-

UD 01.
ARQUITECTURA WEB

Desarrollo web entorno servidor
CFGS DAW

Francisco Aldarias Raya
paco.aldarias@ceedcv.es
2019/2020
Versión:190918.0046

Licencia

Nomenclatura

A lo largo de este tema se utilizarán distintos símbolos para distinguir elementos importantes dentro del contenido. Estos símbolos son:

 Importante

 Atención

 Interesante

ÍNDICE DE CONTENIDO

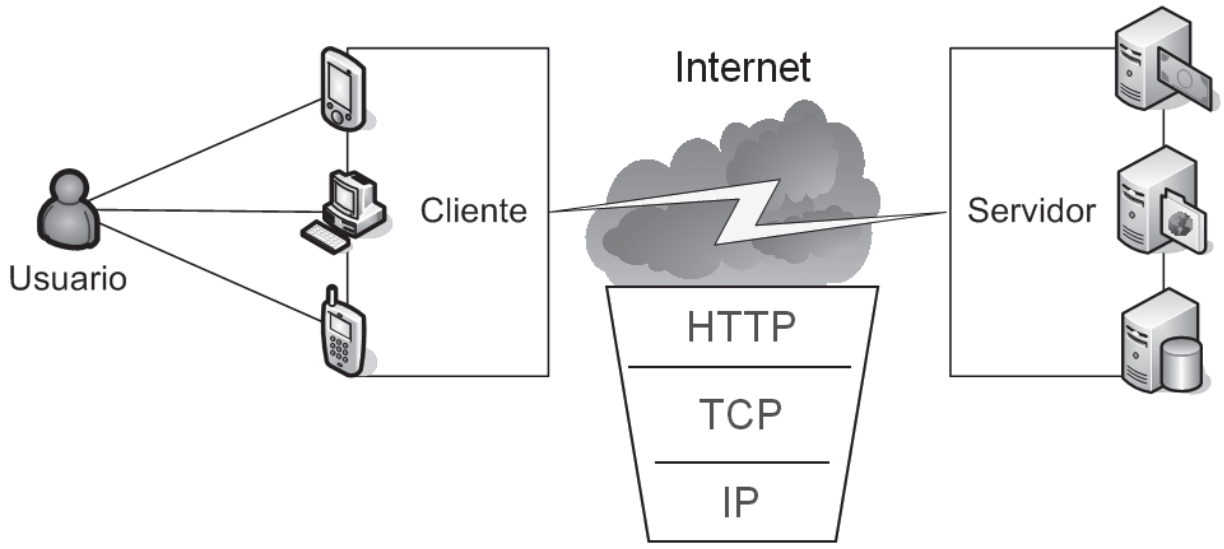
UD01. ARQUITECTURA WEB

1. ARQUITECTURA CLIENTE SERVIDOR

El modelo de desarrollo web se apoya, en una primera aproximación desde un punto de vista centrado en el hardware, en lo que se conoce como arquitectura cliente-servidor que define un patrón de arquitectura donde existen dos actores, cliente y servidor, de forma que el primero es quién se conecta con el segundo para solicitar algún servicio. En el caso que nos ocupa, el desarrollo web, los clientes solicitan que se les sirva una web para visualizarla, aunque también es posible solicitar información si hablamos del caso de los servicios web que también veremos más adelante. En cualquier caso, en ambos casos aparece el mismo escenario, donde un servidor se encuentra ejecutándose ininterrupidamente a la espera de que los diferentes clientes realicen una solicitud.

Normalmente a la solicitud que hacen los clientes al servidor se le llama petición (request) y a lo que el servidor devuelve a dicho cliente le llamamos respuesta (request).

También hay que tener en cuenta que esta arquitectura cliente-servidor plantea la posibilidad de numerosos clientes atendidos por un mismo servidor. Es decir, el servidor será un software multitarea que será capaz de atender peticiones simultáneas de numerosos clientes.



1.1 Clientes

Los clientes en una arquitectura cliente-servidor cumplen:

- Originan el tráfico web ya que envían las peticiones y reciben las respuestas.
- Dos clases de clientes web: navegadores y robots.
- Los navegadores (Netscape, IE, Chrome, Opera, FireFox, Safari).
    - Las peticiones están dirigidas por el usuario.
    - Repiten peticiones al mismo objeto cuando navegan por un site.
    - Utilizan caches de memoria y disco.
- Robots (Motores de búsqueda) donde las peticiones son automatizadas

1.2 Funciones de los navegadores

- Construyen y envían la petición HTTP.
- Reciben, interpretan y presentan la respuesta.
- Proporcionan el interfaz para conectarse y utilizar otros servicios: mail, news, ftp, etc. donde el protocolo por defecto es http.
- La caché local sirve recursos guardados sin conectarse al servidor.
- Manejo de las Cookies.

1.3 Servidores

El intercambio de información cliente/servidor puede requerir también:
Controlar el acceso a recursos restringidos:

- Autenticación.
    - Piden al usuario que se identifique (login y password)
    - La información se incluye en la cabecera del mensaje.
- Autorización.

- Comprobar en la lista de acceso si el usuario está autorizado.

2. APLICACIONES WEB

Una aplicación web es proporcionada por un servidor web y utilizada por usuarios que se conectan desde cualquier punto vía clientes web (navegadores).

> **Importante**
>
> Definición Aplicación web. Son aplicaciones basadas en el modelo Cliente/Servidor gestionadas por servidores web, y que utilizan como interfaz páginas web.

La colección de páginas son en una buena parte dinámicas (ASP, PHP, etc.), y están agrupadas lógicamente para dar un servicio al usuario.

El acceso a las páginas está agrupado también en el tiempo (sesión). Ejemplos: venta de libros, reserva de billetes, etc.

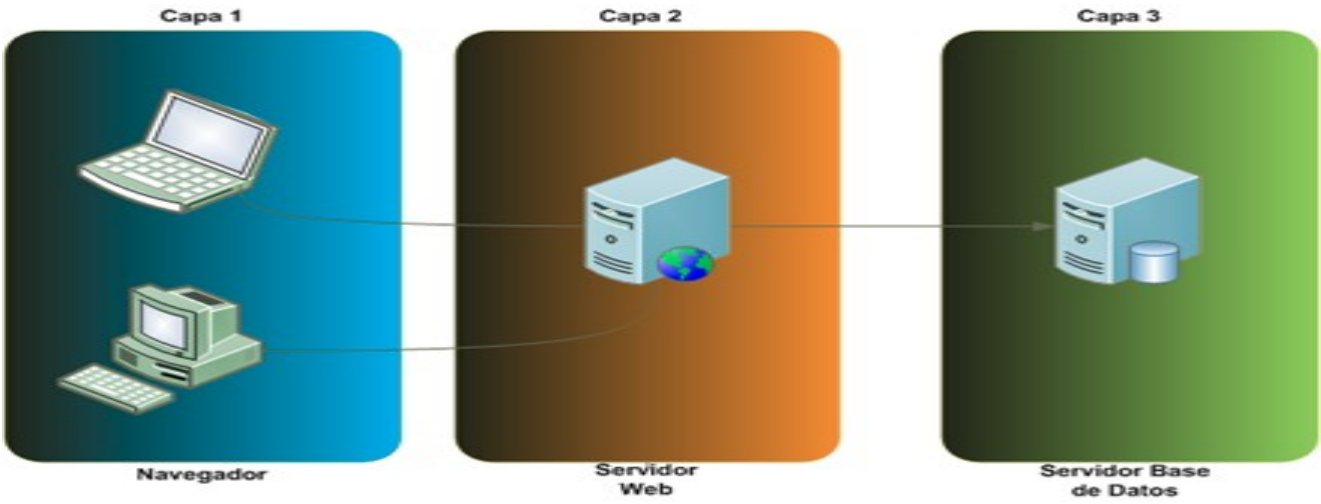
3. ARQUITECTURA DE LAS APLICACIONES WEB

Las funcionalidades en la arquitectura cliente/servidor de la web se agrupan en diferentes capas.

Cada capa se centra en la gestión de una aspecto determinado del sistema web.

El modelo más extendido divide la arquitectura cliente/servidor en tres capas:

- Capa de presentación
- Capa de negocio
- Capa de persistencia o de datos



- **Capa de presentación**: Es la capa donde la aplicación se muestra al usuario. Básicamente es

la (Graphical User Interface, Interfaz Gráfica de Usuario). En el caso de una aplicación web sería el código que se carga directamente en el navegador web. En cualquier caso, se ejecuta directamente en el equipo del cliente.

- **Capa de negocio**: Es la capa intermedia donde se lleva a cabo toda la lógica de la aplicación. Siempre se ejecutará en el lado servidor. Esta capa, tras realizar todos los cálculos y/o operaciones sobre los datos, genera el código que será presentado al usuario en la capa siguiente.
- **Capa de datos**: Es la capa que almacena los datos. Básicamente, en condiciones normales, hace referencia al propio SGBD que es el encargado de almacenar los datos. Dependiendo de la arquitectura de la aplicación, esta capa y la de negocio se pueden encontrar fisicamente en el mismo equipo, aunque también es posible que se tengan que separar por cuestiones de rendimiento. La capa de datos sirve todas la información necesaria a la capa de negocio para llevar a cabo sus operaciones.

Si nos imaginamos una tienda online, la capa de datos almacena toda la información en una Base de Datos (usuarios, pedidos, articulos, ofertas, . . .), la capa de negocio debe acceder a esa información y, tras procesar un pedido, por ejemplo, debe presentar el resultado final al usuario en el navegador, que es la capa de presentación.

4. SERVIDORES WEB Y SERVIDORES DE APLICACIONES

Asi como las aplicaciones de escritorio se ejecutan directamente sobre el propio Sistema Operativo, las páginas y aplicaciones web necesitan de una herramienta adicional que permita desplegarlas para su puesta en marcha. Hablamos de servidores web y servidores de aplicaciones , respectivamente.

4.1 ¿Qué es un servidor web?

Un servidor web es una aplicación que recibe una petición HTTP (normalmente a través de un navegador web) y devuelve la página web solicitada (escrita en lenguaje y pudiendo contener código Javascript inscrustado) para que ésta sea interpretada y visualizada por el navegador de quién realizó la solicitud (el usuario).

4.2 Qué es un servidor de aplicaciones

Un servidor de aplicaciones es una aplicación que contiene una serie de servicios los cuales están accesibles a través de una expuesta a través de Internet. Normalmente los servidores de aplicaciones proporcionan más servicios que los servidores web. Por ejemplo, en el caso de los servidores de aplicaciones para *Java* o *Python*, éstos proporcionan un acceso transparente a la Base de Datos para que el desarrollador se centre exclusivamente en implementar la capa de negocio. Además, pueden proporcionar también servicios como fail-over o balanceo de carga.

4.3 Apache

Apache es uno de los servidores web más conocidos. Es software libre y multiplataforma, aunque aproximadamente el 90% de los servidores Apache se ejecutan actualmente en entornos Linux puesto que es el servidor preferido para esta plataforma.

Es muy modular lo que permite incorporar características una vez instalado y puesto en marcha. Eso le hace también muy flexible pudiendo dar servicio a webs escritas en los lenguajes de programación web más extendidos (como PHP, Python, ASP, . . .) a través del módulo correspondiente.

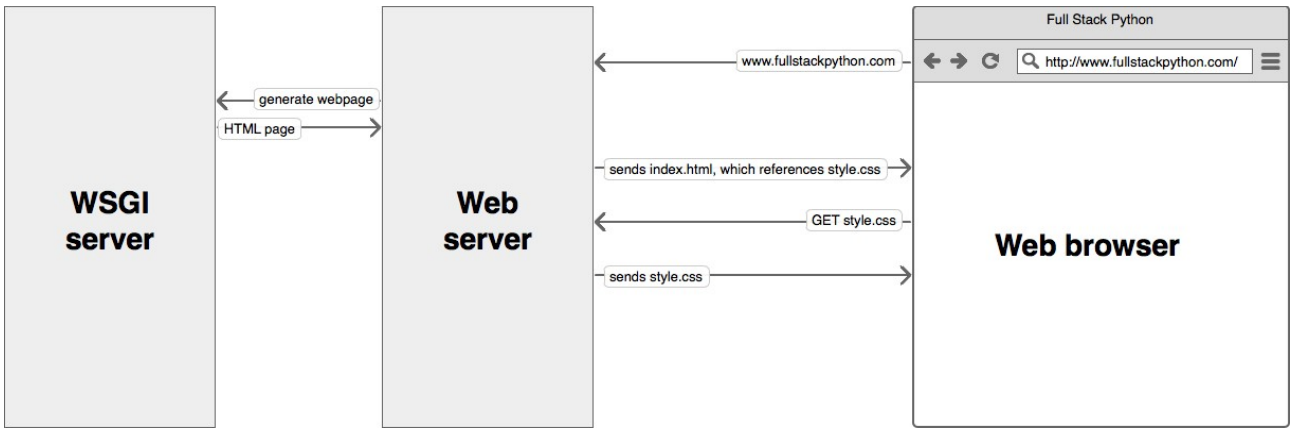
4.4 Apache Tomcat

Apache Tomcat es un servidor de aplicaciones que funciona como contenedor de Servlets de Java. Actualmente es capaz de implementar varias especificaciones de Java EE como Servlets y JSP (Java Server Pages) y además proporciona un servidor web *puro* para que se use en combinación con el entorno Java.

4.5 WSGI (Web Server Gateway Interface)

WSGI es un interfaz que define como se comunica un servidor web con aplicaciones web o frameworks escritos en Python.

En nuestro caso utilizaremos más adelante el módulo WSGI de Apache para utilizar este servidor web como plataforma para las aplicaciones web que desarrollemos en Python con el framework Django.



5. TIPOS DE PÁGINAS WEB

Al observar la capacidad de las aplicaciones web de comunicarse con los usuarios, las podemos clasificar en:

- Aplicaciones web estáticas
- Aplicaciones web dinámicas
- Aplicaciones web interactivas

5.1 Aplicaciones web estáticas

El usuario recibe una página web desde el servidor, que no conlleva ningún tipo de acción, ni en la propia página, ni genera ninguna respuesta por parte del servidor.

Utiliza HTML para la organización visual de la información.

Ejemplo: Aplicación web estática:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es-ES" lang="es-ES">
  <head>
        <title>T&iacute;tulo de la p&aacute;gina</title>
        <meta http-equiv="content-type" content="text/html; charset=utf-8">
  </head>
  <body>
        <p>Ha accedido a las 9:20</p>
  </body>
</html>
```

5.2 Aplicaciones web dinámicas

La interacción del cliente con la página web recibida desde el servidor produce algún cambio en la visualización de la misma (cambio de formato, ocultación de partes de la página, comienzo de animaciones, aparición de elementos nuevos, …). El servidor web no cambia la página.

Incluye DHTML, Flash, CSS, JavaScript, …

Ejemplo aplicación web dinámica:

```
<html>
<head>
<title>Nombre de mes</title>
<body>
<h3>Ejemplo de una p&aacute;gina con calendario que da el mes en letra</h3>
<script language="JavaScript" >
var hoy = new Date();
dia = hoy.getDate();
mes = hoy.getMonth();
ano = hoy.getYear() + 1900;
document.write (dia);
```

```
document.write ("/");
document.write (mes);
document.write ("/");
document.write (ano);
</script>
</body>
</html>
```

5.3 Aplicaciones web interactivas

La interacción del cliente con la página web recibida desde el servidor hace que se genere un diálogo entre ambos.
Son las aplicaciones que más se utilizan en Internet actualmente.
En el lado cliente encontramos HTML, controles ActiveX, Flash, applets, AJAX, etc.
En el lado servidor se utilizan lenguajes embebidos en código HTML como PHP, ASP, JSP, enlaces a ejecutables CGI, servlets, ASP.net.

Ejemplo aplicación web interactiva:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es-ES" lang="es-ES">
  <head>
      <title>T&iacute;tulo de la p&aacute;gina</title>
      <meta http-equiv="content-type" content="text/html; charset=utf-8">
  </head>
  <body>
      <p>Ha accedido a las <?php echo date('G:i');?></p>
  </body>
</html>
```

6. LENGUAJES DE PROGRAMACIÓN EN ENTORNO SERVIDOR

Son aquellos cuyo código, precompilado o interpretado, es ejecutado en el servidor por un software específico para dicho código.

Existen múltiples alternativas a la hora de ejecutar código en el servidor:

- Lenguajes de scripting (PHP, ASP, JSP, …)
- Enlaces a código ejecutable (CGI, JSP, EJB, …)
- Estrategias híbridas que utilizan tanto enlaces a código ejecutable como código embebido en la propia página web (Web Forms de ASP.net).

6.1 Lenguajes de scripting

Código que se intercala con el código HTML de una aplicación web.

El código está formado por instrucciones escritas en multitud de lenguajes de programación que son ejecutadas por un servidor para proporcionar dinamismo a la página web.

El servidor web debe tener instalado un módulo o programa que le permita interpretar el lenguaje de programación del código embebido en la página web.

PHP (Pre Hypertext Processor)

- gratuidad, código abierto, portable en diferentes plataformas
- lenguaje interpretado
- construcciones orientadas a objetos
- lo soportan la mayoría de los servidores web
- lenguaje interpretado

ASP (Active Server Pages)

- tecnología propietaria de código cerrado de Microsoft.
- diseñado para ejecutarse especialmente sobre IIS (Internet Information Server)
- actualmente este lenguaje ha dejado paso a la versión ASP.net
- lenguaje interpretado

Perl

- inicialmente se utilizó para manipular cadenas de caracteres.
- primeros lenguajes para la programación web.
- código abierto
- basado en programación estructurada como C
- lenguaje interpretado

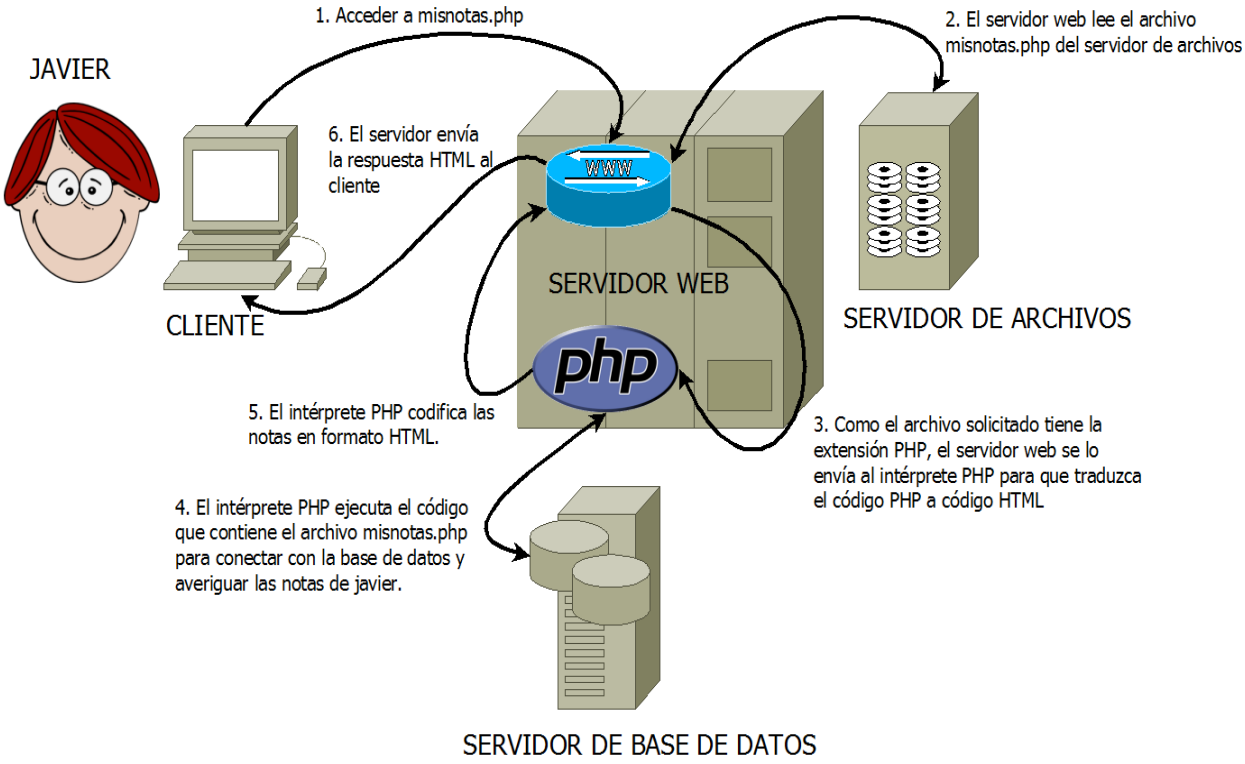
Python

- portable en diferentes plataformas y gratuito
- orientado a objetos
- lenguaje interpretado

JSP (Java Server Pages)

- porciones de código java intercalado con HTML estático
- el código java embebido se denomina servlet
- el servlet se carga en la memoria del servidor web al ser ejecutado la primera vez, para mejorar su ejecución en posteriores llamadas a dicha página web.

El proceso que se realiza a la hora de visitar una página PHP sería:



6.2 Aplicaciones CGI y derivados

CGI (Common Gateway Interface) es la forma más simple de crear páginas web dinámicas.

Un programa independiente al servidor web devuelve como resultado el contenido que debe visualizar el cliente (la página web resultante) a partir de ciertos parámetros de entrada.

La ubicación del programa a ejecutar se indica en la URL que forma la petición HTTP del cliente.

6.3 Aplicaciones híbridas

Tecnología intermedia entre los lenguajes de scripting y los programas CGI. Plataforma de Microsoft.Net Framework con el lenguaje ASP.Net.

ASP.Net es una tecnología totalmente orientada a objetos, que permite la creación de páginas web dinámicas utilizando lenguajes como VB.Net, C#, Jscript.Net.

Las páginas ASP.Net están contenidas en ficheros.ASPX que son los que el cliente solicita a través de una URL al servidor.

7. PARTES DE UNA APLICACIÓN WEB

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! y otras muchas 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.

8. HERRAMIENTAS DE PROGRAMACIÓN

A la hora de ponerte a programar una aplicación web, debes tener en cuenta con que herramientas cuentas que te puedan ayudar de una forma u otra a realizar el trabajo. Además de las herramientas que se tengan que utilizar en el servidor una vez que la aplicación se ejecute, como por ejemplo el servidor de aplicaciones o el gestor de bases de datos, de las que ya conoces su objetivo, existen otras que resultan de gran ayuda en el proceso previo, en el desarrollo de la aplicación.

Desde hace tiempo, existen entornos integrados de desarrollo (IDE) que agrupan en un único programa muchas de estas herramientas. Algunos de estos entornos de desarrollo son específicos de una plataforma o de un lenguaje, como sucede por ejemplo con Visual Studio, el IDE de Microsoft para desarrollar aplicaciones en lenguaje C# o Visual Basic para la plataforma .Net. Otros como Eclipse o NetBeans te permiten personalizar el entorno para trabajar con diferentes lenguajes y plataformas, como Java EE o PHP.
No es imprescindible utilizar un IDE para programar. En muchas ocasiones puedes echar mano de un
simple editor de texto para editar el código que necesites. Sin embargo, si tu objetivo es desarrollar una aplicación web, las características que te aporta un entorno de desarrollo son muy convenientes. Entre estas características se encuentran:

- Resaltado de texto. Muestra con distinto color o tipo de letra los diferentes elementos

del lenguaje: sentencias, variables, comentarios, etc. También genera indentado automático para diferenciar de forma clara los distintos bloques de un programa.
- Completado automático. Detecta qué estás escribiendo y cuando es posible te muestra distintas opciones para completar el texto.
- Navegación en el código. Permite buscar de forma sencilla elementos dentro del texto, por ejemplo, definiciones de variables.
- Comprobación de errores al editar. Reconoce la sintaxis del lenguaje y revisa el código en busca de errores mientras lo escribes.
- Generación automática de código. Ciertas estructuras, como la que se utiliza para las clases, se repiten varias veces en un programa. La generación automática de código puede encargarse de crear la estructura básica, para que sólo tengas que rellenarla.
- Ejecución y depuración. Esta característica es una de las más útiles. El IDE se puede encargar de ejecutar un programa para poder probar su funcionamiento. Además, cuando algo no funciona, te permite depurarlo con herramientas como la ejecución paso a paso, el establecimiento de puntos de ruptura o la inspección del valor que almacenan las variables.
- Gestión de versiones. En conjunción con un sistema de control de versiones, el entorno de desarrollo te puede ayudar a guardar copias del estado del proyecto a lo largo del tiempo, para que si es necesario puedas revertir los cambios realizados.

Los dos IDE de código abierto más utilizados en la actualidad son Eclipse y NetBeans. Ambos permiten el desarrollo de aplicaciones informáticas en varios lenguajes de programación. Aunque en sus orígenes se centraron en la programación en lenguaje Java, hoy en día admiten directamente o a través de módulos, varios lenguajes entre los que se incluyen C, C++, PHP, Python y Ruby.

9. BIBLIOGRAFÍA.

1. López, M.; Vara, JM; Verde, J.; Sánchez, D.M.; Jiménez, J.J.; Castro, V. Desarrollo web en entorno servidor. 2012. RA-MA, Madrid