Unidad 1: Introducción al desarrollo de aplicaciones web

Índice

1.1. Modelos de programación en entornos cliente-servidor

Las aplicaciones web se basan, desde un punto de vista centrado en el hardware, en el modelo cliente-servidor.

En dicho modelo hay dos tipos de elementos con funciones diferenciadas: clientes y servidores.

Los servidores proveen servicios, como información o funcionalidad, a los clientes.

El cliente inicia el proceso cuando envía una solicitud al servidor que, a su vez, envía un mensaje de respuesta.

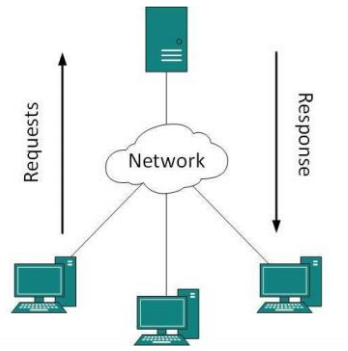


Figura 1. Arquitectura cliente-servidor.

Cliente y servidor se comunican mediante un protocolo que define el formato de solicitudes y respuestas y, en general, se ejecutan en máquinas diferentes conectadas por red, aunque también pueden ejecutarse en el mismo ordenador.

En el caso de los servidores web, los clientes son, la mayoría de las veces, los navegadores que realizan peticiones de páginas al servidor. Éste responde con la página web solicitada o con un mensaje de error si no la encuentra o el acceso no está autorizado.

Figura 2. Modelo cliente-servidor.

La comunicación entre clientes y servidores web se realiza mediante protocolo HTTP o su versión segura, HTTPS, que tienen reservados los puertos 80 y 553 respectivamente en la lista de puertos bien conocidos. Las características del modelo cliente-servidor y del protocolo HTTP condicionan el desarrollo de aplicaciones web.

Al programar una aplicación, se utiliza un lenguaje de programación y ciertas herramientas como un entorno de desarrollo o librerías de código. Además, una vez acabado su desarrollo necesitará ciertos componentes para su ejecución, como por ejemplo una máquina virtual de Java.

Los **componentes principales** con los que debes contar para ejecutar aplicaciones web en un servidor son los siguientes:

- Un **servidor web** para recibir las peticiones de los clientes web (normalmente navegadores) y enviarles la página que solicitan (una vez generada puesto que hablamos de páginas web dinámicas). El servidor web debe conocer el procedimiento a seguir para generar la página web: qué módulo se encargará de la ejecución del código y cómo se debe comunicar con él.
- El **módulo encargado de ejecutar el código** o programa y generar la página web resultante. Este módulo debe integrarse de alguna forma con el servidor web, y dependerá del lenguaje y tecnología que utilicemos para programar la aplicación web.
- Una **aplicación de base de datos**, que normalmente también será un servidor. Este módulo no es estrictamente necesario, pero en la práctica se utiliza en todas las aplicaciones web que utilizan grandes cantidades de datos para almacenarlos.
- El **lenguaje de programación** que utilizarás para desarrollar las aplicaciones.

Además de los componentes a utilizar, también es importante decidir cómo vas a **organizar el código** de la aplicación. Muchas de las arquitecturas que se usan en la programación de aplicaciones web te ayudan a estructurar el código de las aplicaciones en capas o niveles.

El motivo de dividir en capas el diseño de una aplicación es que se puedan separar las funciones lógicas de la misma, de tal forma que sea posible ejecutar cada una en un servidor distinto (en caso de que sea necesario). [1]

Cada capa puede ocuparse de una o varias de las funciones.

Por ejemplo, en la **arquitectura a tres capas** (una ampliación del modelo cliente-servidor) la lógica de la aplicación se separa en:

a) Capa de presentación o de cliente. Es la parte visible de la aplicación. Muestra información y permite interactuar con el sistema al usuario. Aquí se programará todo lo relacionado con la interfaz de usuario

b) Capa de negocio. En esta capa intermedia se programa la funcionalidad de la aplicación. Se comunica con las otras dos capas. Siempre se ejecuta en el lado del servidor. Esta capa, tras realizar todos los cálculos y/o operaciones sobre los datos, genera el código HTML que será presentado al usuario en la capa siguiente.

[1] En una aplicación puedes distinguir, de forma general, funciones de presentación (se encarga de dar formato a los datos para presentárselo al usuario final), lógica (utiliza los datos para ejecutar un proceso y obtener un resultado), persistencia (que mantiene los datos almacenados de forma organizada) y acceso (que obtiene e introduce datos en el espacio de almacenamiento).

c) Capa de datos. Para gestionar la base de datos, es decir se encarga de almacenar la información de la aplicación en una base de datos y recuperarla cuando sea necesario.

La capa de presentación no se comunica directamente con la de datos, sino que se hace a través de la capa de negocio. La capa de negocio recibe las acciones del usuario de la capa de presentación y, si es necesario, se comunica con la capa de datos para consultar o modificar la base de datos. De la misma manera, pasa la salida de la capa de datos a la de presentación.

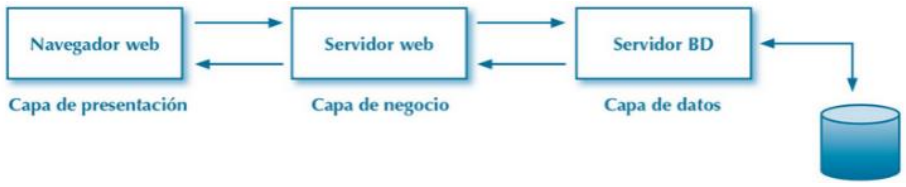


Figura 3. Modelo a 3 capas.

En una aplicación web, la capa de presentación se ejecuta en un navegador, en el ordenador del usuario; la capa de negocio, en el servidor web, y la de datos, en el servidor de bases de datos.

Estas dos pueden estar en equipos diferentes o en el mismo.

El objetivo es desacoplar la lógica de negocio de la de presentación para conseguir código más reutilizable. Además, mediante la separación entre componentes se obtienen aplicaciones más fáciles de mantener y ampliar.

Por ejemplo, al hacer login en una aplicación web:

- Desde el cliente (navegador) se envía un formulario. Es la capa de presentación.
- En el servidor, capa de negocio, se consulta con la base datos (capa de datos) si los datos son correctos.
- En función de la respuesta, se pasan las instrucciones correspondientes a la capa de presentación (se permite el acceso o se muestra un mensaje de error).

1.2. Páginas web estáticas y dinámicas

El lenguaje básico para la web es el HTML. Una página que esté escrita usando solo HTML será estática, es decir, mostrará siempre el mismo contenido. Cuando se utiliza un lenguaje de programación en el servidor se pueden generar páginas en función de la solicitud del cliente.

Por ejemplo, muchas páginas muestran anuncios diferentes según dónde esté el usuario. Es decir, no se ofrece el mismo contenido a todos los usuarios, sino que parte del contenido se genera dinámicamente. En este caso se habla de páginas web dinámicas.

1.2.1 Páginas web estáticas

Estas páginas se encuentran almacenadas en su forma definitiva, tal y como se crearon, y **su contenido no varía.** Son útiles para mostrar una información concreta, y mostrarán esa misma información cada vez que se carguen. La única forma en que pueden cambiar es si un programador la modifica y actualiza su contenido.

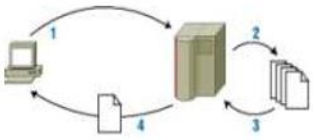
Cuando una **página web se descarga** a tu ordenador, su contenido define qué se debe mostrar en pantalla. Este contenido está programado en un **lenguaje de marcado**, formado por etiquetas, que puede ser HTML o XHTML. Las etiquetas que componen la página indican el objetivo de cada una de las partes que la componen. Así, dentro de estos lenguajes hay etiquetas para indicar que un texto es un encabezado, que forma parte de una tabla, o que simplemente es un párrafo de texto.

Además, si la página está bien estructurada, la información que le indica al navegador el estilo con que se debe mostrar cada parte de la página estará almacenado en otro fichero, una **hoja de estilos** o **CSS**. La hoja de estilos se encuentra indicada en la página web y el navegador la descarga junto a ésta. En ella nos podemos encontrar, por ejemplo, estilos que indican que el encabezado debe ir con tipo de letra Arial y en color rojo, o que los párrafos deben ir alineados a la izquierda.

Estos dos ficheros se descargan a tu ordenador desde un servidor web como respuesta a una petición.

El esquema de funcionamiento es el siguiente:

1. Tu ordenador solicita a un servidor web una página con extensión .htm, .html o .xhtml.



2. El servidor busca esa página en un almacén de páginas (cada una suele ser un fichero).

3. Si el servidor encuentra esa página, la recupera.

4. Y por último se la envía al navegador para que éste pueda mostrar su contenido.

Este es un ejemplo típico de una comunicación cliente-servidor. El cliente es el que hace la petición e inicia la comunicación, y el servidor es el que recibe la petición y la atiende. En nuestro caso, el navegador es el cliente web.

1.2.2 Páginas web dinámicas

Estas páginas, como su nombre indica, se caracterizan porque su contenido cambia en función de diversas variables, como puede ser el navegador que estás usando, el usuario con el que te has identificado, o las acciones que has efectuado con anterioridad.

Dentro de las páginas web dinámicas, es muy importante distinguir dos **tipos**:

- Aquellas **que incluyen código que ejecuta el navegador**. En estas páginas el código ejecutable, normalmente en lenguaje **JavaScript**, se incluye dentro del HTML (o XHTML) y se descarga junto con la página. Cuando el navegador muestra la página en pantalla, ejecuta el código que la acompaña. Este código puede incorporar múltiples funcionalidades que pueden ir desde mostrar animaciones hasta cambiar totalmente la apariencia y el contenido de la página. En este módulo no vamos a ver JavaScript, salvo cuando éste se relaciona con la programación web del lado del servidor.
- Como ya sabes, hay muchas páginas en Internet que no tienen extensión .htm, .html o .xhtml. Muchas de estas páginas tienen extensiones como .php, .asp, .jsp, .cgi o .aspx. En éstas, el contenido que se descarga al navegador es similar al de una página web estática: HTML (o XHTML). Lo que cambia es la forma en que se obtiene ese contenido. Al contrario de lo que hemos visto hasta ahora, esas páginas no están almacenadas en el servidor; más concretamente, el contenido que se almacena no es el mismo que después se envía al navegador. **El HTML de estas páginas se forma como resultado de la ejecución de un programa**, y esa ejecución tiene lugar en el servidor web (aunque no necesariamente por ese mismo servidor).

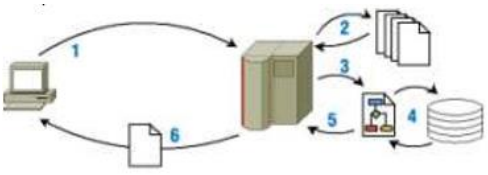
El esquema de funcionamiento de una página web dinámica es el siguiente:

1. El cliente web (navegador) de tu ordenador solicita a un servidor web una página web.

2. El servidor busca esa página y la recupera.

3. En el caso de que se trate de una página web dinámica, es decir, que su contenido deba ejecutarse para obtener el HTML que se devolverá, el servidor web contacta con el módulo responsable de ejecutar el código y se lo envía.

4. Como parte del proceso de ejecución, puede ser necesario obtener información de algún repositorio (Cualquier almacén de información digital, normalmente una base de datos), como por ejemplo consultar registros almacenados en una base de datos.



5. El resultado de la ejecución será una página en formato HTML, similar a cualquier otra página web no dinámica.

6. El servidor web envía el resultado obtenido al navegador, que la procesa y muestra en pantalla.

Este procedimiento tiene lugar constantemente mientras consultamos páginas web. Por ejemplo, cuando consultas tu correo en GMail, Proton Mail, Yahoo o cualquier otro servicio de correo vía web, lo primero que tienes que hacer es introducir tu nombre de usuario y contraseña. A continuación, lo más habitual es que el servidor te muestre una pantalla con la bandeja de entrada, en la que aparecen los mensajes recibidos en tu cuenta. Esta pantalla es un claro ejemplo de una página web dinámica.

Obviamente, el navegador no envía esa misma página a todos los usuarios, sino que la genera de forma dinámica en función de quién sea el usuario que se conecte. Para generarla ejecuta un programa que obtiene los datos de tu usuario (tus contactos, la lista de mensajes recibidos) y con ellos compone la página web que recibes desde el servidor web.

Para **ejecutar código en un servidor web** hay varias opciones. Las más habituales son:

1. *Common Gateway Interface (CGI).* Las peticiones de los clientes se pasan a un ejecutable en el servidor. Este programa genera la salida y el servidor se la pasa al cliente.

2. *Como módulo del servidor web*. En lugar de utilizar un programa externo, el propio servidor cuenta con un módulo, generalmente un intérprete, para ejecutar código.

3. *Servlets*. Un servlet es un programa Java que se ejecuta en un servidor Web y construye o sirve páginas web. De esta forma se pueden construir páginas dinámicas, basadas en diferentes fuentes variables: datos proporcionados por el usuario, fuentes de información variable (páginas de noticias, por ejemplo), o programas que extraigan información de bases de datos.

Para utilizar servlets hace falta un contenedor web (por ejemplo Tomcat), que es el que interactúa con ellos.

Comparado con un CGI, un servlet es más sencillo de utilizar, más eficiente (se arranca un hilo por cada petición y no un proceso entero), más potente y portable.

1.2.3 Ventajas e inconvenientes

Aunque la utilización de páginas web dinámicas parece la mejor opción para construir un sitio web, no siempre lo es. Sin lugar a duda es la que más potencia y flexibilidad permite, pero las páginas **web estáticas** tienen también algunas **ventajas**:

- **No es necesario saber programar** para crear un sitio que utilice únicamente páginas web estáticas. Simplemente habría que conocer HTML/XHTML y CSS, e incluso esto no sería indispensable: se podría utilizar algún programa de diseño web para generarlas.
- La característica diferenciadora de las páginas web estáticas es que **su contenido nunca varía,** y esto en algunos casos también puede suponer una ventaja. Sucede, por ejemplo, cuando quieres almacenar un enlace a un contenido concreto del sitio web: si la página es dinámica, al volver a visitarla utilizando el enlace su contenido puede variar con respecto a cómo estaba con anterioridad. O cuando quieres dar de alta un sitio que has creado en un motor de búsqueda como Google.

Para que Google muestre un sitio web en sus resultados de búsqueda, previamente tiene que indexar su contenido. Es decir, un programa recorre las páginas del sitio consultando su contenido y clasificándolo. Si las páginas se generan de forma dinámica, puede ser que su contenido, en parte o por completo, no sea visible para el buscador y por tanto no quedará indexado. Esto nunca sucedería en un sitio que utilizase páginas web estáticas.

Como ya sabes, para que un servidor web pueda procesar una página web dinámica, necesita ejecutar un programa. Esta ejecución la realiza un módulo concreto, que puede estar integrado en el servidor o ser independiente.

Además, puede ser necesario consultar una base de datos como parte de la ejecución del programa. Es decir, la ejecución de una página web dinámica requiere una serie de recursos del lado del servidor.

Estos recursos deben instalarse y mantenerse. **Las páginas web estáticas sólo necesitan un servidor web que se comunique con tu navegador** para enviártela. Y de hecho para ver una página estática almacenada en tu equipo no necesitas siquiera de un servidor web. Son archivos que pueden almacenarse en un soporte de almacenamiento como puede ser un disco óptico o una memoria USB y abrirse desde él directamente con un navegador web.

Pero si decides hacer un sitio web utilizando páginas estáticas, ten en cuenta que tienen **limitaciones**. La desventaja más importante ya la comentamos anteriormente: la **actualización** de su contenido debe hacerse de forma **manual** editando la página que almacena el servidor web. Esto implica un mantenimiento que puede ser prohibitivo en sitios web con gran cantidad de contenido.

1.3 Aplicaciones web

Una aplicación web es una aplicación informática a la que se accede mediante una interfaz web utilizando un navegador.

Las aplicaciones web emplean **páginas web dinámicas** para crear aplicaciones que **se ejecuten en un servidor web** y se muestren en un navegador. Puedes encontrar aplicaciones web para realizar múltiples tareas. Unas de las primeras en aparecer fueron los clientes de correo, que te permiten consultar los mensajes de correo recibidos y enviar los tuyos propios utilizando un navegador.

Hoy en día existen aplicaciones web para multitud de tareas como procesadores de texto, gestión de tareas, o edición y almacenamiento de imágenes. Estas aplicaciones tienen ciertas ventajas e inconvenientes si las comparas con las aplicaciones tradicionales que se ejecutan sobre el sistema operativo de la propia máquina.

Ventajas de las aplicaciones web:

- No es necesario instalarlas en aquellos equipos en que se vayan a utilizar. Se instalan y se ejecutan solamente en un equipo, en el servidor, y esto es suficiente para que se puedan utilizar de forma simultánea desde muchos equipos.
- Como solo se encuentran instaladas en un equipo, es muy sencillo gestionarlas (hacer copias de seguridad de sus datos, corregir errores, actualizarlas).
- Se pueden utilizar en todos aquellos sistemas que dispongan de un navegador web, independientemente de sus características (no es necesario un equipo potente) o de su sistema operativo.
- Se pueden utilizar desde cualquier lugar en el que dispongamos de conexión con el servidor. En muchos casos esto hace posible que se pueda acceder a las aplicaciones desde sistemas no convencionales, como por ejemplo teléfonos móviles.

Inconvenientes de las aplicaciones web:

- La interfaz de usuario de las aplicaciones web es la página que se muestra en el navegador. Esto restringe las características de la interfaz a aquellas de una página web.
- Dependemos de una conexión con el servidor para poder utilizarlas. Si nos falla la conexión, no podremos acceder a la aplicación web.
- La información que se muestra en el navegador debe transmitirse desde el servidor. Esto hace que cierto tipo de aplicaciones no sean adecuadas para su implementación como aplicación web (por ejemplo, las aplicaciones que manejan contenido multimedia, como las de edición de vídeo).

Hoy en día muchas aplicaciones web utilizan las ventajas que les ofrece la generación de páginas dinámicas. La gran mayoría de su contenido está almacenado en una base de datos. Aplicaciones como Drupal, Joomla!, etc. ofrecen dos partes bien diferenciadas:

- Una parte externa o **front-end**, que es el conjunto de páginas que ven la gran mayoría de usuarios que las usan (usuarios externos).
- Una parte interna o **back-end**, que es otro conjunto de páginas dinámicas que utilizan las personas que producen el contenido y las que administran la aplicación web (usuarios internos) para crear contenido, organizarlo, decidir la apariencia externa, etc.

1.4 Arquitecturas y plataformas

La primera elección que harás antes de comenzar a programar una aplicación web es la arquitectura que vas a utilizar. Hoy en día, puedes elegir entre:

- Java EE (Enterprise Edition), antes conocida como J2EE. Es una plataforma orientada a la programación de aplicaciones en lenguaje Java. Puede funcionar con distintos gestores de bases de datos, e incluye varias librerías y especificaciones para el desarrollo de aplicaciones de forma modular.

  Está apoyada por grandes empresas como Sun y Oracle, que mantienen Java, o IBM. Es una buena solución para el desarrollo de aplicaciones de tamaño mediano o grande. Una de sus principales ventajas es la multitud de librerías existentes en ese lenguaje y la gran base de programadores que lo conocen.

  Dentro de esta arquitectura existen distintas tecnologías como las páginas JSP y los servlets, ambos orientados a la generación dinámica de páginas web, o los EJB, componentes que normalmente aportan la lógica de la aplicación web.
- AMP (Apache, MySQL y PHP/Perl/Python). Las dos primeras siglas hacen referencia al servidor web (Apache) y al servidor de base de datos (MySQL). La última se corresponde con el lenguaje de programación utilizado, que puede ser PHP, Perl o Python,

  Dependiendo del sistema operativo que se utilice para el servidor, se utilizan las siglas LAMP (para Linux), WAMP (para Windows) o MAMP (para Mac). También es posible usar otros componentes, como el gestor de bases de datos PostgreSQL en lugar de MySQL.

  Todos los componentes de esta arquitectura son de código libre. Es una plataforma de programación que permite desarrollar

aplicaciones de tamaño pequeño o mediano con un aprendizaje sencillo. Su gran ventaja es la gran comunidad que la soporta y la multitud de aplicaciones de código libre disponibles.

- CGI/Perl. Es la combinación de dos componentes: Perl, un potente lenguaje de código libre creado originalmente para la administración de servidores, y CGI, un estándar para permitir al servidor web ejecutar programas genéricos, escritos en cualquier lenguaje (también se pueden utilizar por ejemplo C o Python), que devuelven páginas web (HTML) como resultado de su ejecución. Es la más primitiva de las arquitecturas que comparamos aquí.
  El principal inconveniente de esta combinación es que CGI es lento, aunque existen métodos para acelerarlo. Por otra parte, Perl es un lenguaje muy potente con una amplia comunidad de usuarios y mucho código libre disponible.

- ASP.Net es la arquitectura comercial propuesta por Microsoft para el desarrollo de aplicaciones. Es la parte de la plataforma .Net destinada a la generación de páginas web dinámicas. Proviene de la evolución de la anterior tecnología de Microsoft, ASP.

  El lenguaje de programación puede ser Visual Basic.Net o C#. La arquitectura utiliza el servidor web de Microsoft, IIS, y puede obtener información de varios gestores de bases de datos entre los que se incluye, como no, Microsoft SQL Server.

  Una de las mayores ventajas de la arquitectura .Net es que incluye todo lo necesario para el desarrollo y el despliegue de aplicaciones. Por ejemplo, tiene su propio entorno de desarrollo, Visual Studio, aunque hay otras opciones disponibles. La mayor desventaja es que se trata de una plataforma comercial de código propietario.

1.5 Lenguajes de programación en entorno servidor

Los más habituales son:

a) PHP. Sin duda, el lenguaje más extendido en el lado del servidor. Normalmente se ejecuta como un módulo del servidor.

b) JSP. La versión Java de PHP. Para utilizarlo hace falta un contenedor web.

c) ASP.NET. La alternativa a PHP de Microsoft, integrada en la plataforma.NET.

d) PERL. Muy utilizado para CGI. Es un lenguaje especialmente pensado para el procesamiento de expresiones regulares.

e) Ruby. Es un lenguaje orientado a objetos muy apreciado por desarrolladores web.

También existen frameworks para desarrollo de aplicaciones web:

Framework	Descripción
Spring	El *framework* más extendido para JEE.
Ruby on Rails	Muy popular para el desarrollo MVC, ha influenciado otros muchos *frameworks* extendidos. En Ruby.
Django	Basado en Python, sigue el modelo MVT.
AngularJS	*Framework* de Google para el lado del cliente basado en JavaScript.
Symfony	*Framework* para el desarrollo MVC en PHP.

1.6 Integración con los lenguajes de marcas

Una de las formas de realizar páginas web dinámicas es integrar las etiquetas HTML en el código de los programas. De esta forma se programan, por ejemplo, los guiones CGI y los servlets.

Sin embargo, nosotros nos centraremos en un enfoque distinto que consiste en **integrar el código del programa en medio de las etiquetas HTML de la página web.**

El contenido que no varía de la página se introduce en HTML, y en el lenguaje de programación todo aquello que puede variar de forma dinámica.

Cuando se solicita una página, el servidor busca en ella bloques de código. Si los encuentra, los ejecuta y los sustituye por su salida.

En PHP los bloques se delimitan por **<?php** y **¿>**

```
1  <!DOCTYPE html>
2  <html>

3      <head>
4          <title>Número aleatorio</title>
5      </head>
6      <body>
7          <?php
8              echo rand(0, 100);
9          ?>
10     </body>
11 </html>
```

Al solicitar la página, el servidor ejecuta el bloque y lo sustituye por un número aleatorio entre 0 y 100. Si por ejemplo sale un 8, enviará la cliente:

```
<!DOCTYPE html>
<html>
    <head>
        <title> Número aleatorio </title>
    </head>
    <body>
        8
    </body>
</html>
```

1.7 Instalación del entorno de trabajo

Para empezar a trabajar necesitaremos distintas herramientas, además de las que se tengan que utilizar en el servidor una vez que la aplicación se ejecute, que servirán de gran ayuda en el desarrollo de la aplicación.

Imprescindibles para comenzar:

- Un navegador: Chrome, Firefox, Opera, etc.
- Si tienes pocos recursos en tu máquina, instala un editor de código como, por ejemplo, Sublime Text.
- Si no, es mejor instalar un entorno de desarrollo integrado (IDE) que nos proporciona herramientas de reconocimiento de clases avanzado, integración con el SGBD, etc.
    - NetBeans
    - Eclipse PDT
    - PHPStorm
    - Aptana
- Conocer un lenguaje de programación web: PHP
- Un servidor de bases de datos:
    - MySQL
    - MariaDB
- Un servidor web
    - Remoto: podemos usar un hosting gratuito
    - Local:al principio será lo más cómodo para empezar a trabajar, ya que sólo tendremos que descargar e instalar el paquete apropiado para nuestro equipo.
      Se recomienda comenzar en una máquina virtual en la que tengamos todos los componentes.

      En caso de que instalemos Linux la solución será una arquitectura LAMP (**L**inux **A**pache **M**ySQL **P**HP).

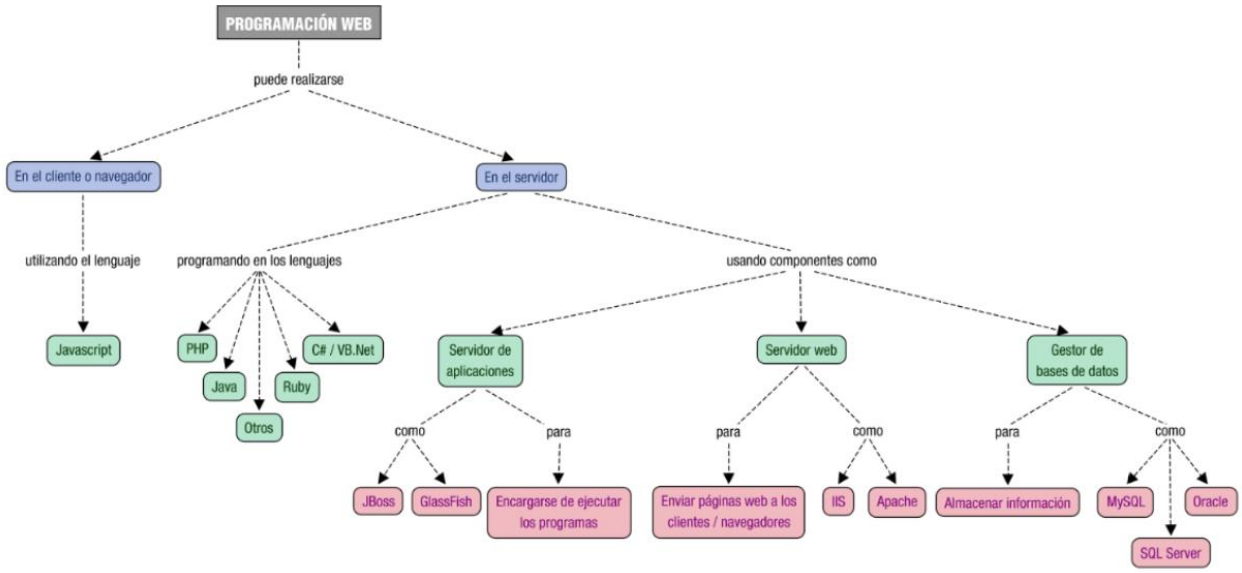
      Si vas a trabajar en Windows puedes usar WampServer.

      Si no lo tienes claro y prefieres un paquete que pueda instalarse tanto en Linux como en Windows utiliza XAMPP.

En cualquiera de ellos se incluye un servidor web Apache y un servidor de bases de datos MySQL o MariaDB.



Nota: *Este documento es una recopilación de datos del libro Desarrollo web en entorno servidor de Xabier Ganzábal y de los apuntes de Desarrollo web en entorno servidor de José Luis Comesaña.*