f23.1e100.net (173.194.34.23): icmp_req=4 ttl=51 time=75.3 ms

--- www-cctld.l.google.com ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3759ms
rtt min/avg/max/mdev = 74.668/79.480/84.406/4.516 ms
```

E incluso si tenemos instalado Apache podemos acceder a la página por defecto escribiendo en el navegador web [www.ejemplo.es](http://www.ejemplo.es).

Si usamos hosts virtuales basados en nombres, necesitaremos crear un registro para el nuevo dominio pero realmente será un alias al anterior. Lo primero es añadir el nuevo dominio a la configuración de BIND.

```
sudo gedit /etc/bind/named.conf.local
```

y añadimos lo siguiente. Por coherencia debería ir antes de la definición de la zona inversa.

```
...

# Esta es la definicion de la zona. Reemplaza el dominio de ejemplo por otro

zone "ejemplo2.es" {
```

```
type master;  
  
file "/etc/bind/zones/ejemplo2.es.db";  
  
};  
  
...
```

Ahora creamos el archivo de la zona

```
sudo gedit /etc/bind/zones/ejemplo2.es.db
```

y añadimos los datos con el registro CNAME adecuado

```
; cambia ejemplo.es por el nombre de tu dominio. No te olvides el punto despues del nombre de dominio  
  
$TTL      3600  
  
ejemplo2.es. IN SOA ns.ejemplo2.es. admin.ejemplo2.es. (  
  
; NO modifiques los siguiente parametros  
  
2006081401  
  
28800  
  
3600  
  
604800
```

```
38400
```

```
)
```

```
; Modifica lo siguiente para que se ajuste a tu nombre de dominio
```

```
ejemplo2.es. IN NS ns.ejemplo.es.
```

```
; Registro CNAME es un alias a otro ya existente. Es útil para los hosts virtuales basados en nombres.
```

```
www.ejemplo2.es. IN CNAME ejemplo.es.
```

tras hacerlo reiniciamos BIND

```
service bind9 restart
```

y podemos acceder a nuestro servidor con el nuevo nombre de dominio por ejemplo desde el navegador Web.

## Instalar MySQL

Para instalar la base de datos MySQL usaremos el comando:

```
sudo apt-get install mysql-server-5.5
```

que puede depender de la versión que haya salido en su momento y que se puede consultar en <http://dev.mysql.com/downloads/mysql/>

Durante la instalación nos pedirán que introduzcamos una contraseña de administrador que aunque no es obligatoria, si muy recomendable. Yo pongo "sergio" porque es una instalación de pruebas.

MySQL se arranca al iniciar la máquina, pero podemos usar los comandos adecuados en cada caso

```
/etc/init.d/mysql start  
  
/etc/init.d/mysql stop  
  
/etc/init.d/mysql restart  
  
/etc/init.d/mysql status
```

aunque el propio MySQL nos recomienda usar los comandos asociados al servicio.

```
service mysql start  
  
service mysql stop  
  
service mysql restart
```

```
service mysql status
```

el archivo de configuración se puede ver en

```
sudo gedit /etc/mysql/my.cnf
```

para conectarnos a MySQL con el usuario root

```
mysql -u root -p
```

lo que hará que nos pida la contraseña que hemos establecido.

Para salir usaremos `quit` o `exit`.