Implantación de arquitecturas web.

Caso práctico

Juan, con su perfil de Técnico Superior en Desarrollo de Aplicaciones Web, va a trabajar para la empresa **BK programación.** Por ese motivo, ha decidido documentar, en una wiki interna para los miembros de dicha empresa, los conocimientos que va a ir adquiriendo durante su vida laboral respecto a los trabajos que le toque realizar. De este modo, a los compañeros que le sucedan o que tengan que realizar labores similares les será útil esta información.

Juan ha pensado en incluir un tema titulado **Implantación de arquitecturas web** que estructurará en varios apartados.

En un principio, considera importante realizar un resumen claro y conciso del concepto de arquitecturas web en donde pretende abarcar como mínimo los siguientes puntos: evolución de los servicios web, tecnologías asociadas a las aplicaciones web, tipos de aplicaciones web, arquitecturas web, etc.

También pretende incluir en otro de los puntos el servidor web Apache, abarcando los siguientes apartados: instalación, configuración básica e inicio de Apache, y está pensando documentar los mismos puntos para el servidor Tomcat.

Finalmente, piensa crear un último punto en donde explicar el proceso de despliegue de una aplicación web; explicando el despliegue de contenido estático y el de una aplicación web Java, la estructura de carpetas y recursos de una aplicación web y el descriptor del despliegue.

Para realizar este trabajo Juan ha contado con la colaboración de sus amigos **Carlos** y **Ana**, estudiantes ambos de Ciclos de Informática.

En esta unidad se introducen los conceptos fundamentales en los que se basa el módulo. Se explican los aspectos generales de las arquitecturas web, entre otros: tecnologías, tipos, modelos, etc.

Se analizan las principales características del servidor web Apache, su instalación y configuración básica; posteriormente se realiza el estudio comparativo entre servidor web y servidor de aplicaciones web, explicando el trabajo básico de instalación y puesta en funcionamiento del servidor de aplicaciones Tomcat.

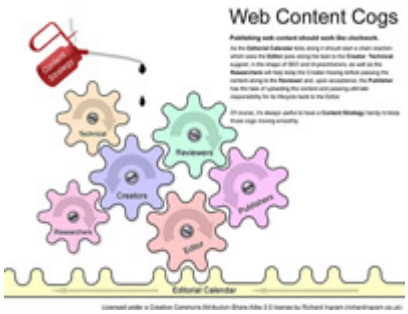
Por último, se aborda el estudio de la estructura y despliegue de aplicaciones web, en donde se introduce el concepto de archivos WAR y descriptor del despliegue.

Materiales formativos de FP Online propiedad del Ministerio de Educación y Formación Profesional.

Aviso Legal

1.- Aspectos generales de arquitecturas web.

Caso práctico



richardjingram (CC BY-SA)

Juan ha decidido empezar la documentación de la wiki explicando los aspectos básicos de la arquitectura web, ya que considera que es un medio que es necesario conocer con precisión si se pretende realizar el **despliegue de una aplicación web**. Para ello ha pensado en definir conceptos básicos de las interfaces web.

La arquitectura World Wide Web (WWW) de Internet provee un modelo de programación sumamente poderoso y flexible. Las aplicaciones y los contenidos son presentados en formatos de datos estándar y son localizados por aplicaciones conocidas como "web browsers", que envían requerimientos de objetos a un servidor y éste responde con el dato codificado según un formato estándar.

Los estándares WWW especifican muchos de los mecanismos necesarios para construir un ambiente de aplicación de propósito general, por ejemplo:

- **Modelo estándar de nombres:** todos los servidores, así como el contenido de la WWW se denominan según un Localizador Uniforme de Recursos (Uniform Resource Locator: URL).
- **Contenido:** a todos los contenidos en la WWW se les especifica un determinado tipo permitiendo de esta forma que los browsers (navegadores) los interpreten correctamente.
- **Formatos de contenidos estándar:** todos los navegadores soportan un conjunto de formatos estándar, por ejemplo HTML, ECMA, JavaScript, etc.
- **Protocolos estándar:** éstos permiten que cualquier navegador pueda comunicarse con cualquier servidor web. El más comúnmente usado en WWW es HTML (Protocolo de Transporte de HiperTexto), que opera sobre el conjunto de protocolos TCP/IP.

Esta infraestructura permite a los usuarios acceder a una gran cantidad de aplicaciones y servicios de terceros. También permite a los desarrolladores crear aplicaciones y servicios para una gran comunidad de clientes.

Salah-Eddine (CC BY-SA)

Los aspectos generales a destacar en una arquitectura web son los siguientes:

- Escalabilidad.
- Separación de responsabilidades.
- Portabilidad.

- Utilización de componentes en los servicios de infraestructura.
- Gestión de las sesiones del usuario.
- Aplicación de patrones de diseño.

El esquema de funcionamiento de los servicios web requiere de tres elementos fundamentales:

1.- **Proveedor del servicio web**, que es quien lo diseña, desarrolla e implementa y lo pone disponible para su uso, ya sea dentro de la misma organización o en público.
2.- **Consumidor del servicio**, que es quien accede al componente para utilizar los servicios que éste presta.
3.- **Agente del servicio**, que sirve como enlace entre proveedor y consumidor para efectos de publicación, búsqueda y localización del servicio.

De forma genérica podríamos decir que la arquitectura web es un modelo compuesto de tres capas:

1.- **Capa de Base de Datos,** donde estaría toda la documentación de la información que se pretende administrar mediante el servicio web y emplearía una plataforma del tipo MySQL, PostgreSQL, etc.
2.- En una segunda capa estarían los **servidores de aplicaciones web,** ejecutando aplicaciones de tipo Apache, Tomcat, Resin, etc.
3.- En una tercera capa estarían los **clientes del servicio web** al que accederían mediante un navegador web como Firefox, Internet Explorer, Opera, etc.

1.1.- Evolución de los servicios web.

Caso práctico

Juan ha enviado un correo electrónico a **Ana** solicitándole la ayuda para poder documentar la evolución de los servicios web en la wiki de su empresa.

Ana ha accedido encantada a su petición ya que está muy interesada en colaborar con la empresa en la que **Juan** trabaja, llegando incluso a pedirle a éste que le cree un usuario para poder acceder a la wiki interna de **BK programación** y ella misma documentar parte de los puntos que **Juan** tiene pensado redactar.

La evolución del uso de Servicios web en las organizaciones está fuertemente ligada al desarrollo de Internet como red prestadora de servicios. Entre los factores que han impulsado el uso de los servicios web se encuentran:

On Alien Cinema (CC BY-NC-SA)

- El **contenido se está volviendo más dinámico**: Los sitios web actuales proporcionan contenidos "instantáneos". Un Servicio web debe ser capaz de combinar contenido proveniente de fuentes muy diferentes.
- El **ancho de banda es menos costoso**: Actualmente un Servicio web puede entregar tipos variables de contenidos como vídeo o audio. A medida que crezca el ancho de banda, los servicios web deben adaptarse a nuevos tipos de contenidos.
- El **almacenamiento es más barato**: Un Servicio web debe ser capaz de manejar cantidades masivas de datos, y debe poder hacerlo de forma inteligente.
- El **éxito de la computación extendida** se está volviendo más importante: Con cientos de millones de dispositivos como teléfonos móviles, agendas electrónicas, etc. existentes actualmente, estamos llegando a un momento en el cual las computadoras están dejando de ser el dispositivo más común en Internet. A medida que las plataformas se hacen más diversas, tecnologías como XML se volverán más importantes. Un servicio web no puede exigir que los usuarios ejecuten, por ejemplo, un navegador web tradicional en alguna versión de Microsoft Windows; por el contrario, los servicios web deben servir a todo tipo de dispositivos, plataformas y navegadores, entregando contendido sobre una amplia variedad de tipos de conexión.

Estos factores, unidos a los beneficios proporcionados por los servicios web en la organización y los buenos productos disponibles para su desarrollo, han hecho que su utilización se extienda sin mayores obstáculos.

En términos generales, cuando se empiezan a utilizar servicios web en una organización, estos se desarrollan e implementan como servicios simples, que poco a poco se van integrando hasta llegar a servicios web mucho más complejos.

En los orígenes del mundo web nos situábamos ante un entorno estático, con páginas en formato HTML que raramente sufrían modificaciones o actualizaciones y en las que apenas había interacción con el usuario.

La **Web 2.0** es la transición que se ha dado desde las aplicaciones tradicionales hacia aplicaciones que funcionan a través de la web y que están fuertemente enfocadas al usuario final. En este nuevo entorno existen una serie de nuevas tecnologías que, en general, tienen como objetivo:

- Transformar software de escritorio hacia la web.
- Separar hojas de estilo.
- Potenciar el trabajo colaborativo y la utilización de redes sociales.
- Dar control total a los usuarios en el manejo de su información.

1.2.- Tecnologías asociadas a las aplicaciones web.

Caso práctico

Carlos, debido a su afición por el diseño web, ha ofrecido a su amigo **Juan** realizar un estudio sobre las tecnologías que se emplean actualmente en esta materia. A **Juan** le ha parecido una idea magnífica puesto que le va a servir para el desarrollo de su wiki.

Petits et Maman (CC BY-NC-ND)

Las aplicaciones web emplean páginas dinámicas, éstas se ejecutan en un servidor web y se muestran en el navegador de un equipo cliente que es el que ha realizado previamente la solicitud. Cuando una página web llega al navegador, es posible que también incluya algún programa o fragmento de código que se deba ejecutar. Ese código, normalmente en lenguaje JavaScript, **lo ejecutará el propio navegador.** Es por ello que en este apartado nos centraremos en las tecnologías asociadas a las aplicaciones web que se ejecutarán tanto del lado del servidor como del cliente, especificando lo que corresponda en cada uno de los casos.

- **ASP (Active Server Pages):** Las "Páginas Activas" se ejecutan del lado del servidor, de este modo se forman los resultados que luego se mostrarán en el navegador de cada equipo cliente que ha realizado la solicitud. Un buen ejemplo de ello son los buscadores, donde un usuario realiza una petición de información y el servidor nos entrega un resultado a medida de nuestra petición. Existen versiones de ASP para Unix y Linux, a pesar de que fue una tecnología desarrollada por Microsoft para la creación dinámica de páginas web ofrecida junto a su servidor IIS.
- **CGI (Common Gateway Interface):** La "Interface Común de Entrada" es uno de los estándares más antiguos en Internet para trasladar información desde una página a un servidor web**. Este estándar** es utilizado para bases de datos, motores de búsqueda, formularios, generadores de email automático, foros, comercio electrónico, rotadores y mapas de imágenes, juegos en línea, etc. Las rutinas de CGI son habitualmente escritas en lenguajes interpretados como Perl o por lenguajes compilados como C.
- **CSS (Cascading Style Sheets):** Las "Hojas de Estilo en Cascada" se usan para formatear las páginas web; se trata de separar el contenido de un documento, de su

Javier Benek (CC BY-NC-SA)

presentación. Cualquier cambio en el estilo marcado para un elemento en la CSS afectará a todas las páginas vinculadas a esa CSS.

- **Java:** Este es un lenguaje que trabaja en el cliente, es decir: se ejecuta en el navegador del equipo cliente y no en el servidor. Es un lenguaje eficiente y muy poderoso, que se caracteriza por:
  - Una misma aplicación puede funcionar en diversos tipos de ordenadores y sistemas operativos: Windows, Linux, Solaris, MacOS, etc., así como en otros dispositivos inteligentes.
  - Los programas Java pueden ser aplicaciones independientes (que corren en una ventana propia) o "applets", que son pequeños programas interactivos que se encuentran incrustados en una página web y pueden funcionar con cualquier tipo de navegador: Explorer, Netscape, Ópera, etc.
  - Se trata de un lenguaje "orientado a objetos". Esto significa que los programas se construyen a partir de módulos independientes, y que estos módulos se pueden transformar o ampliar fácilmente. Un equipo de programadores puede partir de una aplicación existente para extenderla con nuevas funcionalidades.
  - Desarrollado por la empresa Sun Microsystems, pero posteriormente liberado bajo licencia GNU GPL, con lo cual es un software libre.
- **JavaScript:** Lenguaje que se interpreta y se ejecuta en el cliente. Útil para realizar tareas como mover imágenes por la pantalla, crear menús de navegación interactivos, utilizar algunos juegos, etc. En las páginas web suele preferirse JavaScript porque es aceptado por muchos más navegadores que VBScript (creado por Microsoft)
- **PHP (Hypertext Preprocessor):** Este lenguaje es, como ASP, ejecutado en el lado del servidor. PHP es similar a ASP y puede ser usado en circunstancias similares. Es muy eficiente, permitiendo el acceso a bases de datos empleando servidores como MySQL y, por lo tanto, suele utilizarse para crear páginas dinámicas complejas.
- **VBScript (Visual Basic Scripting):** La respuesta de Microsoft a JavaScript. VBScript es una buena herramienta para cualquier sitio destinado a ser mostrado exclusivamente en el navegador Microsoft Internet Explorer. El código en VBScript puede, además, estar diseñado para su ejecución en el lado del cliente o en el del servidor, la diferencia es que un código que se ejecuta en el lado del servidor no es visible en el lado del cliente. Éste recibe los resultados, pero no el código.

the beastieux (CC BY-SA)

Autoevaluación

¿Podemos ver una página web sin que intervenga un servidor web?

- ○ Sí.
- ○ No.

Efectivamente. Podemos ver páginas web con extensión `.htm`, `.html` o `.xhtml` que tengamos almacenadas en nuestro equipo simplemente abriéndolas con el navegador. En este caso la única utilidad del servidor web es enviar la página que solicitemos a nuestro equipo.

Piénsalo bien... ¿Cuál es realmente la utilidad del servidor web, según lo que acabamos de ver?

Solución

1. Opción correcta
2. Incorrecto

1.3.- Tipos de aplicaciones web.

Caso práctico

Tras una gran labor de documentación acerca de las tecnologías de aplicaciones web, **Carlos** es consciente de que existe una gran diversidad de aplicaciones disponibles en la web. Por ello, ha decidido documentar un apartado denominado "Tipos de aplicaciones web" que puede ser de utilidad para la wiki que está creando su amigo **Juan**.

Link576 (CC BY-NC-SA)

Cualquier proyecto que se quiera desarrollar en Internet, bien sea comercio electrónico, reservas de billetes de vuelo on-line, información meteorológica, registro de usuarios, simuladores de hipotecas, etc, conlleva el desarrollo de una aplicación web. En definitiva, una aplicación web es una plataforma orientada a automatizar los procesos de servicios que se quieran ofrecer a usuarios.

Elaboración propia

Establecer una clasificación de los tipos de aplicaciones web es una tarea compleja debido a la dificultad existente para poder establecer algún parámetro en función del cual establecer dicha clasificación, junto con la innumerable cantidad de aplicaciones existentes en el actual entorno **Web 2.0**.

En función de cómo se presenta la aplicación web junto con el contenido que pretende mostrar, se ha establecido la siguiente clasificación:

- **Página web Estática.** Están implementadas en HTML y pueden mostrar en alguna parte de la página objetos en movimiento tales como banners, GIF animados, vídeos, etc.
- **Página web Animada**. Se realizan con la tecnología FLASH; ésta permite que una página web presente el contenido con ciertos efectos animados continuados. El uso de esta tecnología permite diseños más vanguardistas, modernos y creativos. Esta tecnología está descontinuada y desaconsejada ya que los navegadores la bloquean por su gran cantidad de vulnerabilidades.

- **Página web Dinámica.** Existen muchos lenguajes de programación que son la base para la mayoría de páginas web dinámicas. Los que destacamos aquí son los lenguajes PHP y ASP. Estos lenguajes permiten una perfecta estructuración del contenido. Por una parte crearíamos la estructura de las páginas web y por otra, almacenaríamos el contenido en determinados archivos. A partir de ahí, crearíamos el código de llamada, que insertaría el contenido en la propia página web estructurada. Este es el principio básico que siguen los lenguajes de programación. A partir de aquí se desarrollan aplicaciones para poder gestionar el contenido a través de un panel de control.
- **Portal**. Es un sitio web que en su página principal permite el acceso a múltiples secciones que, por lo general, son foros, chats, cuentas de correo, buscador, acceso registrado para obtener ciertas ventajas, las últimas noticias de actualidad, etc.
- **Tienda virtual o comercio electrónico**. Sitio web que publica los productos de una tienda en Internet. Permite la compra on-line a través de tarjeta de crédito, domiciliación bancaria o transferencia bancaria en general. Ofrece al administrador un panel de gestión para poder subir los productos, actualizarlos, eliminarlos, etc.
- **Página web con "Gestor de Contenidos".** Se trata de un sitio web cuyo contenido se actualiza a través de un panel de gestión por parte del administrador del sitio. Este panel de gestión suele ser muy intuitivo y fácil de usar. En aquellas páginas web que requieran una actualización constante, se suele incorporar este panel de gestión para que la web pueda controlarse día a día por parte del cliente.

Reflexiona

Cuando adquirimos un equipo informático nuevo, existen una serie de aplicaciones imprescindibles que es necesario instalar junto con los drivers de nuestro equipo para poder empezar a utilizarlo. Entre estas aplicaciones encontramos aplicaciones ofimáticas, antivirus, aplicaciones de mensajería, compresores, visualizadores, reproductores multimedia, etc.

¿En algún momento te has parado a pensar qué cantidad de aplicaciones web hay disponibles en Internet para substituir a las que tienes pensado instalar en el equipo?

1.4.- Arquitecturas web. Modelos.

Caso práctico

En la wiki que **Juan**, junto con sus amigos está desarrollando, ha decidido integrar un punto en el cual explicar los diversos tipos de modelos de arquitecturas web que se han utilizado a lo largo del tiempo. La finalidad es tener en cuenta las características de cada uno de ellos y, en función de las mismas, distinguir sus ventajas e inconvenientes.

mrjoro (CC BY-NC)

Se puede establecer que la arquitectura de un sitio web comprende los sistemas de organización y estructuración de los contenidos junto con los sistemas de recuperación de información y navegación que provea el sitio web, con el objetivo de servir de ayuda a los usuarios a encontrar y manejar la información.

On Alien Cinema (CC BY-NC-SA)

Centraremos el estudio de los modelos de arquitectura web relacionados, en función de cómo implementan cada una de las capas establecidas en una aplicación web:

1.- **Capa de presentación** es la encargada de la navegabilidad, validación de los datos de entrada, formateo de los datos de salida, presentación de la web, etc.; se trata de la capa que se presenta al usuario.
2.- **Capa de negocio** es la que recibe las peticiones del usuario y desde donde se le envían las respuestas; en esta capa se verifica que las reglas establecidas se cumplen.
3.- **Capa de acceso a datos** es la formada por determinados gestores de datos que se encargan de almacenar, estructurar y recuperar los datos solicitados por la capa de negocio.

La evolución experimentada por los medios informáticos en los últimos años ha convivido con otra evolución paralela, la evolución de la arquitectura de las aplicaciones web, que permite aprovechar las nuevas características que éstas ofrecen. De esta forma, el modelo arquitectónico de las aplicaciones de Internet ha sufrido dos grandes saltos, con algún paso intermedio, desde la aparición de los primeros portales web. Los distintos modelos de aplicación sobre los que se ha ido desarrollando, según diversos autores, se podrían clasificar del siguiente modo:

- **Modelo 1**
  En este caso las aplicaciones se diseñan en un modelo web CGI, basadas en la ejecución de procesos externos al servidor web, cuya salida por pantalla era el HTML que el navegador recibía en respuesta a su petición. Presentación, negocio y acceso a datos se confundían en un mismo script Perl.

- **Modelo 1.5**
  Aplicado a la tecnología java, se da con la aparición de las JSP y los servlets. En este modelo, las responsabilidades de presentación recaen en las páginas JSP, mientras que los beans incrustados en las mismas son los responsables del modelo de negocio y acceso a datos.

- **Modelo 2**
  Como evolución del modelo anterior, con la incorporación del patrón MVC en este tipo de aplicaciones, se aprecia la incorporación de un elemento controlador de la navegación de la aplicación. El modelo de negocio queda encapsulado en los javabeans que se incrustan en las páginas JSP.

- **Modelo 2X**
  Aparecen con el objetivo de dar respuesta a la necesidad, cada vez más habitual, de desarrollar aplicaciones multicanal, es decir, aplicaciones web que pueden ser atacadas desde distintos tipos de clientes remotos. Así, una aplicación web multicanal podrá ejecutarse desde una PDA, desde un terminal de telefonía móvil, o desde cualquier navegador HTML estándar. El medio para lograr publicar la misma aplicación para distintos dispositivos es emplear plantillas XSL para transformar los datos XML.

Para saber más

Esta web está pensada como un curso en español de Java básico. Pretende tener una interacción con los lectores, de forma que se puedan resolver las dudas que surjan.

Java básico en lenguaje español.

1.5.- Plataformas web libres y propietarias.

Caso práctico

El diseño de aplicaciones web requiere de un entorno funcional, que permita el desarrollo de cada uno de los componentes que forman la aplicación web, junto con un entorno complejo de herramientas que ofrecen al cliente los servicios de las aplicaciones web; para todo ello, el mercado pone a disposición de los programadores un abanico de herramientas software, que **Juan** pretende analizar en la wiki de la empresa **BK programación**, estableciendo el criterio de clasificación que considera primordial: software libre o software propietario.

logan x (CC BY)

Una plataforma web es el entorno de desarrollo de software empleado para diseñar y ejecutar un sitio web. En términos generales, una plataforma web consta de cuatro componentes básicos:

1.- El **sistema operativo,** bajo el cual opera el equipo donde se hospedan las páginas web y que representa la base misma del funcionamiento del computador. En ocasiones limita la elección de otros componentes.
2.- El **servidor web** es el software que maneja las peticiones desde equipos remotos a través de la Internet. En el caso de páginas estáticas, el servidor web simplemente provee el archivo solicitado, el cual se muestra en el navegador. En el caso de sitios dinámicos, el servidor web se encarga de pasar las solicitudes a otros programas que puedan gestionarlas adecuadamente.
3.- El **gestor de bases de datos** se encarga de almacenar sistemáticamente un conjunto de registros de datos relacionados para ser usados posteriormente.
4.- **Un lenguaje de programación interpretado** que controla las aplicaciones de software que corren en el sitio web. Diferentes combinaciones de los cuatro componentes señalados, basadas en las distintas opciones de software disponibles en el mercado, dan lugar a numerosas plataformas web, aunque, sin duda, hay dos que sobresalen del resto por su popularidad y difusión: LAMP y WISA.

La plataforma LAMP trabaja enteramente con componentes de **software libre** y no está sujeta a restricciones propietarias. El nombre **LAMP** surge de las iniciales de los

componentes de software que la integran:

- **L**inux: Sistema operativo.
- **A**pache: Servidor web.
- **M**ySQL: Gestor de bases de datos.
- **P**HP: Lenguaje interpretado PHP, aunque a veces se sustituye por Perl o Python.

La plataforma **WISA** está basada en tecnologías desarrolladas por la compañía Microsoft; se trata, por lo tanto, de **software propietario.** La componen los siguientes elementos:

- **W**indows: Sistema operativo.
- **I**nternet Information Services: servidor web.
- **S**QL Server: gestor de bases de datos.
- **A**SP o ASP.NET como lenguaje para scripting del lado del servidor.

Existen otras plataformas, como por ejemplo la configuración Windows-Apache-MySQL-PHP que se conoce como **WAMP.** Es bastante común pero sólo como plataforma de desarrollo local.

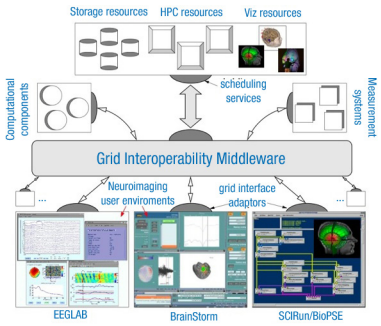
De forma similar, un servidor Windows puede correr con MySQL y PHP. A esta configuración se la conoce como plataforma **WIMP**.

Existen muchas otras plataformas que trabajan con distintos sistemas operativos (Unix, MacOS, Solaris), servidores web (incluyendo algunos que se han cobrado relativa popularidad como Lighttpd y LiteSpeed), bases de datos (Postgre SQL) y lenguajes de programación.

1.6.- Escalabilidad.

Caso práctico

Las aplicaciones web se ejecutan en un entorno donde el número de clientes que solicitan el servicio puede variar en gran medida en función del momento. Es por ello que hay una característica de esencial importancia como es la escalabilidad, al que **Juan** ha dedicado un apartado de su wiki para documentar esta característica.



sam_churchill (CC BY)

En el entorno en que se ubican las aplicaciones web, uno de los principales factores que puede afectar al rendimiento de las mismas es el número de usuarios, ya que éste puede verse incrementado de forma vertiginosa en un periodo de tiempo relativamente corto. El éxito o el fracaso de un sitio web orientado al usuario común vendrá determinado, entre otros aspectos, por el dimensionamiento del sistema sobre el que se instala y soporta el software que sustenta dicho sitio. En consecuencia, uno de los requisitos fundamentales de una aplicación web es que sea completamente escalable sin que un aumento de los recursos dedicados a la misma suponga modificación alguna en su comportamiento o capacidades.

La escalabilidad de un sistema web puede ser:

- **Verticalmente**: de manera ascendente "upgrades" a cada nodo.
- **Horizontalmente**: consiste en aumentar el número de nodos.
- **Cluster**: consiste en crear agrupaciones de servidores.

Escalabilidad vertical.

Habitualmente, la separación lógica en capas se implementa de tal forma que se permita una separación física de las mismas. Interponiendo elementos conectores que actúen de middlewares es posible distribuir la aplicación de forma vertical (una máquina por cada capa del sistema), e incluso si esto no fuera suficiente, distribuyendo los elementos de una misma capa entre distintas máquinas servidoras.

Escalabilidad horizontal.

Se trata de clonar el sistema en otra máquina de características similares y balancear la carga de trabajo mediante un dispositivo externo. El balanceador de carga puede ser:

- **Balanceador Software**: Por ejemplo, habitualmente encontramos un servidor web apache junto con el módulo `mod_jk`, que permite la redirección de las peticiones HTTP que a tal efecto sean configuradas entre las distintas máquinas que forman la granja de servidores. Este tipo de balanceadores examinan el paquete http e identifican la sesión del usuario, guardando registro de cuál de las máquinas de la granja se está encargado de servir a dicha sesión. Este aspecto es importante, dado que nos permite trabajar (de cara al diseño de la aplicación) apoyándonos en el objeto sesión propio del usuario y almacenando información relativa a la sesión del mismo, puesto que tenemos la garantía de que todas las peticiones de una misma sesión http van a ser redireccionadas hacia la misma máquina.
- **Balanceador hardware**: Se trata de dispositivos que, respondiendo únicamente a algoritmos de reparto de carga (Round Robin, LRU, etc.), redireccionan una petición http del usuario a la máquina que, según dicho algoritmo, convenga que se haga cargo de la petición. Son mucho más rápidos que los anteriores, dado que se basan en conmutación de circuitos y no examinan ni interpretan el paquete http. Sin embargo, el no garantizar el mantenimiento de la misma sesión de usuario en la misma máquina, condiciona seriamente el diseño, dado que fuerza a que la información relativa a la sesión del usuario sea almacenada por el implementador del mismo, bien en cookies o bien en base de datos.

phrenologist (CC BY-NC)

- **Balanceador hardware http**: Se trata de dispositivos hardware pero que examinan el paquete http y mantienen la relación usuario-máquina servidora. Mucho más rápidos que los balanceadores software, pero algo menos que los hardware, suponen hoy en día una de las soluciones más aceptadas en el mercado.

Cluster.

Con la aparición de los servidores de aplicaciones en cluster se abre una nueva capacidad de escalabilidad que, dependiendo de cómo se aplique, podría clasificarse como vertical u horizontal. Un cluster de servidores de aplicaciones permite el despliegue de una aplicación web corriente, de forma que su carga de trabajo vaya a ser distribuida entre la granja de servidores que forman el cluster, de modo transparente al usuario y al administrador. El cluster, mediante el mecanismo de replicación de sesión, garantiza que sea cual sea la máquina que sirva la petición http, tendrá acceso a la sesión del usuario (objeto `HttpSession` en java). Este tipo de sistemas, debido precisamente a la replicación de sesión, suele presentar problemas de rendimiento.

Autoevaluación

¿En qué tipo de escalabilidad se emplean los balanceadores de carga?

- ○ Horizontal.
- ○ Vertical.

Correcto. Los balanceadores se encargan de repartir la carga entre las distintas máquinas; pudiendo existir tres tipos de balanceadores :

- Balanceador software: Centran su trabajo basándose en las sesiones establecidas por los usuarios de la aplicación.
- Balanceador hardware: Se basan en algoritmos de reparto de carga; redireccionando las peticiones http del usuario a la máquina indicada por el algoritmo.
- Balanceador hardware-http: Situación intermedia entre los dos anteriores.

Revisa el concepto de escalabilidad vertical

Solución

1. Opción correcta
2. Incorrecto

2.- Servidor web Apache.

Caso práctico

En la empresa **BK programación**, **Ada** ha solicitado a **María** la instalación de un servidor web para albergar, entre otras cosas, la wiki que **Juan**, junto con sus amigos, están construyendo.

María, por su parte, ha decidido instalar el servidor web Apache por ser uno de los más empleados y aportar a **Juan** la información que pueda interesarle acerca de este servidor web para su wiki.

Un servidor web es un programa que se ejecuta de forma continua en un ordenador (también se utiliza el término para referirse al ordenador que lo ejecuta), se mantiene a la espera de peticiones por parte de un cliente (un navegador de Internet) y contesta a estas peticiones de forma adecuada, sirviendo una página web que será mostrada en el navegador o mostrando el mensaje correspondiente si se detectó algún error.

Uno de los servidores web más populares del mercado y el más utilizado actualmente es Apache, de código abierto y gratuito, disponible para Windows y GNU/Linux, entre otros.

En cuanto a su arquitectura podemos destacar lo siguiente:

- Estructurado en módulos.
- Cada módulo contiene un conjunto de funciones relativas a un aspecto concreto del servidor.
- El archivo binario `httpd` contiene un conjunto de módulos que han sido compilados.
- La funcionalidad de estos módulos puede ser activada o desactivada al arrancar el servidor.
- Los módulos de Apache se pueden clasificar en tres categorías:
  - **Módulos base**: Se encargan de las funciones básicas.

- **Módulos multiproceso**: Encargados de la unión de los puertos de la máquina, aceptando las peticiones y atendiéndolas.
- **Módulos adicionales**: se encargan de añadir funcionalidad al servidor.

El servidor Apache se desarrolla dentro del proyecto HTTP Server (**`httpd`**) de la Apache Software Foundation. La licencia de software, bajo la cual el software de la fundación Apache es distribuido, es una parte distintiva de la historia de Apache HTTP Server y de la comunidad de código abierto. La **Licencia Apache** permite la distribución de derivados de código abierto y cerrado a partir de su código fuente original.

Para saber más

Esta web sirve como manual de referencia, guía de usuario, tutoriales prácticos, etc., sobre el servidor web Apache. Se trata de la web oficial de Apache Software Foundation.

Documentación Servidor Apache en español

2.1.- Instalación y configuración.

Caso práctico

En la empresa **BK programación**, **Ada** ha solicitado a **María** la instalación de un servidor web para albergar, entre otras cosas, la wiki que **Juan**, junto con sus amigos, está construyendo.

María debe instalar el servidor Apache debido a que es uno de los más empleados. Se trata de una herramienta de código libre y que funciona en multitud de plataformas; en este caso, será instalado en una máquina Debian 6.0.1 Squeeze, quedando el servidor totalmente operativo.

Apache Software Foundation (Apache 2.0)

Vamos a realizar la instalación de un servidor Apache en una máquina con la distribución Ubuntu 20.04 LTS. Dicha máquina puede ser una máquina real, una máquina virtual o un contenedor.

Empezamos por identificarnos en la máquina con el usuario `root` y, a continuación, ejecutamos:

```
# apt-get update && apt-get install systemd apache2
```

Debido a que pretendemos montar una plataforma LAMP, por sus ventajas derivadas de las características del software libre, instalaremos también los siguientes componentes: MySQL y PHP.

```
# apt-get install php libapache2-mod-php php-mysql
# apt-get install mysql-server
```

También es buena idea hacer que Apache arranque autiomáticamente:

```
# systemctl enable apache2
```

Una vez instalado, para verificar si funciona, podemos hacerlo desde un navegador, escribiendo en la barra de direcciones **`http://localhost`** o **`http://127.0.0.1`**, o bien, si accedemos desde otro equipo de la red a la dirección IP de esta máquina, deberíamos obtener una página similar a la siguiente:

Elaboración propia (CC0)

Por defecto, Apache sirve las páginas web que están en la carpeta "**`/var/www/html/`**"; si nos situamos en esa carpeta, encontramos un archivo "**`index.html`**" que es el que contiene la página por defecto de Apache. En esta carpeta podemos crear nuevas carpetas en donde ubicaremos nuevas páginas web que deseamos servir, todas ellas accesibles a través del puerto 80.

Si la única pretensión es servir una página web, podemos integrar su contendido aquí. En caso que se pretenda servir más páginas web, es más recomendable la utilización de los **Hosts Virtuales**; para ello accedemos a la carpeta "**`/etc/apache2/sites-enabled`**", donde hay un fichero llamado "**`000-default`**", que nos va a servir de ejemplo para la creación de hosts virtuales, los cuales van a permitir servir varias web desde una sola dirección IP utilizando para cada una un puerto distinto.

Apache se configura colocando directivas en archivos de configuración de texto plano. El archivo principal de configuración se llama **`apache2.conf`**. Además, se pueden añadir otros archivos de configuración mediante la directiva "**`Include`**", y se pueden usar comodines para incluir muchos archivos de configuración. Todas las directivas deben colocarse en alguno de esos archivos de configuración. Apache2 sólo reconocerá los cambios realizados en los archivos principales de configuración cuando se inicie o se reinicie.

Como ya hemos comentado, el archivo de configuración predeterminado de Apache2 es **`/etc/apache2/apache2.conf`**. Se puede editar este archivo para configurar el servidor Apache2, para configurar el número de puerto, la raíz de documentos, los módulos, los archivos de registros, los hosts virtuales, etc. Pasamos a ver alguna de las principales directivas:

- **`ServerTokens`**, para configurar la cantidad de información que Apache aporta sobre sí mismo.
- **`ServerSignature`**, para indicar datos sobre Apache en el pie de los mensajes de error.
- **`Alias`** permite direccionar a una carpeta que puede estar fuera del árbol de directorios especificado en **DocumentRoot.**
- **`userDir`** permite redireccionar al directorio personal del usuario si se recibe una solicitud de tipo **`~usuario`**.

Para modificar el servidor virtual predeterminado, editamos el archivo **`/etc/apache2/sites-available/default`**. En el caso de querer configurar un nuevo servidor o sitio virtual, copiaríamos ese archivo dentro del mismo directorio con el nombre que se haya elegido, y

editaríamos el nuevo archivo para configurar el nuevo sitio usando algunas de las directivas que se describen a continuación:

- **`ServerName`**, en el caso de no tener un dominio registrado emplearíamos **`localhost`**.
- **`CustomLog`** define el archivo **`.log`** donde se guardan los logs de acceso.
- **`ServerAdmin`** especifica la dirección de correo del administrador del servidor. El valor por omisión es **`webmaster@localhost.`**
- **`Listen`** especifica el puerto (y, opcionalmente, la dirección IP) por el que escuchará Apache2. La directiva se puede encontrar y cambiar en su propio archivo de configuración, **`/etc/apache2/ports.conf`**.
- **`DocumentRoot`** especifica dónde Apache debe buscar los archivos que forman el sitio. El valor predeterminado es **`/var/www`**.
- **`RedirectMatchen`** en **`/etc/apache2/apache2.conf`**, las peticiones se redirigirán a **`/var/www/apache2-default`**, que es donde reside el sitio predeterminado de Apache2. Cambiar este valor en el archivo de host virtual implica crear ese directorio si fuese necesario.

Debes conocer

A continuación se exponen brevemente dos aplicaciones fundamentales para la gestión de servidores en general y LAMP en particular. **phpMyAdmin** y **Webmin**. Su instalación es muy recomendable en el caso de que gestionemos nuestros propios servidores Web. en el caso de que usemos un proveedor de alojamiento, normalemente ya tendremos phpMyAdmin, aunque no Webmin ya que la administración del sistema correrá de su parte.

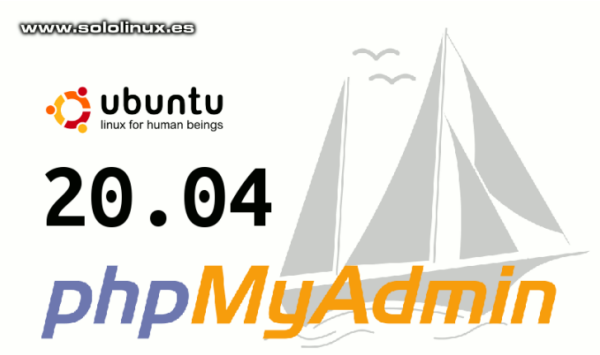
phpMyAdmin

phpMyAdmin es una herramienta fundamental en los servidores LAMP para configurar el servidor MySQL. Nos permite configurar prácticamente todos los aspectos de MysSQL de manera remota a la vez que mantiene la facilidad de uso de una interfaz gráfica.

Al tener una interfaz Web, podemos acceder desde un navegador sin tener que instalar nigún software específico adicional.

En el siguiente enlace puedes acceder a la web oficial de phpMyAdmin.

En el siguiente enlace podemos ver una descripción detallada de cómo instalar phpMyAdmin sobre Ubuntu 20.04:



Webmin

Webmin es una herramienta de configuración de sistemas accesible vía web. Nos permite configurar prácticamente todos los servicios de nuestros servidores de manera remota a la vez que mantiene la facilidad de uso de una interfaz gráfica.

Al ser modular, podemos adaptar la instalación a prácticamente cualquier configuración de ervidores que tengamos, cargando los módulos correspondientes. Por otro lado, al tener una interfaz Web, podemos acceder desde un navegador sin tener que instalar nigún software específico adicional.

En el siguiente enlace puedes acceder a la página oficial de Webmin.

En el siguiente enlace podemos ver una descripción detallada de cómo instalar Webmin sobre Ubuntu 20.04:

2.2.- Iniciar Apache.

Caso práctico

Ahora **María** debe tener claro cómo realizar cambios en la configuración de Apache, que cree que ya lo tiene claro y que dichos cambios se vean reflejados en el servidor. Así que se pone manos a la obra.

The Apache Software Foundation (Apache 2.0)

Una vez hayamos instalado Apache, podemos probar su configuración por defecto de la siguiente forma:

```
# apachectl configtest
```

Si todo está correcto debería devolver un mensaje del tipo "Syntax Ok" y el servidor debería estar arrancado, con lo cual, si en un navegador introducimos la URL: **`http://localhost`** veríamos la página de bienvenida de Apache.

Si el puerto especificado en la directiva **`Listen`** del fichero de configuración es el que viene por defecto, es decir, el puerto **80** (o cualquier otro puerto por debajo del 1024), entonces es necesario tener privilegios de usuario **`root`** (superusuario) para iniciar Apache, de modo que pueda establecerse una conexión a través de esos puertos privilegiados. Una vez que el servidor Apache se ha iniciado y ha completado algunas tareas preliminares, tales como abrir sus ficheros log, lanzará varios procesos, procesos hijo, que hacen el trabajo de escuchar y atender las peticiones de los clientes. El proceso principal, **`httpd`**, continúa ejecutándose como **`root`**, pero los procesos hijo se ejecutan con menores privilegios de usuario.

El demonio **`httpd`** se debería invocar empleando el script de control **`apachectl`**, que es el que se encarga de fijar variables de entorno y pasa al demonio (**`httpd`**) cualquier opción que se le pase cómo argumento por línea de comandos.

El script **`apachectl`** es capaz de interpretar los argumentos **`start`**, **`restart`**, y **`stop`** y traducirlos en las señales apropiadas para **httpd**, y es el comando genérico para Linux, sin embargo para Debian y Ubuntu es mas habitual usar los comandos que se comentan a continuación.

También podemos ver el estado del servicio con el comando:

```
# systemctl status apache2
```

Y nos mostrará si el servidor está arrancado, desde cuando, la versión, etc.

Si en cualquier momento deseásemos parar, reiniciar o arrancar el servidor, podríamos emplear los siguientes comandos respectivamente**:**

```
# /etc/init.d/apache2 stop
# /etc/init.d/apache2 restart
# /etc/init.d/apache2 start
```

O bien, para versiones mas actuales Debian y Ubuntu:

```
# systemctl stop apache2
# systemctl restart apache2
# systemctl start apache2
```

Reflexiona

Una vez instalado el servidor Apache, es necesario acceder a su funcionalidad y gestionarlo como si de un servicio se tratase, de modo, que cuando establecemos cambios en su configuración, los mismos se vean reflejados.

¿Será necesario reiniciar el servicio Apache si, mediante la creación de un host virtual, hemos cambiado el puerto por el que escucha?

3.- Aplicaciones web y servidores de aplicaciones.

Caso práctico

Hoy en día existen innumerables aplicaciones web. Por eso, en la wiki que está construyendo, **Juan** ha decidido dedicar un apartado a dos conceptos de vital importancia: Aplicación web y Servidor de Aplicaciones; ambos conceptos estrechamente relacionados.

Sun Microsystems (Dominio público)

Se define una aplicación web como una aplicación informática que se ejecuta en un entorno web, de forma que se trata de una aplicación cliente-servidor junto con un protocolo de comunicación previamente establecido:

- **Cliente**: navegador.
- **Servidor**: servidor web.
- **Comunicación**: protocolo HTTP.

Un **servidor de aplicaciones** es un software que proporciona aplicaciones a los equipos o dispositivos cliente, por lo general, a través de Internet y utilizando el protocolo HTTP. Los servidores de aplicación se distinguen de los servidores web en el uso extensivo del contenido dinámico y por su frecuente integración con bases de datos.

Un servidor de aplicaciones también es una máquina en una red de computadores que ejecuta determinadas aplicaciones, gestionando la mayor parte de las funciones de acceso a los datos de la aplicación.

Las principales ventajas de la tecnología de los servidores de aplicaciones es la **centralización** y **disminución** de la complejidad en el desarrollo de las aplicaciones, ya que no necesitan ser programadas, sino que son ensambladas desde bloques provistos por el servidor de aplicación.

Otra de las ventajas es la **integridad de datos y código** ya que, al estar centralizada en una o un pequeño número de máquinas servidoras, las actualizaciones están garantizadas para todos los usuarios.

El término servidor de aplicaciones se aplica a todas las plataformas. Dicho término se utiliza para referirse a los servidores de aplicaciones basadas en web, como el control de las plataformas de comercio electrónico integrado, sistemas de gestión de contenido de sitios web y asistentes o constructores de sitios de Internet.

Uno de los ejemplos destacados es el de Sun Microsystems, la plataforma J2EE. Los servidores de aplicaciones Java se basan en la Plataforma Java ™ 2 Enterprise Edition (J2EE ™). J2EE utiliza un modelo de este tipo y en general, incluye un nivel Cliente, un nivel Medio, y un EIS. El servidor de tipo Cliente puede contener una o más aplicaciones o navegadores. La Plataforma J2EE es del Nivel Medio y consiste en un servidor web y un servidor EJB. (Estos servidores son también llamados "contenedores".) También podría haber subniveles adicionales en el nivel intermedio. El nivel del Sistema Enterprise Information System (EIS, o "Sistema de Información Empresarial") contiene las aplicaciones existentes, archivos y bases de datos.

3.1.- El servidor de aplicaciones Tomcat.

Caso práctico

Instalar un servidor de aplicaciones web para la empresa **BK programación** ha sido una solicitud que **Ada** había propuesto hace tiempo a **María**, quien, llegado el momento, y habiendo estudiado las posibles opciones, se ha decidido por instalar el servidor Tomcat

shane_curcuru (CC BY-NC)

Tomcat es el servidor web (incluye el servidor Apache) y de aplicaciones del proyecto Yakarta, con lo cual, gestiona las solicitudes y respuestas HTTP y, además, es servidor de aplicaciones o contenedor de Servlets y JSP.

Incluye el compilador Jasper, que compila JSP covirtiéndolas en servlets.

Tomcat es un contenedor de servlets con un entorno JSP. Un contenedor de servlets es un shell de ejecución que maneja e invoca servlets por cuenta del usuario. Podemos dividir los contenedores de servlets en:

1.- Contenedores de servlets **stand-alone** (independientes): Estos son una parte integral del servidor web. Este es el caso en el que se usa un servidor web basado en Java, por ejemplo, el contenedor de servlets es parte de JavaWebServer (actualmente sustituido por **iPlanet**). Por defecto Tomcat trabaja en este modo, sin embargo, la mayoría de los servidores no están basados en Java.
2.- Contenedores de servlets **dentro-de-proceso:** El contenedor servlets es una combinación de un plugin para el servidor web y una implementación de contenedor Java. El plugin del servidor web abre una JVM (Máquina Virtual Java) dentro del espacio de direcciones del servidor web y permite que el contenedor Java se ejecute en él. En el caso de que una petición debiera ejecutar un servlet, el plugin toma el control sobre la petición y lo pasa al contenedor Java (usando JNI). Un contenedor de este tipo es adecuado para servidores multi-thread de un sólo proceso y proporciona un buen rendimiento pero está limitado en escalabilidad.
3.- Contenedores de servlets **fuera-de-proceso:** El contenedor servlets es una combinación de un plugin para el servidor web y una implementación de contenedor Java que se ejecuta en una JVM fuera del servidor web. El plugin del servidor web y el JVM del contenedor Java se comunican usando algún mecanismo IPC (normalmente sockets TCP/IP). Si una cierta petición tuviese que ejecutar un servlets, el plugin toma el control sobre la petición y lo pasa al contenedor Java (usando IPCs). El tiempo de

respuesta en este tipo de contenedores no es tan bueno como el anterior, pero obtiene mejores rendimientos en otras cosas (escalabilidad, estabilidad, etc.).

Tomcat puede utilizarse como un contenedor solitario (principalmente para desarrollo y depuración) o como plugin para un servidor web existente (actualmente soporta los servidores Apache, IIS). Esto significa que siempre que despleguemos Tomcat tendremos que decidir cómo usarlo y, si seleccionamos las opciones 2 o 3, también necesitaremos instalar un adaptador de servidor web.

Autoevaluación

Las funciones del Servidor Apache y las funciones del servidor Tomcat, ¿son equivalentes?

- ○ Sí.
- ○ No.

Recopila y analiza documentación relativa a cada uno de los servidores.

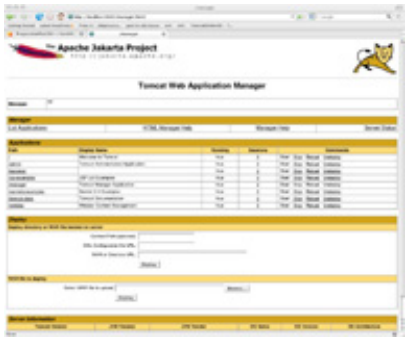
Básicamente, el servidor Apache es únicamente un servidor web, mientras que el servidor Tomcat es un servidor de aplicaciones.

Solución

1. Incorrecto
2. Opción correcta

3.1.1.- Instalación y configuración básica.

En primer lugar, destacar que para instalar cualquier versión de Tomcat es necesario tener instalado JDK (Kit de desarrollo de Java), ya que el objetivo es que las peticiones a Apache se redirijan a Tomcat empleando un conector proporcionado por Java en este caso.



nchenga (CC BY-NC)

Empezamos actualizando el sistema:

```
$ sudo apt update
```

Instalamos el siguiente paquete por ser el que más se adapta a nuestras necesidades:

```
$ sudo apt install default-jdk
```

Crearemos un grupo **`tomcat`**, y un usuario **`tomcat`** miembro del grupo anterior, con un directorio de inicio **`/opt/tomcat`** (donde instalaremos Tomcat) y un shell de **`/bin/false`** (de modo que nadie pueda iniciar sesión en la cuenta):

```
$ sudo groupadd tomcat
$ sudo useradd -s /bin/false -g tomcat -d /opt/tomcat tomcat
```

Ahora descargaremos e instalaremos Tomcat. Para descargar la última versión de Tomcat 9.0.x, iremos a su página de descargas. En este ejemplo vamos a usarla última versión **9.0.35**, copiamos el enlace del archivo **`.tar.gz`**.

Nos movemos al directorio **`/tmp`** y descargamos el archivo: (si no disponemos de **`curl`**, lo instalamos con **`sudo apte install curl`**)

```
$ cd /tmp
$ curl -O https://apache.brunneis.com/tomcat/tomcat-9/v9.0.35/bin/apache-tomcat-9.0.35.tar.gz
```

Una vez que se complete la descarga, extraeremos el archivo comprimido en el directorio **`/opt/tomcat`**:

```
$ sudo mkdir /opt/tomcat
$ sudo tar xzvf apache-tomcat-*tar.gz -C /opt/tomcat --strip-components=1
```

Nos cambiamos al directorio en el que desempaquetamos la instalación de Tomcat y le damos propiedad sobre todo el directorio de instalación al grupo **`tomcat`**:

```
$ cd /opt/tomcat
$ sudo chgrp -R tomcat /opt/tomcat
```

A continuación, le proporcionamos al grupo **`tomcat`** acceso de lectura al directorio **`conf`** y a todos sus contenidos, y acceso de ejecución al directorio, además nos aseguramos de que el usuario **`tomcat`** sea el propietario de los directorios **`webapps`**, **`work`**, **`temp`** y **`logs`**:

```
$ sudo chmod -R g+r conf
$ sudo chmod g+x conf
$ sudo chown -R tomcat webapps/ work/ temp/ logs/
```

Ahora tenemos que crear el archivo **`tomcat.service`** en el directorio **`/etc/systemd/system`**:

```
$ sudo nano /etc/systemd/system/tomcat.service
```

Y escribiremos lo siguiente:

```
[Unit]
Description=Apache Tomcat Web Application Container
After=network.target

[Service]
Type=forking

Environment=JAVA_HOME=/usr/lib/jvm/java-1.11.0-openjdk-amd64
Environment=CATALINA_PID=/opt/tomcat/temp/tomcat.pid
Environment=CATALINA_HOME=/opt/tomcat
Environment=CATALINA_BASE=/opt/tomcat
Environment='CATALINA_OPTS=-Xms512M -Xmx1024M -server -XX:+UseParallelGC'
Environment='JAVA_OPTS=-Djava.awt.headless=true -Djava.security.egd=file:/dev/./urandom'

ExecStart=/opt/tomcat/bin/startup.sh
ExecStop=/opt/tomcat/bin/shutdown.sh

User=tomcat
Group=tomcat
UMask=0007
RestartSec=10
Restart=always

[Install]
WantedBy=multi-user.target
```

En la variable de entorno **`JAVA_HOME`** tendremos que poner la ruta que tengamos en nuestro equipo.

Por último, recargamos el demonio **`systemd`** para que reciba información sobre nuestro archivo de servicio e iniciamos el servicio Tomcat:

```
$ sudo systemctl daemon-reload
$ sudo systemctl start tomcat
```

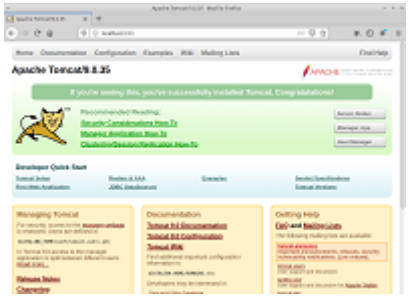
Para comprobar que el servicio está iniciado sin errores podemos escribir:

```
$ sudo systemctl status tomcat
```

Y nos mostrará si el servidor está arrancado, desde cuando, la versión, etc.

En Tomcat, la gestión del servicio se realiza a través del script incluido llamado **catalina**, al que le podemos proporcionar los parámetros **`start`** y **`stop`**, con lo que arrancaríamos o pararíamos el servicio manualmente.

Para comprobar que nuestro servidor está ya escuchando, introducimos en un navegador la URL: **`http://localhost:8080`** o **`http://127.0.0.1:8080`**, y éste debería mostrar la página de inicio de Tomcat similar a la siguiente:



Elaboración propia

3.1.2.- Iniciar Tomcat.

betacontinua (CC BY-NC-SA)

Tomcat va a estar escuchando en el puerto **8080** y va a tener su propio directorio de trabajo. La misión de **apache2** va a ser interceptar todas las peticiones en el puerto **80** y derivar las que considere necesarias a Tomcat; de este modo observamos la ventaja de la escalabilidad, ya que apache, al funcionar como proxy, puede tener una batería de tomcats a los que balancear las conexiones, haciendo que, si nuestras necesidades crecen, nuestras máquinas puedan ampliarse en número siendo completamente transparente para los usuarios.

Apache por defecto busca los ficheros en **/var/www/html**, Tomcat trabaja sobre la carpeta **/opt/tomcat/webapps**. La petición de una URL se puede gestionar, parte por apache y parte por Tomcat, por lo que vamos a cambiar la carpeta por defecto de trabajo para unificarlo. Para ello editamos el fichero **/opt/tomcat/conf/conf/server.xml**.

```
$ sudo nano /opt/tomcat/conf/server.xml
```

en donde encontraremos una línea con "**Host name=**" y debería tener:

```
Host name=”localhost” appBase=”wwebapps"
```

Para usar la aplicación de administración web que viene con Tomcat, debemos añadir un inicio de sesión a nuestro servidor de Tomcat. Para ello editaremos el archivo **tomcat-users.xml**:

```
$ sudo nano /opt/tomcat/conf/tomcat-users.xml
```

Deberíamos añadir un usuario que pueda acceder a **manager-gui** y **admin-gui** (aplicaciones web que vienen con Tomcat). Podemos definir un usuario, por ejemplo como el que se muestra a continuación, entre las etiquetas **tomcat-users**, y poniendo nombres de usuarios y contraseñas seguras:

```
<tomcat-users . . .>
    <user username="admin" password="password" roles="manager-gui,admin-gui"/>
</tomcat-users>
```

Si en cualquier momento deseásemos parar o arrancar el servidor, podríamos emplear los siguientes comandos respectivamente:

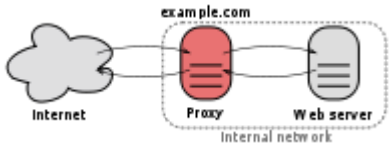
```
$ sudo systemctl stop tomcat
$ sudo systemctl start tomcat
```

O bien:

```
$ sudo service tomcat stop
$ sudo service tomcat start
```

Debes conocer

Proxy inverso. Un proxy inverso es un tipo de servidor proxy que recupera recursos en nombre de un cliente desde uno o más servidores. Estos recursos son entonces redirigidos al cliente como si se originaran en el propio servidor Web. Son también conocidos como "servidores de paso" o gateway.



H2g2bob (CC0)

En tales escenarios, el propio servidor no genera contenido o aloja datos, en su lugar el contenido se obtiene de uno o varios servidores backend, que normalmente no tienen conexión directa con redes externas.

Cuando el proxy inverso recibe una petición de un cliente, reenvía esta petición a uno de estos servidores backend, que gestiona la petición, genera el contenido y entonces envía este contenido de vuelta al proxy, que entonces genera la respuesta HTTP definitiva que se envía de vuelta al cliente.

Existen muchas razones para usar esta implementación, pero generalmente las razones típicas se deben a seguridad, alta disponibilidad, balanceo de carga, y centralización de autenticación/autorización. Es crítico en estas implementaciones que la arquitectura y el diseño de la infraestructura de los backend (esos servidores que son los que acaban gestionando las peticiones) estén aislados y protegidos del exterior; en cuanto al cliente se refiere, el proxy inverso és la única fuente de todo el contenido.

Apache2 se puede configurar como proxy inverso, para ello cargaremos los módulos siguientes para poder conseguir que Apache funcione como proxy:

```
$ sudo a2enmod proxy
$ sudo a2enmod proxy_ajp
$ sudo a2enmod proxy_balancer
$ sudo /etc/init.d/apache2 restart
```

ajp es un protocolo de comunicación interno y muy rápido que usa conexiones TCP persistentes. Es este protocolo el que vamos a utilizar para comunicar apache2 con Tomcat, aunque podría ser utilizado HTTP, indicando que pregunte en el **8080**. El puerto de trabajo por defecto para Tomcat es el **8009**, aunque este puede ser variado desde **/opt/tomcat/conf/server.xml**.

Modificamos el fichero de configuración del virtualhost que se pretenda utilizar, empleando el establecido por defecto.

```
$ sudo nano /etc/apache2/sites-enabled/000-default.conf
```

en donde añadimos lo siguiente:

```
<Proxy balancer://tomcat_cluster>
  Order allow,deny
  Allow from all
  BalancerMember ajp://localhost:8009
</Proxy>
ProxyPreserveHost On
ProxyPass /phpmyadmin/ !
ProxyPass / balancer://tomcat_cluster/
ProxyPassReverse / balancer://tomcat_cluster/
```

Pasamos a definir cada uno de los parámetros anteriores:

- **Proxy balancer://tomcat_cluster**: Estamos definiendo un cluster con nombre "Tomcat_cluster"
- **BalancerMember ajp://localhost:8009**: Se define un miembro a Tomcat_cluster, protocolo, IP y puerto.
- **ProxyPass / balancer://tomcat_cluster/**: "/" y todo lo que cuelgue de ella, sea pasado al cluster del tomcat para que lo procese él.
- **ProxyPreserveHost On**: Mantiene la cabecera http host original, en vez de reescribirla.

Lo último es cambiar de **/etc/apache2/sites-enabled/000-default.conf** el "**DocumentRoot**" y "**<Directory /var/www/html>**" para que apunten a /var/www/ROOT, de esta manera podemos decidir qué parte gestiona cada aplicación desde un solo directorio.

4.- Estructura y despliegue de una aplicación web.

Caso práctico

Una vez la empresa **BK programación** dispone de un servidor de aplicaciones, **Ada** considera necesario formar al personal acerca de cómo desplegar aplicaciones, especificar la estructura a seguir, etc.; para lo cual ha indicado a **Juan** que documente en la wiki un punto en donde se explique cada uno de estos pasos.

Una aplicación web está compuesta de una serie de servlets, páginas JSP, ficheros HTML, ficheros de imágenes, ficheros de sonidos, texto, clases, etc.; de forma que todos estos recursos se pueden empaquetar y ejecutar en varios contenedores distintos.

Un servlet es una aplicación java encargada de realizar un servicio específico dentro de un servidor web. La especificación Servlet 2.2 define la estructura de directorios para los ficheros de una aplicación web. El directorio raíz debería tener el nombre de la aplicación y define la raíz de documentos para la aplicación web. Todos los ficheros debajo de esta raíz pueden servirse al cliente excepto aquellos ficheros que están bajo los directorios especiales **`META-INF`** y **`WEB-INF`** en el directorio raíz. Todos los ficheros privados, al igual que los ficheros **`.class`** de los servlets, deberían almacenarse bajo el directorio **`WEB-INF.`**

Durante la etapa de desarrollo de una aplicación web se emplea la estructura de directorios, a pesar de que luego en la etapa de producción, toda la estructura de la aplicación se empaqueta en un archivo **`.war`**.

El código necesario para ejecutar correctamente una aplicación web se encuentra distribuido en una estructura de directorios, agrupándose ficheros según su funcionalidad. Un ejemplo de la estructura de carpetas de una aplicación web puede ser el siguiente:

```
/index.jsp
/WebContent/jsp/welcome.jsp
/WebContent/css/estilo.css
/WebContent/js/utils.js
/WebContent/img/welcome.jpg
/WEB-INF/web.xml
```

```
/WEB-INF/struts-config.xml
/WEB-INF/lib/struts.jar
/WEB-INF/src/com/empresa/proyecto/action/welcomeAction.java
/WEB-INF/classess/com/empresa/proyecto/action/welcomeAction.class
```

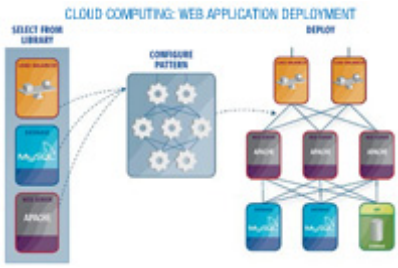
De forma genérica podríamos decir que una aplicación web se estructura en tres capas:

1. Navegador web.
2. Tecnología web dinámica (PHP, Java Servlets, ASP, etc.)
3. Base de datos encargada de almacenar de forma permanente y actualizada la información que la aplicación web necesita.

4.1.- Archivos WAR.

Caso práctico

Una vez la empresa **BK programación** dispone de un servidor de aplicaciones, **Ada** considera necesario formar al personal acerca de cómo desplegar aplicaciones, especificar la estructura a seguir, etc. Un punto importante a detallar es la distribución de aplicaciones web mediante los archivos WAR; **Juan** lo detalla en su wiki.



louisvolant (CC BY-NC-SA)

Su nombre procede de Web Application Archive (Archivo de Aplicación Web); permiten empaquetar en una sola unidad aplicaciones web de Java completas, es decir que su contenido sea:

- Servlets y JSP.
- Contenido estático: HTML, imágenes, etc.
- Otros recursos web.

Aportan como ventaja, la simplificación del despliegue de aplicaciones web, debido a que su instalación es sencilla y solamente es necesario un fichero para cada servidor en un cluster, además de incrementar la seguridad ya que no permite el acceso entre aplicaciones web distintas.

Su estructura es la siguiente:

- `/`: En la carpeta raíz del proyecto se almacenan elementos empleados en los sitios web: documentos html, hojas de estilo y los elementos JSP (`*.html`, `*.jsp`, `*.css`).
- `/WEB-INF/`: Aquí se encuentran los elementos de configuración del archivo `.war` como pueden ser: la página de inicio, la ubicación de los servlets, parámetros adicionales para otros componentes. El más importante de éstos es el archivo `web.xml`.
- `/WEB-INF/classes/`: Contiene las clases Java empleadas en el archivo `.war` y, normalmente, en esta carpeta se encuentran los servlets.
- `/WEB-INF/lib/`: Contiene los archivos JAR utilizados por la aplicación y que normalmente son las clases empleadas para conectarse con la base de datos o las empleadas por librerías de JSP.

Para generar archivos `.war` se pueden emplear diversas herramientas desde entorno IDE. Por ejemplo, encontramos: **NetBeans** y **Eclipse**, ambos Open-Source y también **Jbuilder**

de Borland, **Jdeveloper** de Oracle; otro modo de construir archivos `.war` es mediante **Ant**. Se trata de una herramienta Open-Source que facilita la construcción de aplicaciones en Java. No es considerado un IDE pero para los que conocen el entorno Linux, es considerado el `make` de Java.

Autoevaluación

Un archivo `.WAR` ...

- [ ] Es un archivo comprimido que se puede generar con cualquier tipo de compresor, por ejemplo winzip, winrar, tar, etc.
- [ ] Es una aplicación web formada únicamente por archivos `.html`.
- [ ] Es un archivo en el cual se empaqueta en una sola unidad, aplicaciones web completas.
- [ ] Es un archivo que engloba el protocolo de comunicación de las aplicaciones web generadas con Java.

Mostrar retroalimentación

Solución

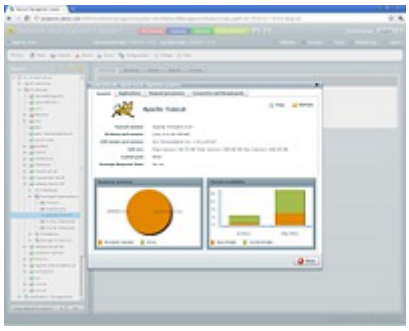
1. Incorrecto
2. Incorrecto
3. Correcto
4. Incorrecto

4.2.- Despliegue de aplicaciones con Tomcat.

Caso práctico

Una vez que la empresa **BK programación** dispone de un servidor de aplicaciones, **Ada** considera necesario formar al personal acerca de cómo desplegar aplicaciones, especificar la estructura a seguir, etc.

El empleo del servidor de aplicaciones Tomcat es una actividad bastante común y útil para empezar a desplegar aplicaciones web. **María** ha decidido emplear dicho servidor en su empresa y aquí nos proporciona una serie de pasos.



Verax Systems Corp. (CC BY-NC-ND)

Una aplicación web puede ser desplegada empleando uno de los siguientes métodos:

- Por medio de archivos `.war` (Web Archive).
- Editando los archivos `web.xml` y `server.xml`. Este método es el que se pasa a tratar a continuación.

Los directorios que forman una aplicación compilada suelen ser : `www`, `bin`, `src`, `tomcat` y `gwt-cache`.

La carpeta `www` contiene, a su vez, una carpeta con el nombre y ruta del proyecto que contiene los ficheros que forman la interfaz (`.html`, `.js`, `.css`, etc.). La carpeta `bin` contiene las clases de java de la aplicación.

Para desplegar la aplicación en Tomcat:

1.- Copiar la carpeta contenida en `www` (con el nombre del proyecto) en el directorio `webapps` de Tomcat.
2.- Renombrar la nueva carpeta así creada en Tomcat con un nombre más sencillo. Esa será la carpeta de la aplicación en Tomcat.
3.- Crear, dentro de dicha carpeta, otra nueva, y darle el nombre `WEB-INF` (respetando las mayúsculas).

4.- Crear, dentro de **`WEB-INF`**, otros dos subdirectorios, llamados **`lib`** y **`classes`**.
5.- Copiar en **`lib`** todas las librerías (**`.jar`**) que necesite la aplicación para su funcionamiento.
6.- Copiar el contenido de la carpeta **`bin`** de la aplicación en el subdirectorio **`WEB-INF/classes`** de Tomcat.
7.- Crear en **`WEB-INF`** un fichero de texto llamado **`web.xml`**, con las rutas de los servlets utilizados en la aplicación.
8.- A la aplicación ya puede accederse en el servidor, poniendo en el navegador la ruta del fichero **`.html`** de entrada, que estará ubicado en la carpeta de la aplicación en Tomcat.

Para saber más

En la web que a continuación se detalla se muestran los pasos implicados en el despliegue de un servlet. Describe cómo tomar un servlet y crear una aplicación web, tanto en formato expandido como en un WAR. Ilustra cómo desplegar una aplicación web en Apache Tomcat y en WebLogic Server 6.0, un completo servidor de aplicaciones J2EE.

Despliegue de aplicaciones en Tomcat

4.3.- Descriptor de despliegue.

Caso práctico

Una vez la empresa **BK programación** dispone de un servidor de aplicaciones, **Ada** considera necesario formar al personal acerca de cómo desplegar aplicaciones, especificar la estructura a seguir, etc.

El empleo del servidor de aplicaciones Tomcat es una actividad bastante común y útil para empezar a desplegar aplicaciones web. **Juan** ha dedicado en este punto de la wiki explicar en qué consiste el descriptor del despliegue.

Un Descriptor de Despliegue es un documento XML que describe las características de despliegue de una aplicación, un módulo o un componente. Por esto, la información del descriptor de despliegue es declarativa, y esta puede ser cambiada sin la necesidad de modificar el código fuente.

Cualquier aplicación web tiene que aportar un descriptor de despliegue situado en **`WEB-INF/web.xml`**; en el caso concreto de Tomcat el descriptor **`<TOMCAT_HOME>/conf/web.xml`** es un descriptor por defecto que se ejecuta siempre antes del descriptor de la aplicación y, solamente, debería contener información general y no específica de la aplicación.

Un ejemplo de descriptor de despliegue puede ser el siguiente archivo **`web.xml`**:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE web-app PUBLIC
  "-//Sun Microsystems, Inc.//DTD Web Application 2.2//EN"
  "http://java.sun.com/j2ee/dtds/web-app_2_2.dtd">
<web-app>
  <!-- Tus definiciones van aquí -->
</web-app>
```

Situadas entre las etiquetas **`<web-app>`** y **`/<web-app>`** estarían los descriptores de despliegue de servlets, los cuales deben contener las siguientes etiquetas en el siguiente orden:

```
<servlet>
  <servlet-name>nombre</servlet-name>
  <servlet-class>package.nombre.MiClass</servlet-class>
</servlet>
```

Para probar el servlet, una vez arrancado el servidor Tomcat, abrimos un navegador web, en el cual escribiríamos una URL con el siguiente formato:

```
http://{address}:{port}/{servletName}
```

por ejemplo:

```
http://localhost:8080/Servlet_de_prueba
```

Citas para pensar

> *No hay secretos para el éxito. Éste se alcanza preparándose, trabajando arduamente y aprendiendo del fracaso.*
>
> *Colin Powell*

Autoevaluación

Escribe el nombre de tecnologías asociadas a aplicaciones siguiendo los enunciados siguientes:

1. La Interface Común de Entrada es uno de los estándares más antiguos en Internet para trasladar la información desde una página web a un servidor web:
2. Las Hojas de Estilo en Cascada se usan para formatear las páginas web:
3. Las Páginas Activas se ejecutan del lado del servidor:
4. Este lenguaje es, como ASP, usado en el lado del servidor, es similar a ASP y puede ser usado en circunstancias similares:

5. Algo así como lenguaje práctico de extracción y de informes, nace con el objetivo principal de simplificar las tareas de administración de un sistema UNIX: [ ]

Comprobar