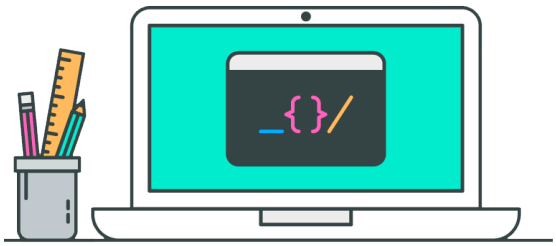
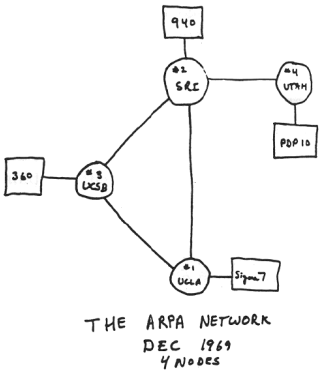


Tema 1
Desarrollo Web

Aunque los primeros avances en el estudio de redes de computadoras se dieron en los EEUU, el elemento detonador de todo el proceso es necesario buscarlo en otro país, la URSS. Tras el lanzamiento del primer satélite artificial, Sputnik, por parte la URSS, en los EEUU se inició un período de crisis. Se sentían derrotados en la guerra fría y necesitaban una revisión de las políticas de desarrollo científico y tecnológico que se habían realizado hasta entonces. En este marco, surge en 1975 la agencia **DARPA** (Defense Advance Research Projects Agency), con unos objetivos claramente vinculados con el desarrollo tecnológico aplicado a la defensa, pues se consideraba altamente peligroso que la URSS fuese por delante en las distintas carreras emprendidas en la guerra fría.

A pesar de habernos remontado a finales de los años cincuenta, no se producen verdaderos avances hasta comienzos de la década de los 60, aunque dichos avances se centran en aspectos más conceptuales que tecnológicos. Así, en 1962, J.C.R. Licklider (psicólogo e informático de la agencia DARPA) propuso la interconexión de ordenadores para el desarrollo de trabajo colaborativo entre investigadores. De esta manera, la agencia DARPA inició un programa de investigación de técnicas y tecnologías para unir diversas redes de conmutación de paquetes, permitiendo así a los ordenadores conectados a estas redes comunicarse entre sí de forma fácil y transparente.

De esta manera, el 21 de noviembre de 1969 nació **ARPANET**. Considerada la primera red de ordenadores, ARPANET conectaba inicialmente la UCLA (Universidad de California en Los Ángeles) y el Instituto de Investigaciones de Stanford, añadiéndose posteriormente dos nodos más, en la Universidad de Utah y en la UCSB (Universidad de California en Santa Bárbara). Oficialmente se inauguró oficialmente en 1972 y creció tan rápidamente que, en 1981 ya contaba con 213 nodos.



Viendo el éxito alcanzado, los ingenieros de la DARPA comenzaron a investigar sobre la posibilidad de interconectar diferentes tipos de redes. Así pues, ARPANET se adaptó a un nuevo protocolo de comunicación, nacido en 1983 de la mano de Robert Kahn y Vinton Cerf, y bautizado como **TCP/IP** (Transmission Control Protocol/Internet Protocol). Realmente, el éxito de Internet, que nació años después sobre **NSFNET** (National Science Foundation's Network), reside en el empleo de este conjunto de protocolos de comunicación que solucionan el problema de heterogeneidad de los sistemas informáticos.

Pero fue en 1989, cuando Tim Berners-Lee, trabajando para el **CERN** (European Organization for Nuclear Research / Organisation Européenne pour la Recherche Nucléaire), crea la World Wide Web que, junto con el correo electrónico, se ha convertido en uno de los principales pilares de la sociedad moderna. A partir de este momento, la Web comenzó a extenderse de tal manera que, en la actualidad para la mayoría de nosotros es indistinguible qué es la web y qué es Internet. Desde la web es posible realizar la mayoría de tareas que un usuario desea de Internet, aunque esto no ha sido así siempre.

Inicialmente Internet no tenia web: tenía servicios como el correo electrónico, transmisión de ficheros, grupos de noticias, etc. Todos estos servicios eran utilizados a través de interfaces poco amigables (para leer el correo electrónico hacía falta un cliente de correo, la transmisión de ficheros se realizaba mediante un cliente FTP, los foros de debate se leían a través de gestores de news, etc.), de manera que los profesionales de la informática eran los únicos que se atrevían a hacer uso de ellos. Aún pueden utilizarse estas aplicaciones para trabajar en Internet, pero lo cierto es que actualmente, casi todo el mundo realiza las tareas en Internet a través de un único servicio: la web. Pero, ¿por qué…?

a. **Fácil uso**. Clics de ratón nos llevan de un punto a otro y eso es muy fácil de entender.
b. **Visualmente atractivo**. La web fue el primer servicio que permitió mostrar la información de Internet acompañada de imágenes, sonidos y vídeos.
c. **Uso de una única aplicación**: el navegador. Aplicación presente en todos los ordenadores, lo que hace que, siempre que deemos acudir a cualquier servicio, no tengamos que abrir diferentes programas, sino que utilicemos el que usamos habitualmente para acceder a cualquier otro servicio.

En sus inicios, la web estaba constituida por páginas compuestas pácticamente en su totalidad de texto y enlaces, pero en poco tiempo, los usuarios comenzaron a demandar más servicios, como la inclusión de imágenes, vídeos, animaciones y, más adelante el acceso a servidores de bases de datos, correo electrónico, transmisión de ficheros, tiendas online, etc. En definitiva, como se dijo anteriormente, hoy en día podemos realizar cualquier tarea desde la web, como editar documentos, retocar imágenes, ver películas, escuchar música, etc. Tan solo el mundo de las apps en dispositivos móviles, parece arrojar una, cada vez más alargada sombra, sobre la web.

1. La web 2.0

Como hemos visto anteriormente, el auge de Internet se debe a la facilidad con la que podemos acceder a la información; textos, gráficos, datos y enlaces a diferentes documentos, que pueden estar almacenados en diferentes ordenadores, creando una **telaraña global**, que conocemos como **World Wide Web**.

En sus comienzos, la web estaba constituida por **páginas estáticas** que, para poder reflejar una actualización requerían de los servicios de programadores. Simples documentos HTML, muy sencillos de crear, pero que no iban más allá de textos planos, acompañados de imágenes y de determinados contenidos multimedia.

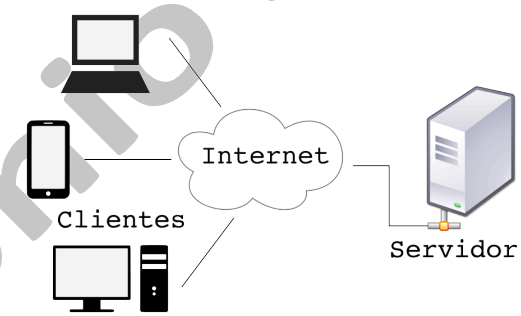
Esta **web 1.0** se vio rápidamente inundada de contenido, y el creciente éxito de Internet dependía cada vez más de webs con contenidos dinámicos, cuya información era generada en tiempo real, utilizando servicios de bases de datos. Esta forma de funcionamiento, junto con la estética visual, fue promoviendo un mayor uso de la web que, poco a poco ha ido desembocando en lo que hoy en día conocemos como **web 2.0** (término acuñado por Tim O'Reilly), un modelo que nace de la necesidad de actualizar la manera en que los desarrolladores y usuarios finales utilizan la web, y donde la interactividad e interconexión entre *sites* es un pilar básico.

Aunque en la actualidad ambos modelos de web conviven en esa gran telaraña mundial, la **web 2.0** abarca aquellas páginas que ofrecen servicios orientados al usuario, de manera que le permitan un manejo más rico e incluso le hagan partícipe del contenido.

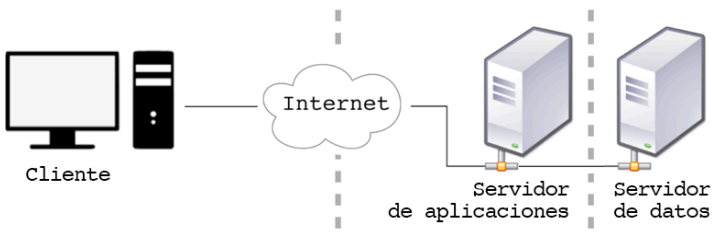
El principal objetivo de esta asignatura es el de diseñar y desarrollar aplicaciones Web, alejándonos sustancialmente del concepto tradicional de Web. Actualmente se tiende a diseñar aplicaciones que permitan la interacción directa entre el usuario, e inclusive con otras aplicaciones.

2. Modelo Cliente-Servidor

El concepto de **cliente-servidor** se define como un modelo de comunicación que vincula diferentes dispositivos informáticos a través de una red. En este sentido, el **cliente** realiza una petición de servicios, y el **servidor** se encargará de procesar dicha petición y proporcionar una respuesta adecuada.



Esta arquitectura permite distribuir las tareas entre servidores y clientes, repartiendo de esta manera la capacidad de procesamiento. Tradicionalmente se ha empleado una **arquitectura en 2 niveles**, para describir un sistema cliente-servidor en el que el servidor responde directamente a la solicitud con sus propios recursos. Este modelo se amplía con una capa intermedia, donde encontraremos el **servidor de aplicaciones**, y finalmente la capa de **servicios**.



De esta manera, el **navegador** (capa 1) implementará la lógica de presentación, el **servidor de aplicaciones** (capa 2) gestiona la solicitud, toma decisiones, moviendo y procesando datos entre las otras capas. Finalmente, el **servidor de datos** proporciona acceso a servicios dedicados, tales como un servidor de bases de datos, de archivos, multimedia, etc. De esta manera, la funcionalidad de una arquitectura cliente-servidor se organiza en las capas:

a. **Capa de presentación**: cuyo objetivo es recibir las peticiones HTPP y devolver la respuesta adecuada en un formato inteligible por el navegadores. Esta tarea recae normalmente en el cliente.

b. **Capa de negocio**: gestiona y envía al cliente la información resultante del procesamiento de dicha información por parte del servidor, con objeto de resolver un determinado problema, de forma totalmente transparente al usuario. Esta capa, lógicamente, la encontramos del lado del servidor.

c. **Capa lógica**: esta capa equivale a la que anteriormente ocupaba el servidor de aplicaciones, que será invocado por el servidor web cuando a éste le llega la petición de un recurso que debe de ser tratado en el lado del servidor. Por ejemplo, cuando al servidor le llega una determinada petición que requiera la interpretación de código PHP, la capa lógica será la encargada de realizar dicha interpretación y enviar al servidor web el resultado de dicha interpretación, y que será inteligible por el navegador.

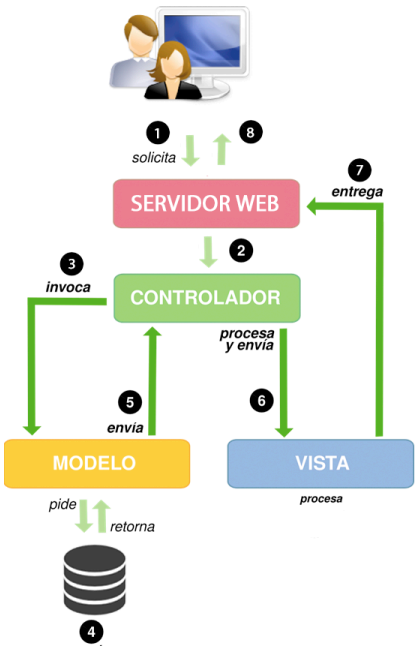
Aunque la arquitectura en 2 capas supone un modelo polivalente, en la arquitectura de 3 capas las aplicaciones al nivel del servidor se encuentran descentralizadas, permitiendo una mayor flexibilidad, seguridad y un mejor rendimiento. Actualmente, suele ser habitual que ser el servidor de aplicaciones y datos sea el mismo.

En la última década se ha impuesto el paradigma **modelo-vista-controlador** (MVC) que consta de tres componentes básicos.

a. **Modelo**: supone la representación de la información, por lo tanto gestiona todos los accesos a dicha información, tanto consultas como actualizaciones, etc. Para ello, envía a la **vista** aquella parte de la información que se le solicita en cada momento para que sea mostrada (típicamente a un usuario). Las peticiones de acceso o manipulación de información llegan al **modelo** a través del **controlador**.

b. **Vista**: presenta la información del **modelo** en un formato adecuado para interactuar, esto es, la interfaz de usuario.

c. **Controlador**: responde a eventos (usualmente acciones del usuario) y realiza solicitudes al **modelo** cuando se hace alguna petición de información (por ejemplo, editar un documento o un registro en una



base de datos). Se podría decir, por tanto, que el **controlador** hace de intermediario entre la **vista** y el **modelo**.

Si bien es cierto que este modelo es similar a la arquitectura en 3 capas, existen ciertas diferencias. Este último es lineal, esto es, no existe una comunicación directa entre las diferentes capas. No ocurre igual en el MVC, donde existe una relación entre las diferentes capas.

3. Tecnologías de la Web 2.0

Hoy en día, los desarrolladores combinan las tecnológias del lado del cliente y del servidor, para obtener aplicaciones web muy potentes y permitiendo un alto grado de interactividad con el usuario final, lo que nos permite introducir a éste en una web social que lo hace partícipe de lo ocurre, interactuando con otros usuarios, participando, creando u opinando sobre los contenidos.

En definitiva, la web 2.0 se basa en una arquitectura cliente-servidor tradicional, donde los servicios web requieren por lo general de un soporte de bases de datos y un flujo de trabajo mucho más consistente. Generalmente, los desarrolladores optan por un enfoque de servidor universal, es decir, los servicios y la mayor parte de la funcionalidad necesaria se encontrará bajo un mismo servidor. El contenido no es sólo creado por los propietarios de las aplicaciones, sino que se hace partícipe a los usuarios de ello, facilitando controles e interfaces más manejables, intuitivas y atractivas. Los servicios ofrecidos por las distintas páginas webs interaccionan entre sí para ofrecer al usuario una experiencia de última generación, en la que el contenido no está centralizado, sino que se distribuye en una maraña de servidores a la que conocemos genéricamente como nube. Asociado a todo esto, surgen diferentes tecnologías (algunas ya conocidas) que se enlazan de forma directa con el modelo 2.0.

a. **XML (eXtensible Markup Language)**. Metalenguaje definido por el W3C (World Wide Consortium), que simplifica y adapta el lenguaje de marcado SGML, permitiéndonos definir la estructura de la gramática de un lenguaje específico. Esto es, XML no es en sí un lenguaje, sino una manera de definir lenguajes que, posteriormente, podrán ser utilizados con diferentes finalidades; generalmente para el intercambio de información entre diferentes sistemas.

b. **XHTML (eXtensible Hypertext Markup Language)**. El lenguaje extensible de marcas de hipertexto es una versión XML más avanzada del lenguaje HTML, que se utiliza para la creación y visualización de páginas web.

c. **CSS (Cascading Style Sheets)**. Las hojas de estilo en cascada, estudiadas en el curso anterior, no es más que un lenguaje que nos permite definir la estética de la interfaz de una página web, de manera que el aspecto quede separado del contenido en sí. Esto facilita la continua mejora, ó actualización de la interfaz, así como la definición de diferentes entornos dependiendo del usuario final.

d. **AJAX (Asynchronous JavaScript and XML)**. Se conoce por este nombre a una técnica de desarrollo web que permite modificar la información de una página web sin tener que recargarla completamente, agilizando de esta manera la interacción con el usuario. En contraposición al modelo tradicional de comunicación con el servidor, donde éste espera las solicitudes del cliente para responderle de forma apropiada, la tecnología AJAX funciona de forma asíncrona. De esta manera, los datos viajan el cliente y servidor, sin que el usuario intervenga en ningún momento.

e. **RSS (Really Simple Syndication)**. Formato estándar para la sindicación (redifusión) de contenidos, a los que cualquier usuario puede suscribirse mediante una aplicación agregadora de feeds.

f. **Mashup**. Aplicación web que utiliza información de diversas fuentes relevantes (normalmente empleando servicios web) para crear un nuevo servicio en base a ellas.

g. **Widget**. Aplicación o módulo que realiza una función concreta, generalmente de tipo visual, dentro de otras aplicaciones o sistemas operativos.

En estos términos surgen las **aplicaciones ricas de Internet** (**RIA**, Rich Internet Application), que ofrecen servicios que se asemejan con las aplicaciones de escritorio, tales como Google Drive, Office Online, etc. Años atrás, las aplicaciones eran creadas para ser ejecutadas de forma local y aprovechar al máximo los recursos de nuestras máquinas. La web y los navegadores han evolucionado de tal manera que, hoy en día permiten la ejecución de aplicaciones web complejas que ofrecen un sinfín de posibilidades, siendo éstas capaces de competir con aplicaciones que hasta ahora tan sólo podíamos instalar y ejecutar de forma local en nuestros equipos.

Ligado a todo esto nace la **computación en la nube** como respuesta tecnológica a los retos impuestos a empresas como Google empresas o Amazon, que deben responder rápidamente a las peticiones de millones de usuario. Su funcionamiento se basa en la distribución de aplicaciones entre diferentes servidores, permitiendo responder a una importante demanda de peticiones de servicio y, además poseer una alta capacidad de tolerancia a fallos. Al igual que la web 2.0, con la que está íntimamente relacionada, el cloud computing permite utilizar el software como un servicio (**SaaS**, Software as a Service), esto es, al usuario le basta con un navegador o una pequeña aplicación para acceder y utilizar el servicio. El concepto de computación en la nube sustituye al propio ordenador personal, delegando en la nube el proceso de las tareas y el almacenamiento de la información.

4. Tecnologías para el desarrollo de aplicaciones Web

A la hora de desarrollar e implantar una aplicación web, entran en juego numerosos factores que han de tenerse en cuenta. Como se estudió en el tema anterior, deberemos dedicar una parte importante de la especificación de nuestro proyecto, a justificar aquellos elementos de índole tecnológico que hemos elegido, independientemente de aquellos aspectos relacionados directamente con la funcionalidad de la web.

4.1. Servidores web

Aunque siempre que hablamos de servidores hacemos referencia a un dispositivo físico, podemos extender el concepto de servidor, definiéndolo como una aplicación software que realiza tareas en beneficio de otras aplicaciones llamadas clientes, como por ejemplo, servicios de almacenamiento de archivos, de impresión, de correo, servicio web, etc. Precisamente, los **servidores web** tienen como objetivo el de recibir y servir las peticiones de páginas o elementos web almacenados en el servidor.

En realidad en ocasiones es indistinguible un servidor web de un servidor de aplicaciones. La diferencia radica en que un servidores web simplemente atiende peticiones HTTP y los de aplicaciones interpretan código escrito en algún lenguaje de desarrollo web. Lo cierto es que casi todos los servidores web actuales permiten actuar de servidores de aplicaciones gracias a la posibilidad de añadir componentes capaces de ejecutar tecnologías del lado del servidor.

a. **Apache**. Es indudablemente el servidor web más popular de la actualidad. Se trata de una aplicación de software de código abierto que utiliza una licencia de tipo Apache License, variante de la licencia GPL de Linux, lo cual significa que se puede distribuir libremente e incluso modificar el código siempre y cuando el resultado mantenga la licencia original. Dispone de multitud de módulos (SSL, Perl, Python, PHP, etc.) que convierten a Apache en un servidor capaz de gestionar todo tipo de aplicaciones, lo que también le convierte en el servidor de aplicaciones más popular de la actualidad.

b. **IIS (Internet Information Server)**. Servidor de aplicaciones de Microsoft que está presente en las versiones profesionales de Windows y en todas las de servidor. Viene con el propio sistema operativo y para instalarle basta con agregarlo como componente del sistema. Incluye un servidor web (http/https), servidor ftp, webDAV (Creación y control de versiones distribuidas por web) y smtp. Además se comporta como servidor de aplicaciones web .NET y admite extensiones para diversos tipos de aplicaciones (incluido PHP).

c. **Nginx**. Muy ligero (con pocas funcionalidades) pero presume de su alta velocidad especialmente con un número muy alto de sesiones concurrentes. Tiene licencia de uso BSD, que permite su distribución y modificación libre. Pueden añadírsele numerosas funcionalidades, entre ellas la capacidad de servir PHP, Ruby on Rails, Java, etc.

d. **Lighthttpd**. Servidor web que presume de muy poco consumo de CPU por su ligereza, pero de una alta velocidad y que, mediante FastCGI (protocolo de interconexión de aplicaciones con un servidor web) permite servir muchos lenguajes de servidor, especialmente Ruby on Rails y PHP.

e. **Apache Tomcat**. Creado por la fundación Apache es un servidor web escrito en Java capaz de interpretar servlets (lenguaje de programación Java, utilizado para ampliar las capacidades de un servidor) y páginas JSP escritas en Java.

f. **Webrick**. Servidor de aplicaciones popular para pruebas, debido a su ligereza, aunque no es lo suficientemente potente para usarlo a nivel comercial. Es el utilizado generalmente por desarrolladores de Ruby On Rails.

4.2. Sistemas gestores de bases de datos

Otro de los elementos dinamizadores de la web son las bases de datos. Las aplicaciones que nos permiten crear una base da datos y almacenar información para, posteriormente poder gestionarla de forma dinámica, se conocen con el nombre de **Sistemas Gestores de Bases de Datos** (SGBD). Estos complejos sistemas software, persiguen diferentes objetivos: **abstracción de la información**, **independencia de los dat**os, **consistencia** y **seguridad**.

Los sistemas de gestión de la información se han convertido en parte fundamental de la web, siendo tan importante el valor de la información actualizada, que el gestionarla eficientemente supone una baza en favor de los beneficios. Dado que disponemos de diferentes opciones a la hora de elegir un SGBD resulta imprescindible contar con elementos de juicio a la hora de optar por una u otra solución.

El principal objetivo reside en encontrar el SGBD que sea capaz de responder de forma adecuada a nuestras necesidades. De esta manera, hemos de centrarnos en primer lugar en diferentes características que definen a un gestor de bases de datos: **modelo de datos** (relacional, jerárquico, en red, etc.), **lenguajes de definición y manipulación**, y las **herramientas de ayuda para el desarrollo** (Toad, SQL Server, SQLite, phpMyAdmin, etc.).

En segundo lugar, hemos de evaluar el rendimiento del SGBD, para lo cual podemos basarnos en bancos de pruebas realizados sobre diferentes gestores de bases de datos. Estos bancos de pruebas están comprendidos por diferentes tests que miden la eficiencia de cada SGBD a la hora de realizar determinadas tareas, tales como: **capacidad de almacenamiento** y **recuperación de datos**, **protección de datos**, **control de accesos**, **consumo de recursos**, etc.

En base a los resultados obtenidos podremos hacer nuestra elección. Usado en muchos de los sitios web más grandes y populares de hoy en día, tales como Wikipedia, Facebook, Twitter, Flickr, Youtube, etc., **MySQL** es uno de los motores de bases de datos más extendidos actualmente. Este motor es una solución sencilla y de libre acceso que se ofrece bajo licencia GNU GPL, aunque será necesario adquirir una licencia de uso comercial para entornos empresariales. Al ser multiplataforma, y dada su versatilidad, encontraremos muy ligado a **MySQL** con PHP como base de numerosos gestores de contenidos web como Drupal, Joomla, Moodle, etc. Su popularidad va ligada al uso intensivo en la lectura de datos (utilizando el motor **MyISAM** – mecanismo de almacenamiento de datos), en contraposición con los problemas de integridad que pueden ocasionarse en entornos de alta concurrencia para la modificación de datos. Por lo tanto, aunque no es aconsejable su uso con grandes bases datos, **MySQL** es un motor ideal para aplicaciones web que precisan de un entorno intensivo de lectura y presentan una baja concurrencia en la modificación de información.

PostgreSQL es otra de las soluciones de libre distribución más extendidas en la actualidad y usada en aplicaciones web como IMDb, Skype, TiVo, Sony, VeriSign, etc. Competidor directo de MySQL, PostgreSQL, entre otras muchas interesantes características, ofrece un sistema denominado **MVCC** (MultiVersion Concurrent Commit) que permite a un proceso, sin necesidad de bloqueos, modificar los datos de una tabla mientras otros acceden a la misma tabla. Sin embargo, aunque es una muy buena solución, ya que soporta mucha más carga que su competidor, implementando el uso de rollbacks, subconsultas y transacciones, consume una gran cantidad de recursos convirtiendo a postgreSQL en un producto mucho más lento que MySQL.

Otra distribución con licencia GPL que parece extenderse entre diferentes entornos de desarrollo, nace de la mano de Michael Wildenious, uno de los fundadores de MySQL y recibe el nombre **MariaDB**. Este motor no es más que un fork directo de MySQL que cuenta con las mismas órdenes, interfaces y bibliotecas que éste, con el único objetivo de mantener una alta compatibilidad con MySQL. Entre sus características más destacadas señalamos la inclusión de dos nuevos motores de almacenamiento: **Aria** (que reemplaza con ciertas ventajas a MyISAM) y **XtraDB** (que sustituye a InnoDB).

Finalmente, entre las soluciones comerciales, encontramos **dBase**, **Fox Pro**, **IBM DB2**, **Microsoft Access**, **Microsoft SQL Server**, **Paradox**, **Sybase** y **Oracle**, siendo este último uno de los mejores y más completos gestores de bases de datos del mercado, ya que ofrece soporte multiplataforma, permite transacciones, presenta una gran estabilidad y es altamente escalable

4.3. Lenguajes orientados a la web

Como ya hemos dicho anteriormente, nuestro objetivo en esta asignatura será el de diseñar, desarrollar, instalar y mantener aplicaciones web complejas, con lo que nos alejamos sustancialmente del concepto tradicional, y algo obsoleto, de web. Como hemos visto anteriormente, en la actualidad se tiende a diseñar aplicaciones que permitan la interacción directa entre el usuario e inclusive con otras aplicaciones. En este sentido, dejaremos de lado las **páginas estáticas**, que no son más que simples documentos HTML, muy sencillos de crear, pero que no van más allá de textos planos, acompañados de imágenes y de determinados contenidos multimedia.

A la hora de desarrollar **aplicaciones web dinámicas** debemos diferenciar entre los diferentes lenguajes que podemos utilizar. Unos se ejecutan del lado del cliente (HTML, JavaScript, CSS, Java, etc.) y nos proporcionarán características algo limitadas ya que, en ocasiones, los navegadores deberán contar con un conjunto de capacidades añadidas. En este sentido, los lenguajes **del lado del servidor** cuentan con un gran éxito, ya que extienden su funcionalidad del lado del servidor (PHP, CGI, Perl, Python, JSP, etc.), no solo liberando al navegador de la carga de ejecución, sino facilitando además un amplio abanico de funciones que permitirán al desarrollador crear aplicaciones que se ajusten mucho más al concepto de web dinámica, aunque en realidad, generalmente las aplicaciones web dinámicas hacen un uso combinado de numerosas tecnologías y lenguajes.

En los próximos temas, centraremos nuestro estudio en **PHP** (Hipertext Pre Processor) es uno de los lenguajes del lado del servidor más extendidos. PHP es un lenguaje de script, esto es,

su código se incrusta dentro de código escrito en otro lenguaje. Gratuito y de código abierto, guarda una excelente relación con Apache, MariaDB, MySQL sobre Windows y Linux, aunque también puede instalarse en servidores como IIS de Microsoft y usar otros motores de bases de datos como Oracle, Informix, DB2, etc. PHP se considera uno de los lenguajes más flexibles, potentes y de alto rendimiento conocidos hasta el día de hoy, lo que ha llevado a sitios como Facebook a optar por este lenguaje como tecnología de servidor.

5. Paquetes completos

A la hora de desarrollar aplicaciones web mediante software de código abierto y gratuito, en seguida pensamos en un conjunto de herramientas que dependiendo del sistema operativo sobre el que se ejecutan reciben un nombre u otro: Microsoft Windows (**WAMP**), Mac OS (**MAMP**), Linux (**LAMP**), etc. Sea como sea, el conjunto de estas soluciones incluyen un servidor web (**Apache**), un motor de bases de datos (**MariaDB**) y diferentes lenguajes de programación (**PHP**, **Perl** ó **Python** – dependiendo de la distribución). Los aspectos que hacen tan interesante el uso de estas herramientas para el desarrollo y la publicación de información en Internet son:

a. El **ahorro** de costes es notable, ya que las herramientas contenidas en estos paquetes son gratuitas.

b. La **independencia** de la plataforma posibilita instalar los servicios en casi cualquier tipo de arquitectura, sin tener que usar dispositivos o sistemas operativos de un fabricante en particular. En todo caso, el acceso al código fuente de las aplicaciones permite compilar el código según las necesidades del servidor.

c. La **versatilidad** de las herramientas facilita la creación de configuraciones a medida para cada sistema, ya que, tanto Apache, como MySQL, PHP, Perl y Python, son escalables, permitiendo la ampliación mediante módulos que complementan o añaden beneficios al servicio.

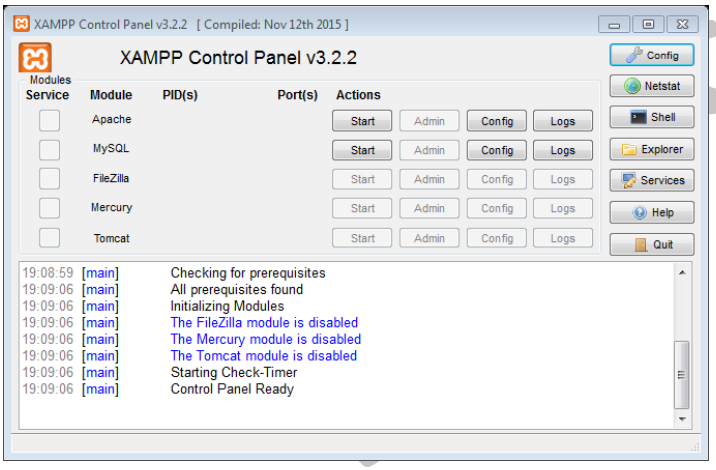
d. La estructura de estas herramientas las hace idóneas para la creación de gestores de contenidos, y servidores de aplicaciones. En su contra podemos destacar que el sistema empieza a tener ciertos problemas cuando las bases de datos son excesivamente grandes; en este caso, deberá optarse por otras soluciones.

En esta asignatura utilizaremos la distribución **XAMPP** (**X** – cualquier sistema operativo -, **A**pache, **M**ariaDB, **P**HP y **P**erl), un paquete multiplataforma disponible para diferentes sistemas operativos, entre ellos Windows y Linux, y que, aunque inicialmente fue pensado como una simple herramienta de desarrollo, se ha convertido en una de las soluciones más utilizada para servidores web.

Accediendo a la remozada página del paquete XAMPP (www.apachefriends.org) podremos descargar la última versión (**v7.1.9** en la fecha de elaboración de estos apuntes) para la plataforma deseada. De esta manera, elegimos el sistema operativo deseado (Windows o Linux) y descargamos la distribución correspondiente.

5.1. Instalación en Windows

Descargado el paquete **XAMPP** a nuestra máquina, ejecutamos el archivo y comenzará el proceso automatizado de instalación y, como todo instalador de Windows que se precie, prácticamente tan solo tendremos que pulsar en el botón **`SIGUIENTE`** en cada paso. Llegado el momento, podremos igualmente seleccionar la ruta de instalación, aunque es altamente recomendable instalar el contenido del paquete bajo el directorio **`C:\xampp\`**. Durante el proceso de instalación se nos preguntará si deseamos instalar las aplicaciones del paquete como servicios del sistema operativo; en este punto podemos elegir arrancar Apache y MariaDB en el inicio de Windows. No obstante, si no lo hacemos, siempre podemos configurar esta opción desde el **panel de control** de XAMPP, una vez instalado en nuestro ordenador. Éste será también el encargo de indicarnos qué aplicaciones están corriendo, permitiéndonos poner en marcha, detener y/o administrar cada una de ellas.



Finalizada la instalación, debemos comprobar que todo ha ido correctamente. Para ello, abrimos el navegador y escribimos la dirección http://localhost o http://127.0.0.1 y, si no se ha producido ningún error, aparecerá en la ventana un mensaje de bienvenida.

Por defecto, XAMPP no asigna clave de acceso al usuario **root** del motor MariaDB, haciendo accesible a través de la red, no sólo el servicio, sino también la aplicación de gestión **phpMyAdmin**, que nos servirá para crear y administrar las bases de datos en las que se apoyan las aplicaciones que estemos desarrollando. Dado que cada uno de estos puntos supone un alto riesgo en la seguridad del sistema, es de especial interés corregirlos lo antes posible, para lo cual tendremos que recurrir a los archivos de configuración.

5.2. Instalación en Linux

Generalmente, los repositorios de prácticamente todas las distribuciones Linux facilitan la descarga e instalación de **XAMPP**. No obstante, siempre podemos optar por descargar e instalar manualmente el paquete y, una vez alojado en nuestra máquina y utilizando el terminal como administrador, cambiaremos los permisos al instalador:

```
chmod 755 xamp-linux-*-installer.run
```

Y seguidamente lo lanzaremos:

```
sudo ./xamp-linux-*-installer.run
```

Eso es todo. A partir de este momento tendremos instalado el paquete XAMPP en el directorio **/opt/lamp**. Lanzaremos o detendremos el servicio utilizando los comandos:

```
sudo /opt/lampp/lampp start        sudo /opt/lampp/lampp stop
```

Sea como sea, en ambos casos se nos mostrará en el terminal una serie de mensajes que nos indicarán si el proceso de inicio o fin ha ido correctamente.

```
Starting XAMPP 1.8.2...
LAMPP: Starting Apache...
LAMPP: Starting MySQL...
LAMPP Started.
```

Llegados a este punto todo está preparado para empezar a trabajar con Apache, MySQL y PHP/Perl. XAMPP bajo Linux también nos ofrece un panel de control gráfico con el que podremos trabajar; accederemos a él utilizando el siguiente comando y desde el directorio de instalación:

```
sudo ./manager-linux.run
```

Podremos comprobar que la instalación y puesta en marcha del servicio ha ido sin problemas, accediendo a la dirección local de nuestra máquina, obteniendo una imagen similar a la que vimos en el caso de Windows. No debemos olvidarnos de corregir los posibles problemas de seguridad planteados, ya que, aunque el proceso de instalación y puesta en marcha es bastante simple, siempre pueden surgir problemas inesperados. No obstante, siempre podemos visitar la página web del proyecto XAMPP donde encontraremos abundante ayuda... eso sí, en inglés.