Introducción a la Arquitectura Web

Tabla de Contenidos

Introducción a la Arquitectura Web

¿Que entendemos por WWW ?

Sintéticamente podemos decir que el WWW (World Wide Web) es un sistema de Información, soportado por redes de computadoras, especialmente aquellas basadas en el protocolo TCP/IP, y basado en una arquitectura cliente/servidor. Se deben reunir, como mínimo, tres elementos para que opere: un cliente, un servidor y una plataforma de red que los conecte.

Asimismo se pude definir el WWW como un servicio de visualización de documentos basados en un lenguaje especial de marcas, el HTML, en un protocolo especial para la transferencia de los mismos, el HTTP, y en un concepto de enlaces entre diferentes documentos, el HIPERTEXTO. Sin duda alguna, este servicio es el más conocido de INTERNET y el de más repercusión social.

Como vemos para el funcionamiento correcto del servicio Web deben existir una serie de elementos combinados que en conjunto le dan su esencia: Una serie de lenguajes compresibles por computadoras (como HTML, Hypertext Markup Language) que proporcionan la codificación necesaria para el intercambio de información, un software cliente (browser) quien realiza la interfaz entre estos lenguajes y el usuario final, un software especializado en el proveedor de servicios que atienda las peticiones de los usuarios y les de curso (HTTP Server), y la propia red, basada en el conjunto de protocolos TCP/IP y especialmente un protocolo especializado (el HTTP, Hypertext Transfer Protocol) encargados estos, de realizar el transporte de la información a través de la red entre los proveedores (servidores) y los usuarios finales (clientes).

(Ver Introducción a la WWW)

Por lo tanto cuando hablamos de construir desde una simple página web a un complejo sistema que gestione información basado en tecnología web, nos encontramos siempre con el desafío de trabajar e interactuar con un mundo sumamente heterogéneo, donde para un diseñador ya no es suficiente, como en otros tiempos, el conocer un lenguaje especifico de programación o una plataforma determinada; la misma diversidad que le da a la Web su potencia, implica la necesidad de conocer varios lenguajes y plataformas para poder concretar un diseño funcional, eficaz y eficiente.

Para comenzar deberemos definir lo que entendemos por aplicación web o aplicación basada en arquitectura web.

Aplicación: Primero trataremos de definir lo que entendemos por Aplicación. Todos, antes o después, hemos escuchado la palabra *programa* en relación a una computadora o al mundo de la informática, y en general lo definimos como un conjunto de instrucciones que una computadora puede seguir para llegar a un fin determinado. Un programa puede ser muy sencillo como imprimir el mensaje *Hola mundo,* donde a la computadora simplemente se le indica que muestra esas pala-

bras por pantalla, como a algo tan complejo como resolver cálculos de física cuántica.

Todos los programas están escritos por programadores, personas especializadas en escribir estas instrucciones para la computadora, utilizando un lenguaje especial, que con su debido proceso, se convierte en instrucciones ejecutables por la dicha computadora. Cada entorno en particular posee una serie de lenguajes que son los necesarios para poder dar instrucciones a las computadoras del mismo. Por ejemplo en el entorno Windows existen VisualBasic, C#, C++, JAVA, etc. Algunos lenguajes son multiplataforma, esto es existen en diferentes entornos (como JAVA) y otros son propietarios de un entorno particular (VisualBasic con Microsoft Windows).

A un conjunto de programas, sin importar el lenguaje en que hayan sido escritos, se los puede clasificar por la función que cumplen, por ejemplo, un Sistema Operativo, un Utilitario, un programa de comunicaciones, etc. Dentro de esta clasificación, en general se define como aplicación a un conjunto de programas con un fin específico para el usuario final, por ejemplo una aplicación de contabilidad, un liquidador de sueldos, un sistema de comercio electrónico, etc.

Aplicación Web: En la definición de aplicación que dimos más arriba, no hicimos distinción en la tecnología asociada a dicha aplicación, esto es, sobre que plataforma corre, que medios de comunicación utiliza, como se comunica con el usuario, etc. Y aquí es donde entra la gran diferencia entre las Aplicaciones Web y aquellas diseñada para una plataforma monolítica.

En una aplicación monolítica, como podría ser el Microsoft Excel o el Microsoft Word, se parte de una plataforma homogénea (en ese caso una computadora tipo PC y un sistema operativo de la familia del Microsoft Windows) que es quien, en general, maneja la comunicación con el usuario (GUI interfaz gráfica de usuario) y gestiona el acceso al hardware disponible por parte de la aplicación. Esto implica que la arquitectura de dicha aplicación estará contenida en un solo entorno (el Sistema Operativo Windows en el ejemplo) y por lo tanto compartirá sus ventajas y sus limitaciones.

En el caso de una aplicación basada en tecnología web, la plataforma sobre la que se desempeñará y sobre la que tendremos que desarrollarla, es una plataforma heterogénea, con un mayor número de actores implicados, que por un lado nos brindad mayor flexibilidad, una accesibilidad mejorada y mayores posibilidades de distribución; pero por otro lado implican un desafío tecnológico mayor al tener que compatibilizar todas las tecnologías involucradas.

Una Aplicación Web entonces es una aplicación que utiliza tecnología desarrollada para la Web para desempeñarse, específicamente:

- Está distribuida geográficamente, esto es, el usuario final, no tiene por qué ser operador de la computadora que contiene la aplicación.
- Basa el intercambio de información entre la aplicación y el cliente, sea local o remota, en protocolos web (Básicamente el protocolo HTTP y derivados)

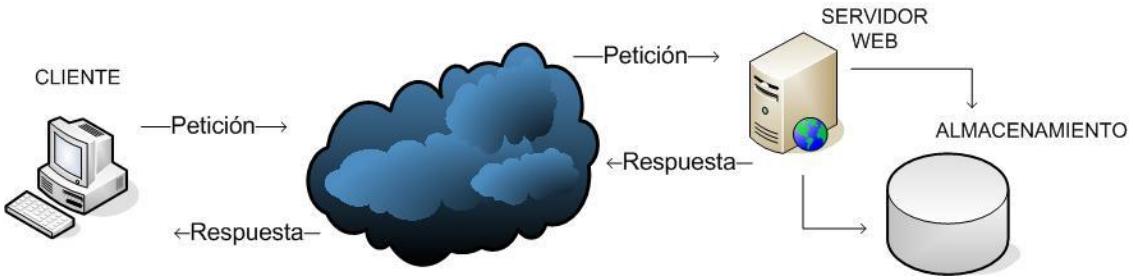
- Codifica la GUI utilizando tecnología y protocolos disponibles en la Web (HTML, XML, Plug-ins, etc.)
- Utiliza como mediador en la comunicación entre el Usuario y la Aplicación un navegador web o una aplicación específica basada en él.
- Hay una separación física entre el usuario final y la aplicación, esto es el usuario no puede acceder a la aplicación, ni a la información gestionada por ella si no es mediado por el navegador.

Entonces a diferencia de una aplicación monolítica, un aplicación web es independiente de la plataforma en la que se utilice, puesto que no hace uso de los servicios del Sistema Operativo Local; toda la comunicación entre ella y el usuario esta mediada por un software específico, esto es un navegador web.

Por otra parte la lógica de negocios, esto es la parte algorítmica de la aplicación, no residirá en el cliente, si no en un alojamiento externo, llamado servidor, accesible desde el cliente mediante facilidades de red.

Entonces, repasando, una aplicación web necesita de tres partes principales:

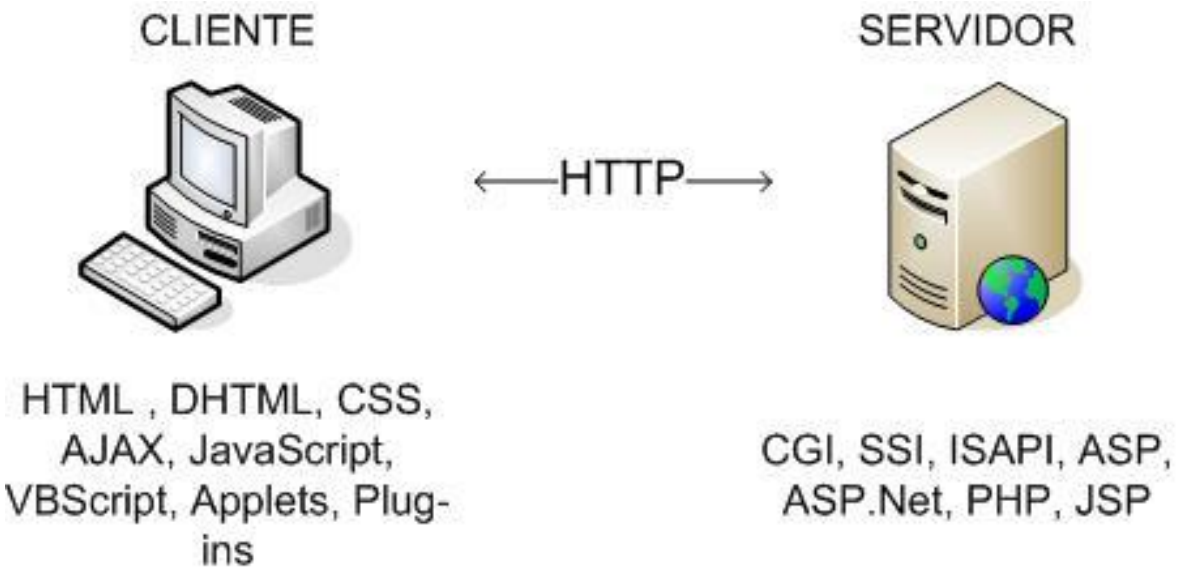
- Un servidor web, esto es un dispositivo de hardware asociado a un software específico que le permite atender peticiones de clientes, gestionadas mediante el protocolo HTTP, y en función de sus capacidades ir desde un simple envío de archivos (Pagina Web Estática), hasta la ejecución de programas complejos que impliquen accesos a bases de datos, otros sistemas, etc. (Sistema de Información basado en tecnología Web).
- Una aplicación cliente (en general un navegador web) que traduzca la información enviada por el servidor web al cliente de manera que sea comprensible por el usuario final, esto es, que resuelva la lógica de presentación. Así mismo este software debe ser capaz de interpretar las órdenes realizadas por el usuario final (por ejemplo un *click* en un vínculo) y traducirlas en peticiones comprensibles por el servidor web.
- Una red capaz de gestionar las peticiones y respuestas entre servidores y clientes, y establecer un idioma común de acceso a los recursos disponibles en la red (*DNS*, *URL*, etc).



Cada una de estas partes, necesitan que apliquemos técnicas específicas para cada una de ellas con el fin de llevar a buen puerto el desarrollo de nuestra aplicación.

En una aplicación web, ya abordada desde el punto de vista del desarrollo, siempre hay una serie de lenguajes de programación involucrados, por lo general algunos para la codificación de la interface (en el cliente) y otros para la codificación de la lógica de negocio o la algoritmia en el servidor.

Entonces en una aplicación web siempre debemos convivir con por lo menos dos mundos: uno para construir la interfaz con el cliente y otro para realizar el procesamiento en el servidor.



La interfaz con el cliente debe ser construida en un lenguaje que sea entendido por el navegador web, toda vez que es el quien muestra la información al usuario final e interpreta las acciones de este.

El lenguaje que se utiliza primariamente para esto es el HTML (HyperText Markup Language), este es un lenguaje de marcas, donde cada marca indica al navegador una acción a realizar o un elemento a mostrar en el espacio delimitado por el mismo. Este lenguaje es el principal sostén de la Web, puesto que podemos crear una Pagina Web solamente con él. Por supuesto utilizando solamente HTML se puede obtener únicamente una Web estática. El HTML se procesa completamente en el cliente (Navegador web) y por lo tanto no tiene posibilidades de acceder a ningún recurso en el servidor más que archivos direccionables por vínculos. El HTML en sí mismo no tiene capacidades algorítmicas y por lo tanto solo sirve para estructurar la información que se mostrara en el navegador y para capturar las acciones que realice el usuario sobre dicho navegador.

Aun con estas limitaciones el HTML es un lenguaje imprescindible para construir una aplicación web, puesto que es el único medio disponible para mostrar información accesible para el cliente.

El lenguaje HTML es estandarizado por el Word Wide Web Consortium (http://www.w3c.org) organismo internacional no gubernamental que regla todas

las cuestiones relacionadas con el lenguaje HTML y otras referidas al funcionamiento de la Web.

El HTML en sus inicios era el único medio de intercambiar información en la Web, a medida que la Web se fue haciendo popular, los requerimientos de interactividad e interacción se fueron haciendo cada vez más complejos, surgiendo alrededor de él una serie de tecnologías que lo potenciaron. A ellas las podemos dividir básicamente en dos tipos:

Scripting del lado del Cliente:

A medida que el funcionamiento de la Web se hizo más complejo, se empezaron a requerir capacidades algorítmicas en el cliente, por ejemplo la posibilidad de brindar interacción (presiono un botón y este cambia de forma) puesto que esperar a una petición / respuesta desde un servidor, muchas veces implicaba una demora inaceptable para el funcionamiento de la página (piense en un menú desplegable que implicara recargar la página web cada vez que se presionara una opción). Para encarar este problema, numerosas compañías desarrollaron tecnologías propias. De estas tecnologías, las que más éxito tuvieron y han perdurado hasta hoy son: Adobe (originalmente Macromedia) con Flash y ActionScript, Netscape (hoy desaparecida) con JavaScript, y Sun Microsystems con JAVA Applets. Cada una de las soluciones encara el problema desde una óptica distinta, por un lado Adobe y Sun plantean la necesidad de instalar una porción de Software adicional que entienda el lenguajes de sus productos (Plug-ins) y por el otro Nestcape, que era un fabricante de navegadores, incluyo el intérprete de su lenguaje en los propios navegadores. En la actualidad, JavaScript se ha convertido en un estándar (European Computer Manufacturers Association(ECMA-262) y está distribuido en todos los navegadores comerciales. En el caso de Flash y JAVA Applets, y a partir de la aparición de dispositivos móviles que no soportan la tecnología, poco a poco se ha empezado a dejar de lado en favor de nuevas tecnologías como HTML5/CSS3.

En el lenguaje JavaScript, la porción de código, se inserta como código fuente (esto es sin ninguna traducción a código de maquina) en la estructura HTML, funcionando e interactuando ambos como si fueran un único lenguaje, ejecutándose en el navegador web, el cual debe tener las capacidades necesarias para hacerlo. Es por esto que a esta tecnología se le llama *Scripting* del lado del cliente, *scripting* por ser un programa normalmente simple que es interpretado por el entorno, no se puede ejecutar sin el navegador que lo interprete; y del lado del cliente, porque su ejecución, como ya mencionamos, ocurre en el navegador web, no habiendo ningún contacto entre un *script* y el servidor web.

Scripting del lado del Servidor

Una vez instalada la Web como medio de intercambio de información masivo, los desarrolladores empezaron a notar una seria deficiencia en su uso, esto es, no permitía, en sus orígenes, que el cliente ejecutara ningún tipo de proceso en el servidor que le proveía de la página web, y además no se podía acceder a ningún

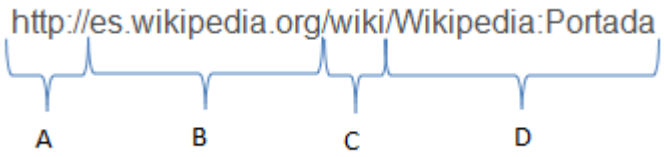
recurso en el servidor que no fuera mediado por el mismo. Estas limitaciones impedían, por ejemplo, que se consultara una base de datos corporativa a partir de una petición o parámetros decididos por el usuario puesto que el software de los servidores solo estaba desarrollado para entregar los archivos que le solicitara el cliente. A partir de esta deficiencia se realizaron varios intentos para dotar a los servidores web de la capacidad necesaria para ejecutar programas a pedido de los usuarios finales, sin vulnerar la seguridad y la consistencia de los sistemas alojados. Actualmente existe varias formas de dotar a los servidores web de capacidad de procesamiento y las más conocidas son:

- ASP / ASP.Net (Microsoft)
- PHP (Código Libre)
- JSP (Sun Microsystems)

Todas ellas parten de premisa de aislar la ejecución del programa en servidor de aquellos que corren en el usuario final.

En la mayoría de los casos, los *script* del lado del servidor, son lenguajes interpretados (por ejemplo en PHP), esto es, no necesitan traducirse a código de máquina, el proceso se realiza dinámicamente al invocarse una porción de código por parte del usuario final.

En el procesamiento normal de una página web, el usuario solicita un *url* con su navegador, en dicha *url* está indicada la dirección del servidor y el recurso que se le solicita:



A: Protocolo de transferencia (HTTP)

B: Dirección del Servidor (Nombre DNS traducible a una dirección IP)

C: Locación dentro del servidor (Carpeta wiki)

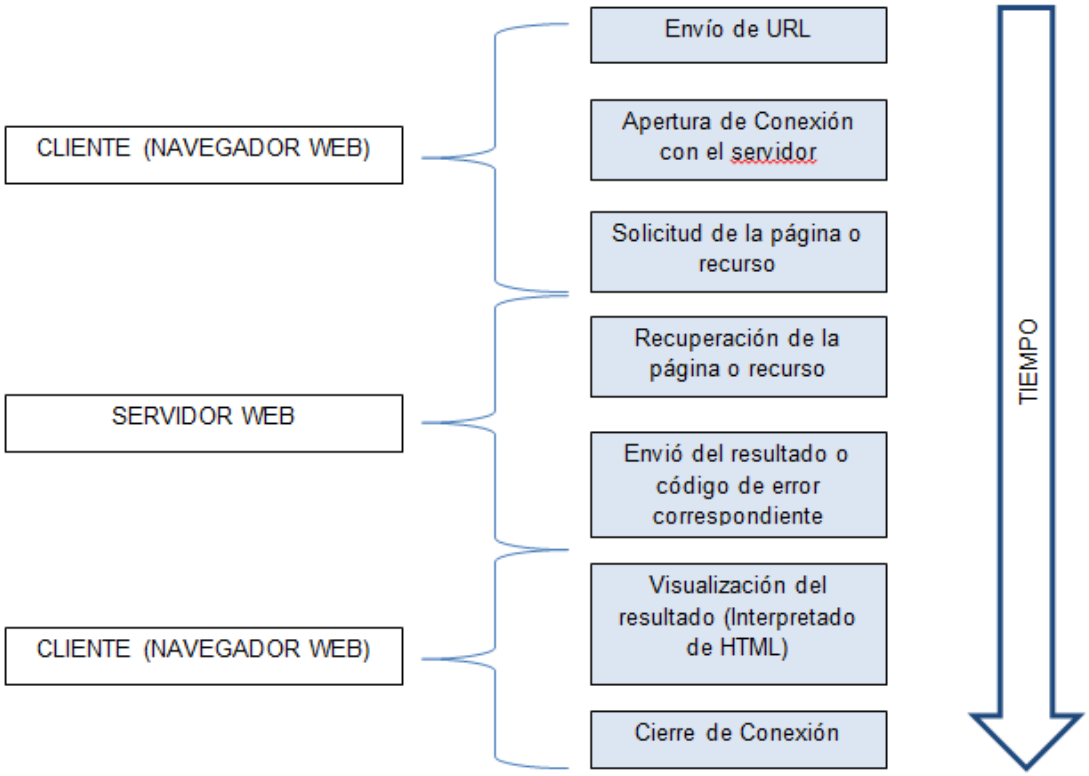
D: Recurso solicitado

El servidor, una vez recibida la petición, buscará el recurso solicitado en la locación indicada, y de encontrarlo se lo enviara al cliente, de no hallarlo generará el error correspondiente. Este proceso será así en el caso que el recurso solicitado sea un recurso estático como un archivo, una imagen, un documento de texto, etc.

Cuando la solicitud involucra un documento conteniendo un *script* ejecutable por el servidor, el procesamiento cambia. El servidor localizará el recurso solicitado, al detectar que se trata de un *script* (generalmente por la extensión del archivo, por ejemplo .php), llamara internamente al interprete que corresponda (php por ejem-

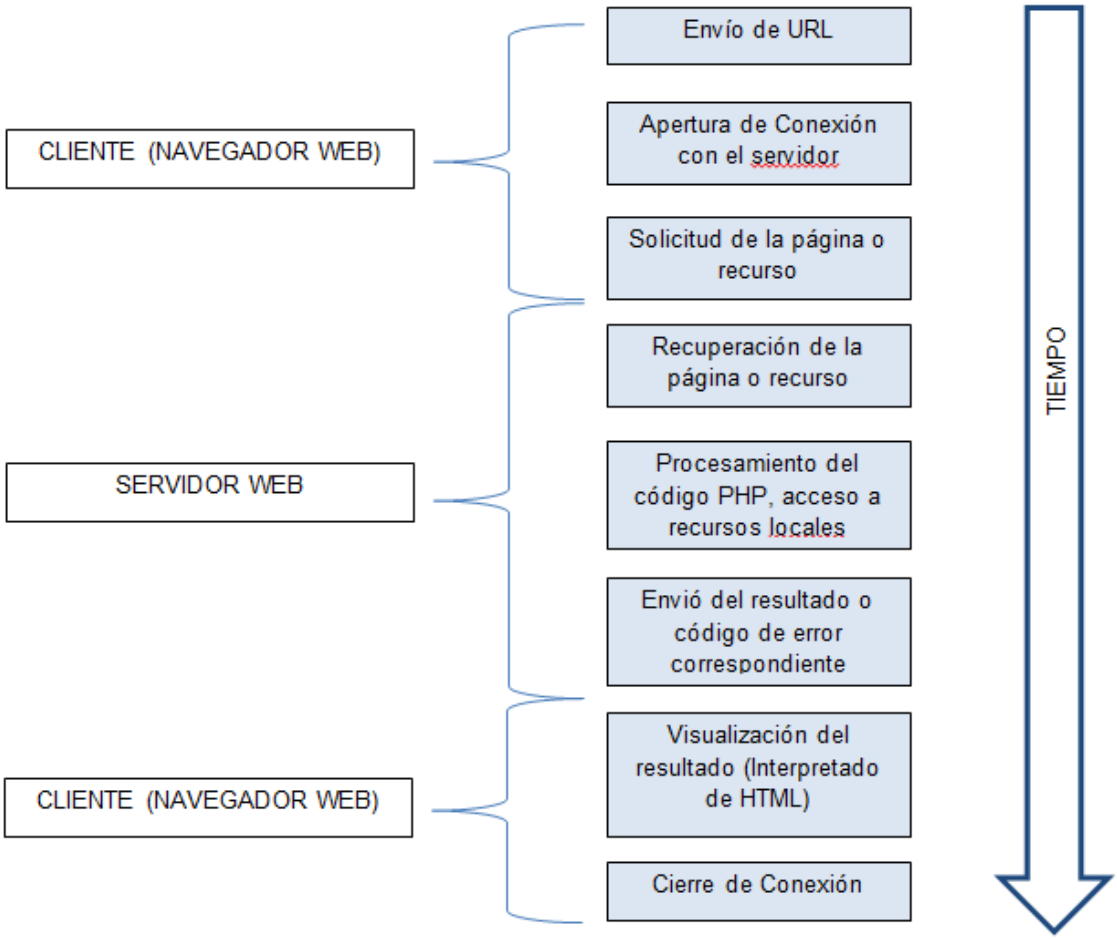
plo), ejecutará las instrucciones que contenga, siempre en el entorno del servidor, y enviara el resultado al cliente.

Por lo tanto, los script del lado del servidor nos dan la posibilidad de ejecutar programas complejos en el servidor, sin posibilidad, por parte del cliente, de afectar su funcionamiento más allá del intercambio de información realizado por HTTP. Este aislamiento entre los que se ejecuta en el servidor y lo que se ejecuta en el cliente nos brinda enormes ventajas en cuanto a la seguridad y la confidencialidad, puesto que el cliente no puede hacer en el servidor nada que yo no lo autorice y el servidor no puede ejecutar en el cliente nada que no esté dentro del entorno del navegador.



Procesamiento de una petición estática

Procesamiento de una petición dinámica (*Server Side Scripting*)



Por supuesto cuando hablamos de *scripting* del lado del servidor, estamos involucrando una porción adicional de tecnología, esto es la capacidad del servidor web, ya no de procesar peticiones HTTP, sino de brindarnos un lenguaje de programación que nos permita construir aplicaciones completas puesto que la única manera de ejecutar porciones de código en el servidor, es mediante el lenguaje que él nos provea.

La gran mayoría de los Servidores Web actualmente disponibles nos proveen de algún entorno de programación, en algunos casos más de uno. Para los entornos de programación web más conocidos, nombrados anteriormente, corresponde una implementación de servidor particular, a saber:

ASP / ASP.Net -> Internet Information Services (IIS) (Micrsosoft)

PHP -> Apache (Apache Foundation)

JSP -> Apache Tomcat (Apache Foundation + Sun Microsystems)

En todos los casos nombramos a su implementación nativa, esto es, donde fue implementado originalmente el lenguaje, existiendo portaciones de los mismos a otros entornos (por ejemplo php sobre IIS).

En el caso particular de este curso, nos centraremos en el entorno PHP montado en el servidor web Apache.

(Ver instalación de XAMPP)

Servidor Web

Como hemos dejado claro, cuando se trata de generar una aplicación de *scripting* del lado del servidor, esta porción de software llamada “Servidor”, pasa a ser un elemento crucial en el éxito de nuestro proyecto, por lo tanto vamos a estudiar un poco más su funcionamiento.

La necesidad en internet de separar el procesamiento de información en dos partes (cliente / servidor) arranca de las propias condiciones de la red que le da sustento, esto es, su objetivo principal era el brindar acceso a información remota. En sus inicios Internet era una red de muy baja velocidad (lo sigue siendo si lo comparamos con otras tecnologías de comunicación) y de poca confiabilidad, lo que hacía que el intercambio de información deba ser controlado minuciosamente para asegurar su concreción en tiempo y forma. La mejor manera de controlar que un intercambio de información sea correcto, es dotar de la suficiente “inteligencia” a remitente y destinatario, de manera de proveerles medios para controlar la secuencia y corrección de la información enviada / recibida. Par esto se construyeron porciones de *software* específicos para cada extremo de la comunicación, esto es, para el cliente el navegador, quien provee todos los mecanismos necesarios para, por un lado presentar la información de manera inteligible para el usuario final y por el otro le provee los medios para realizar las peticiones necesarias al otro extremo, el Servidor. El Servidor, no es más que una porción de software, alojado en una computadora, cuya tarea es estar escuchando a la red, a la espera de peticiones de un cliente, al recibir la petición, procesa la misma y entrega el resultado a quien se lo solicito.

En un principio los *software* de servidor eran extremadamente simples, solo podían atender a un cliente por vez y solamente podían responder a peticiones de documentos estáticos; eran simplemente despachadores de archivos. Con el progreso de internet, estos productos se fueron haciendo cada vez más complejos hasta llegar a los distintos servidores que podemos utilizar hoy, algunos de ellos capaces de atender miles de peticiones simultáneamente, balancear carga entre cientos de servidores en paralelo, y ejecutar aplicaciones complejas al nivel de un Sistema Operativo. Solo basta pensar que absolutamente todas las operaciones que realizamos en internet, ya sea leer el diario o consultar nuestro *Home Banking* están mediadas por un Servidor Web.

En nuestro caso vamos a utilizar un servidor web específico, que está construido bajo el concepto de código libre y que es el servidor web más utilizado en el mundo. Este es el servidor Apache (http://httpd.apache.org)

Siempre hay que tener en cuenta, que como ya explicamos anteriormente, Apache es simplemente una “marca” de servidor y por lo tanto existen otros servidores que

podemos utilizar para construir aplicaciones web. Decidimos utilizar Apache por diversos motivos, algunos de ellos son los siguientes:

- Es de código libre y por lo tanto podemos utilizarlo sin tener que pagar una licencia comercial.
- Es multiplataforma, esto es, existen versiones para los más variados entornos operativos.
- Implementa PHP en forma nativa. No debemos realizar ninguna configuración especial para que corra PHP (por lo menos en las distribuciones que vamos a utilizar).
- Es parte de la mayoría de las distribuciones de Linux.
- Tiene la mayor distribución mundial dentro de los Servidores de HTTP

Developer	September 2013	Percent	October 2013	Percent	Change
Apache	99,354,736	52.30%	98,646,225	52.14%	-0.16
nginx	24,426,727	12.86%	24,255,815	12.82%	-0.04
Google	22,527,229	11.86%	22,146,345	11.71%	-0.15
Microsoft	20,177,662	10.62%	20,119,180	10.64%	0.01

Fuente: http://www.netcraft.com/

Entonces, Apache es un servidor web HTTP de código abierto, multiplataforma, que implementa el protocolo HTTP/1.12 y la noción de sitio virtual. Su desarrollo comenzó en 1995 se basó inicialmente en código de un servidor preexistente, el NCSA HTTPd 1.3, pero más tarde fue reescrito por completo.

Actualmente es mantenido por una Fundación (http://www.apache.org), la *Apache Software Foundation*, que además del servidor HTTPD propiamente dicho, provee otros productos de software siempre destinados a su utilización en la web.

Como todo servidor Web, Apache implementa en el equipo servidor el protocolo HTTP (http://www.w3.org/Protocols/), protocolo de la familia del TCP/IP sobre el que está basado todo el funcionamiento de la web.

El Apache es el encargado de atender las peticiones que los clientes realizan a través de su navegador y la red, y devolver los documentos HTML, imágenes, porciones de código, etc. que sean requeridos.

La componente técnica de la comunicación entre el servidor web y el cliente este proceso esta explicada en el documento *Introducción a la Wold Wide Web* y por lo tanto no abundaremos en ella. Lo que si nos interesa es ver como el Apache or-

ganiza los recursos del sitio web de manera de poder entregarlos a los clientes que lo requieran.

Primero, el Apache es un servidor que soporta el concepto de sitio virtual, esto es, puede alojar numerosas páginas web independientemente unas de otras. Por ejemplo podemos tener en un mismo servidor físico (una PC para simplificar) corriendo un Apache, la página del Ministerio de Planificación, otra del Restaurant de la esquina y un sitio de ventas on-line. Para ello tenemos que comprender el concepto de sitio para el Apache.

Para el servidor Apache el sitio web es un conjunto de carpetas dentro del servidor, definidas desde una raíz virtual, a partir de la cual los documentos pertenecientes a un sitio determinado pueden ser accedidos por el cliente que los requiera. Por ejemplo si tenemos alojado en nuestro servidor el sitio http://www.comidasenlinea.com.ar , para el Apache se definirá una carpeta, con los permisos de seguridad correspondientes, a partir de la cual existirá dicho sitio.

Ej:

C:\xampp\htdocs\comidasenlinea\

Si un cliente nos requiere cualquier documento para el dominio www.comidasenlinea.com.ar, dichos documentos residirán a partir de la carpeta indicada en el punto anterior, que para el navegador cliente se convertirá en la carpeta raíz (/). Cualquier documento o carpeta a la que se quiera acceder a partir del domino que indicamos, deberá residir en la carpeta física indicada o por debajo de ella.

Recurso	Carpeta Física
www.comidasenlinea.com.ar/index.html	c:\xampp\htdocs\comidasenlinea\index.html
www.comidasenlinea.com.ar/imagenes/logo.png	c:\xampp\htdocs\comidasenlinea\imagenes\logo.png

Por supuesto podríamos tener otro sitio definido en otra carpeta física y los documentos nunca se mezclarán.

¿Cómo hace Apache para saber a cuál sitio corresponde una petición determinada? Normalmente de dos maneras, o bien cada sitio responde a un dirección IP diferente, por lo tanto www.pirulo.com.ar se resuelve en un IP diferente que www.comidasenlinea.com.ar y el Apache atiende ambas IP (por supuesto el equipo anfitrión, esto es, el equipo que corre el Apache, debe escuchar ambas IP); o se utiliza lo que se llama *hosting* por nombre. Para ello hay que saber que en una petición http, el cliente incluye en la petición el nombre del *host* que quiere consultar, en el ejemplo que estamos dando www.comidasenlinea.com.ar. El Apache al recibir la petición, identifica el nombre del *host* que se está solicitando y redirige las peticiones a la carpeta física asociada a ese sitio.

Entonces, una vez teniendo en funcionamiento nuestro Apache (ver el documento *instalación de XAMPP*) lo que tenemos que hacer para poder tener un sitio en funcionamiento, es simplemente poner en la carpeta por defecto del XAMPP (c:\xampp\htdocs) los documentos HTML o PHP que queremos visualizar, y realizar las peticiones correspondientes mediante un navegador.

La configuración del Apache para que soporte múltiples sitios o para cambiar las carpetas por defecto se realizan en el archivo de configuración que en el caso del XAMPP es c:\xampp\apache\conf\httpd.conf. El alcance de este curso no abarca la descripción de las instrucciones de configuración del Apache y por lo tanto usaremos la configuración estándar que implica un solo sitio disponible localizado en c:\xampp\htdocs.

Referencias

World Wide Web Consortium. http://www.w3c.org

W3Schools. http://www.w3schools.com

Apache Software Foundation. http://www.apache.org