UNIDAD DIDÁCTICA 0:
INTRODUCCION A LAS ARQUITECTURAS DE DESARROLLO EN ENTORNO SERVIDOR

Módulo profesional:
Desarrollo Web en Entorno Servidor

Duración: 160 Horas.
Código: 0613

Ciclo Formativo de Grado Superior:
Desarrollo Aplicaciones Web

Contenido

Resumen Introductorio

En esta unidad nos situaremos dentro del mundo del desarrollo de aplicaciones web, una visión general del funcionamiento de Internet, así como diferentes clasificaciones de los lenguajes de programación.

Esta visión es importante para tener un punto de partida base y comprender la terminología que vamos a comenzar a utilizar.

Introducción

Internet ha evolucionado enormemente desde que surgió como un proyecto muy particular en el gobierno de defensa de los Estados Unidos. Esta evolución en la arquitectura, en los sistemas y también en los hábitos de consumo de los usuarios ha hecho evolucionar los diferentes lenguajes de programación utilizados para desarrollar aplicaciones web.

Nos encontramos por lo tanto que para el funcionamiento de una aplicación debemos manejar diferentes lenguajes, muy diferente a la forma de programar para desktop o para móviles. Esta diversidad hace que por un lado tengamos alternativas y flexibilidad, pero por otro que sea más complejo tanto durante la programación como en la integración.

En este módulo se utilizará PHP para mostrar el uso de un lenguaje para la programación en la parte de servidor, lenguaje muy maduro y con una gran comunidad de programadores que hace que sea fácil encontrar ejemplos, código y sobre todo ayuda y respuestas a posibles problemas.

Caso Introductorio

Un médico a través del ordenador de su consulta solicita la historia médica del paciente que acaba de atender. Una vez abierta la página introducirá la prescripción de la medicación dada y la información de la visita quedando almacenada cuando aprieta el botón “Salvar”.

Cuando finalices esta introducción tendrás una visión general de cómo ha sido esa petición y qué lenguajes han interactuado en cada uno de los momentos.

Desarrollo:

1. Funcionamiento de Internet

Antes de comenzar a programar, incluso antes de comenzar a conocer el funcionamiento de los sistemas basados en cliente y servidor, necesitamos tener un "backend" mínimo de conocimiento en lo que hoy en día denominamos Internet.

Nuestra misión en estos inicios no tiene como pretensión conocer al mínimo detalle los protocolos, los estándares y la forma de funcionamiento detallada de Internet, sino tener una visión general de lo que ocurre cuando nosotros utilizamos un navegador (Chrome, Firefox, Safari, …) y escribimos una dirección de Internet en el mismo

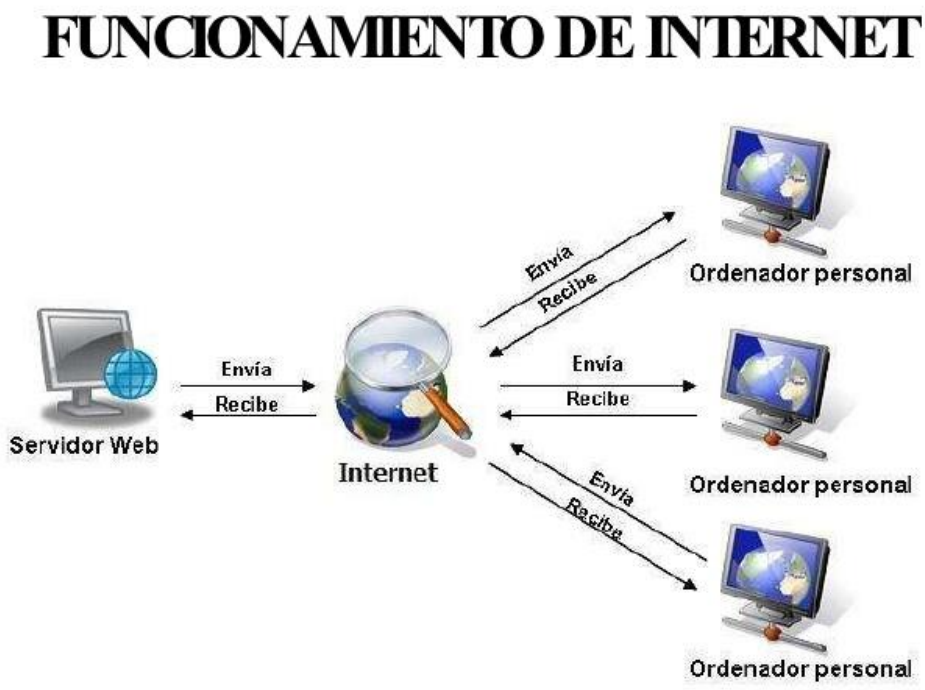


Imagen 1: Funcionamiento de Internet

Fuente de la imagen:

http://internet-aam.blogspot.com.es/2015/11/funcionamiento-del-internet-internet.html

La imagen anterior es muy clarificadora del funcionamiento de Internet, pero vamos a describir los pasos que se están produciendo y de esa forma también introduciremos algunos tecnicismos que enriquecerán también nuestro vocabularios:

0 En primer lugar el usuario abre un navegador y escribe en la URL (Uniform Resource Locator) http://www.cookingsoftware.es

1 El navegador descompone dicha URL en varias partes, como el protocolo que va a ser el HTTP, el servidor, el puerto y la ruta o recurso. En nuestro caso el servidor que está solicitando el usuario es www.cookingsoftware.es

2 Existe otro protocolo que convertirá esta petición a una IP, puesto que todos los dispositivos con conexión a Internet tienen una identificación única (no es del todo así, pero podemos ahora mismo explicarlo así).

3 Ese servidor, tendrá instalado un sistema operativo y un servidor que recogerá la petición del cliente, la sabrá entender y analizar y devolverá una respuesta.

4 Por último esta respuesta recorre el camino inverso hasta llegar al usuario, al navegador, el cual mostrará el resultado

En los siguientes enlaces encontrarás las recomendaciones del grupo de trabajo IETF 39/2006, el cual regula la normativa de Internet y la formación de las URLs

RFC 1630, RFC 1736, RFC 1737, RFC 1738, RFC 1808, RFC 2396,RFC 2732, RFC 3986

Lo que hemos explicado de forma rápida, es un conjunto de tecnologías, protocolos, servidores y softwares, que en sí constituyen un ámbito de estudio, y que en gran medida se estudiará en otros módulos.

En nuestro caso, en nuestro módulo, nos quedamos con el paso 4, ya que es en este punto y en este lugar donde nuestro código será el encargado de recoger, interpretar la información que el usuario nos solicita para devolver otra información, lo que comúnmente denominamos "página web", y que el navegador puede presentar por pantalla.

1.1. Protocolos

Uno de los retos para ser un desarrollador profesional es la de adquirir una serie de conocimientos a nivel de programación pero también a nivel del lenguaje. Ha surgido en el anterior apartado la palabra protocolo cuando hablábamos de cómo funciona Internet.

Según la norma del grupo de trabajo referente en Internet (IETF) (a partir de ahora referente para cualquier concepto referente a Internet), un protocolo de comunicaciones es un sistema de reglas que permiten que **dos o más entidades** de un sistema de comunicación se comuniquen entre ellas para **transmitir información** por medio de cualquier tipo de variación de una magnitud física. Se trata de las reglas o el estándar que define la sintaxis, semántica y sincronización de la comunicación, así como también los posibles métodos de recuperación de errores. Los protocolos pueden ser implementados por hardware, por software, o por una combinación de ambos.

En el siguiente enlace encontrarás las recomendaciones del grupo de trabajo IETF 791, el cual regula el *Internet Protocol*. (Defense Advanced Research Projects Agency, DARPA, 1981)

https://tools.ietf.org/html/rfc791

Un protocolo, al fin y al cabo, podríamos entenderlo como ese idioma que aprendemos, con sus reglas y con su léxico, y que nos permite comunicarnos con diferentes personas.

Durante todo el curso, aparecerá en más de una ocasión esta palabra, puesto que es la base para saber cómo nuestro programa va a funcionar correctamente o encontrar el error producido.

1.2. Protocolo HTTP

Dentro de los protocolos existentes que utilizamos día a día sin saberlo para realizar transacciones dentro de Internet, el más importante para nosotros como desarrolladores de aplicaciones Web es el protocolo HTTP.

Según Ilya Grigorik (Grigorik, 2013) en su libro "High Performance Browser Networking: What every web developer should know about networking and web performance", el protocolo de transferencia de hipertexto (HTTP) es uno de los protocolos de aplicación más omnipresentes y ampliamente adoptados en Internet: es el lenguaje común entre clientes y servidores, lo que permite la web moderna. Es el protocolo de comunicación que permite las transferencias de información en la World Wide Web.

En el siguiente enlace encontrarás más información sobre el protocolo HTTP
https://hpbn.co/brief-history-of-http/

Este protocolo, como muchos protocolos y mecanismos que intervienen en la interacción dentro de Internet es un protocolo de tipo Cliente-Servidor (concepto que también introduciremos en esta unidad) y que con una sintaxis pequeña permite comunicar a un navegador con un servidor de aplicaciones web para devolver resultados como las conocidas "páginas web".

Una de las ventajas o desventajas, dependiendo del punto de vista que estemos trabajando, es que el HTTP como bien indica su definición (Text), es un protocolo textual que nos permite visualizar tanto lo que el navegador está enviando o como la respuesta del servidor (lo veremos más adelante

con muchos ejemplos). Por lo tanto no hará falta ningún desencriptador ni ningún mecanismo adicional para que cualquiera, que sepa de la sintaxis de HTTP, pueda interpretar qué está ocurriendo con el mismo.

Si por el contrario, estamos analizando el protocolo desde el punto de vista de la seguridad, el protocolo es **muy inseguro**, ya que cualquiera puede visualizar y capturar los mensajes que estamos transmitiendo entre nuestra máquina y la remota. Imaginemos que estamos realizando una compra online, y como proceso habitual de esa compra nos soliciten introducir la tarjeta de crédito, esa número de tarjeta, que no es ni más ni menos que un campo tipo texto sería visible para cualquier atacante que interceptase la comunicación. Es en este punto donde se utiliza el protocolo HTTPS, protocolo que no vamos a explicar en detalle, y que proporciona un recubrimiento **seguro** de nuestras comunicaciones. En el mismo ejemplo que antes planteábamos en el momento de enviar nuestro número de tarjeta, ese mismo texto iría codificado y gracias al cifrado aplicado sería indescifrable.

En el siguiente enlace encontrarás las recomendaciones del grupo de trabajo IETF 2660, el cual regula el *HTTPS*. (Rescorla, RTFM, Schiffman, & Terisa Systems, 1999)

https://tools.ietf.org/html/rfc2660

1.3. Cliente-Servidor

En apartados anteriores hemos nombrado el concepto "cliente-servidor", vamos a desarrollarlo un poco más en este apartado. Comencemos con una imagen muy representativa del sistema:

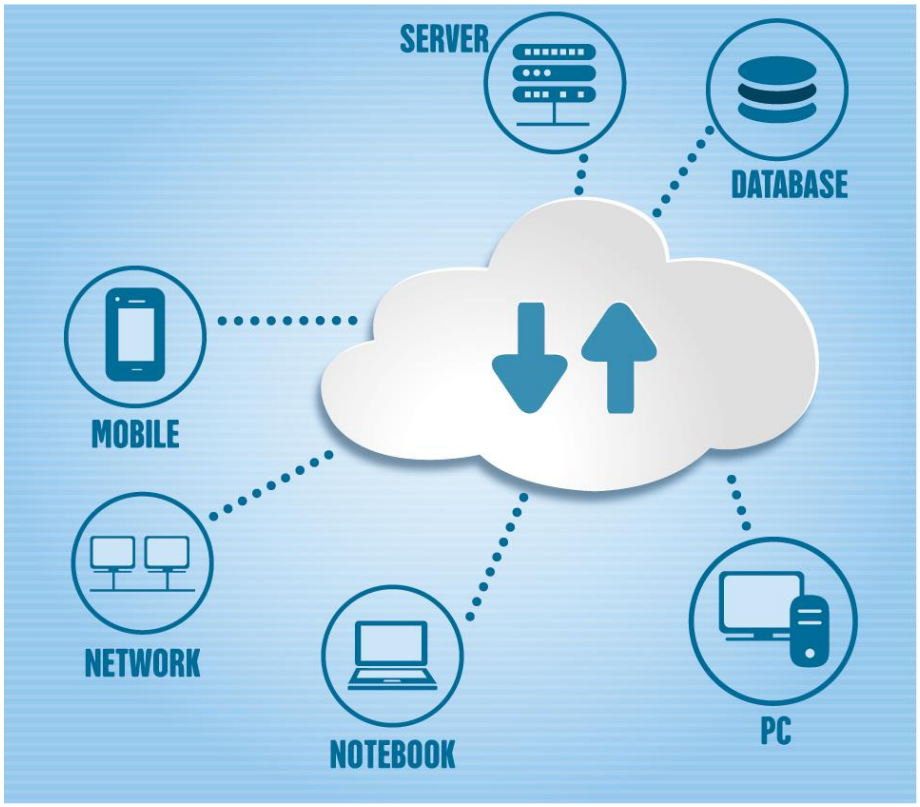


Imagen 2: Cliente-Servidor

Fuente de la imagen:
https://www.vexels.com/vectors/preview/72547/remote-server-infographics

¿Qué observamos en la imagen? Por un lado tenemos elementos que están realizando consultas, pidiendo información, visitando páginas, también todos ellos generando tráfico y nueva información. Son los denominados clientes dentro de este binomio que estamos analizando. En el desarrollo web, en el mundo de aplicaciones web, los clientes actualmente han dejado de ser elementos fijos, un *desktop* por ejemplo, como se puede observar en la imagen ahora mismo existen múltiples orígenes que actúan como clientes, un móvil, una tableta, un portátil, un coche, un reloj, …

En el otro lado tenemos los servidores. Estos reciben las peticiones y/o demandas de los clientes, y dependiendo de diferentes variables, como quien solicita la pregunta, desde dónde la solicita, el navegador que está

utilizando u otros parámetros es capaz de procesarla y devolver una respuesta. Las respuestas a las que estamos acostumbrados en un entorno web, son las páginas web, pero hoy en día, cualquier aplicación de móvil está interactuando con un servidor para pedirle datos, sólo datos, que la aplicación transformará en la aplicación para presentarlos de una forma atractiva y útil.

2. Lenguajes de programación

A estas alturas, y más teniendo en cuenta que estamos en un 2º Curso de un Ciclo Grado Superior, saber qué es un lenguaje de programación e incluso conocer diferentes lenguajes de programación es un prerrequisito para alcanzar este módulo. Pero no está de más realizar una breve introducción de los diferentes tipos de lenguajes para acabar aterrizando en los lenguajes específicos para el desarrollo en el ámbito de aplicaciones web.

2.1. Lenguaje compilado frente interpretado

Esta es una de las primeras y principales distinciones que debemos realizar cuando clasificamos los lenguajes de programación.

Un lenguaje de programación se dice que es compilado cuando existe un traductor que recoge el programa/aplicación realizada por un programador y lo **compila** para un determinado sistema operativo (o arquitectura de forma más exacta), es decir lo convierte en un ejecutable que un sistema operativo (arquitectura) entiende y puede ejecutarlo.

Un claro ejemplo es la programación es **C#**, donde cada vez que queremos comprobar o cuando hemos desarrollado nuestro código finalizado, realizamos la acción de compilar nuestro código, si seguimos con nuestro ejemplo y nos encontramos en un sistema Windows, se generará un código .exe que podrá ser ejecutado por la misma máquina que lo generó o por cualquier otra con un sistema operativo Windows compatible con la versión y el número de bits de la arquitectura.

Otro ejemplo de lenguaje compilado es **Swift**, el cual deberá ser utilizado en un sistema Mac para que una vez compilado sólo pueda ser ejecutado en una máquina/dispositivo con iOs instalado.

En el siguiente enlace encontrarás referencia completa del lenguaje Swift dada por el fabricante Apple:

https://www.apple.com/es/swift/

El caso del lenguaje **Java**, como bien conocéis, es un caso especial, ya que se encuentra en medio de lo que es un lenguaje compilado y uno interpretado (que veremos a continuación). Java no realiza la compilación pura como lo realizan el resto de lenguajes compilados, sino que realiza un paso intermedio que consiste

En la siguiente imagen podemos ver la secuencia que tendríamos con Java, en el que el compilador traduce el código de alto nivel a un código intermedio denominado **bytecodes**. Este código intermedio es posible distribuirlo a cualquier plataforma, pero para poder ser ejecutado, el ordenador destino deberá tener instalado un ejecutor denominado Java Virtual Machine, el cual sí que es dependiente del sistema operativo y por lo tanto del hardware.

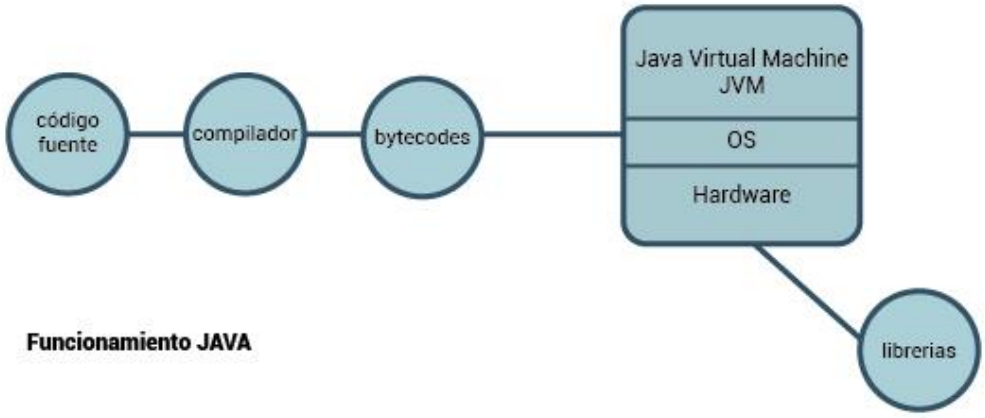


Imagen 3: Funcionamiento Java

Fuente de la imagen: propia

Los lenguajes interpretados, tal y como indica la palabra, son lenguajes de programación que necesitan de un **intérprete** para ser ejecutados. Es decir, no se compilan como C# o Java, y por lo tanto no son dependientes de la máquina y del sistema operativos como los lenguajes compilados.

Pertenecen a esta categoría casi todos los lenguajes que utilizamos en desarrollo de aplicaciones Web. Ejemplos de estos lenguajes son el lenguaje PHP, JavaScript, Ruby, HTML, ...

A partir de ambas definiciones ya podemos hacernos una idea de cuáles son las diferencias, ventajas y desventajas:

- Los lenguajes interpretados son multiplataforma ya que será el intérprete quien se adapte al sistema operativo y hardware final. Como hemos explicado anteriormente, Java es un lenguaje intermedio, aunque se aproxima más a un lenguaje compilado ya que realiza una compilación intermedia del código de alto nivel que los lenguajes interpretados no realizan.
- Los lenguajes interpretados a priori tienen un peor rendimiento que los compilados, ya que estos últimos se encuentran optimizados para el hardware final.

2.2. Lenguajes cliente frente lenguajes servidor

Ya centrados en el desarrollo de aplicaciones web, podemos clasificar los lenguajes de programación dependiendo de dónde se vayan a ejecutar.

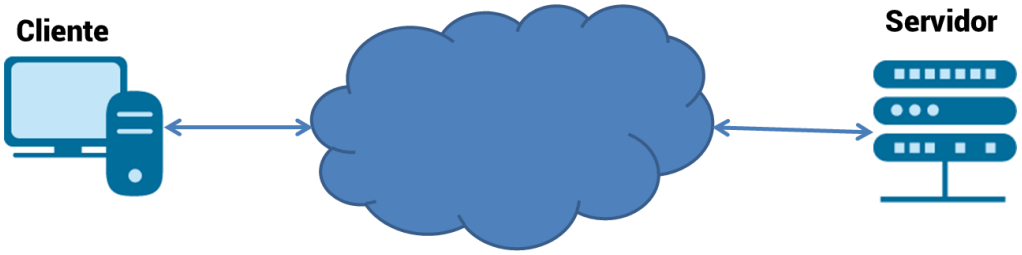


Imagen 4: Lenguajes clientes-servidor

Fuente de la imagen: propia

Hemos utilizado la imagen anterior que explicábamos la arquitectura cliente-servidor para aislar los dos elementos que vamos a tratar en este punto. E incluso podríamos aterrizar mejor la explicación mediante un ejemplo propio dentro del modelo web:

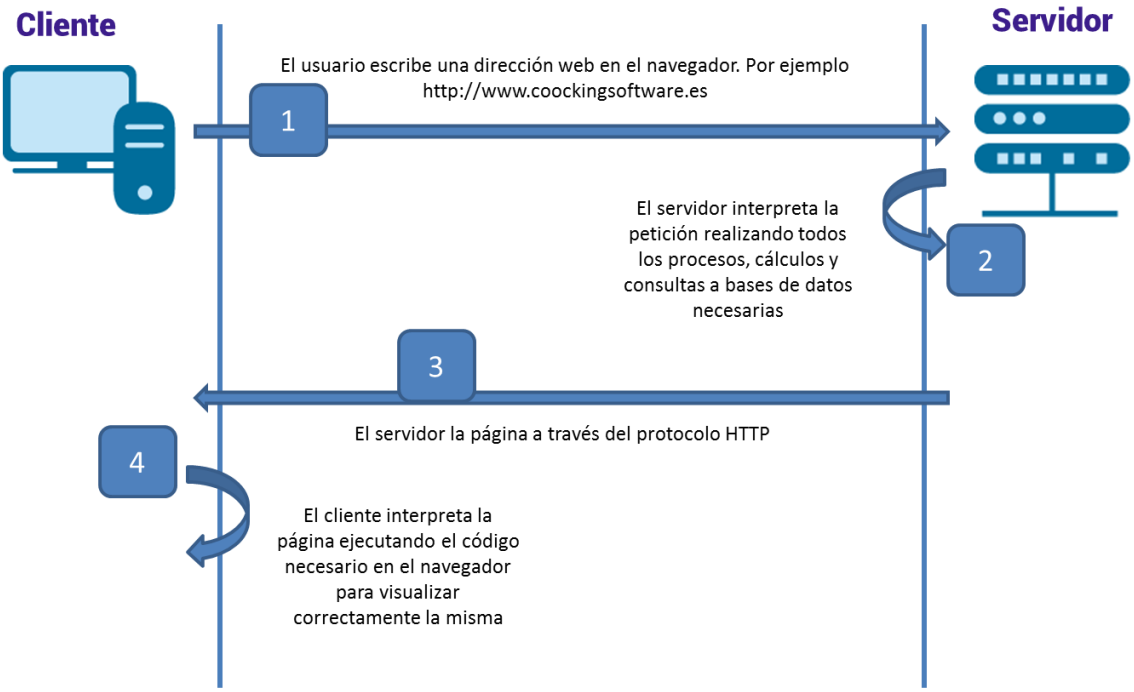


Imagen 5: Petición de página web

Fuente de la imagen: propia

Como podéis observar, un navegador cliente realiza una petición, es decir, escribe en el navegador una dirección web. Este navegador lanza la petición a través de Internet, y mediante diferentes mecanismos y protocolos que no vienen al caso alcanzará el servidor remoto. En este caso la petición llegará a un servidor que realizará una serie de acciones, tales como realizar consultas contra la base de datos, comprobar la seguridad, almacenar peticiones, conectarse con otros servicios, etc. Una vez que el servidor ha realizado todas estas acciones, devolverá la contestación por el protocolo HTTP hacia el mismo cliente que había realizado la petición. Una vez que la contestación llegue al cliente, será el momento en el que el navegador realizará la interpretación de la respuesta, colocando los elementos en pantalla de acuerdo al dispositivo, programando eventos, e incluso utilizando recursos locales para diversas acciones.

Es en este momento, y una vez que hemos comprendido el anterior esquema cuando podemos añadir los lenguajes de programación allí donde se ejecutan:

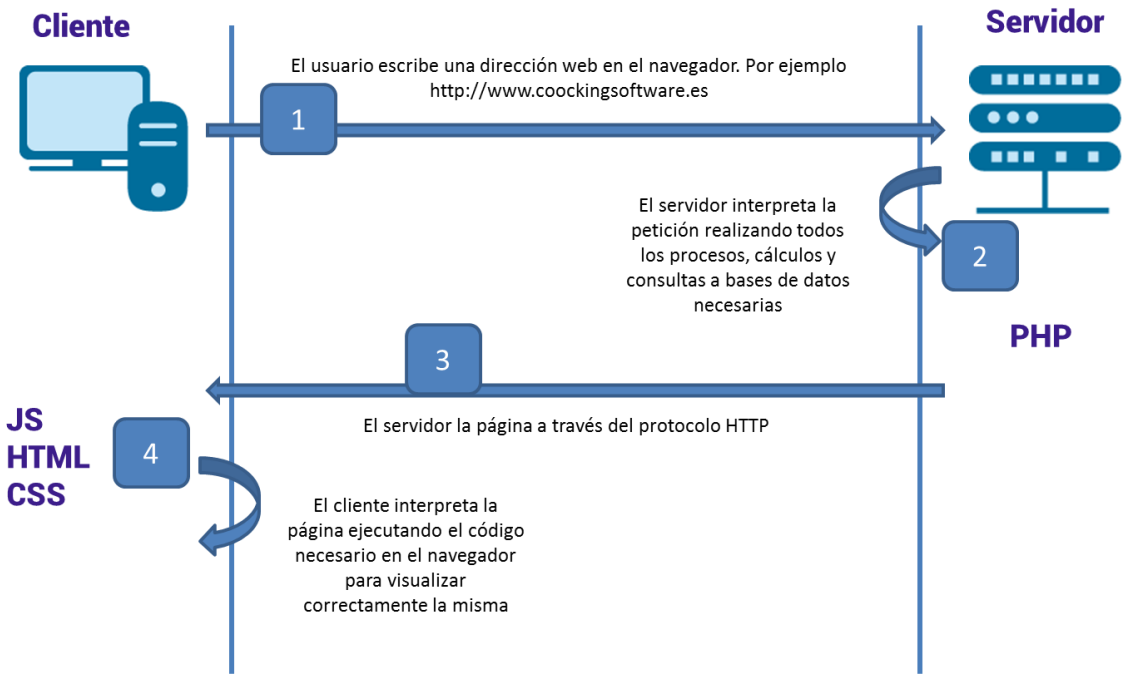
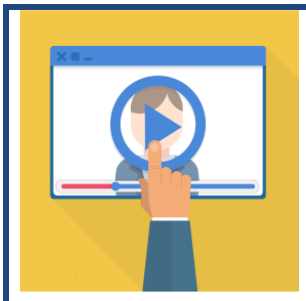


Imagen 6: Lenguajes de programación

Fuente de la imagen: propia

En la anterior imagen, hemos sumado un ejemplo de posibles lenguajes, por un lado tenemos el lenguaje PHP, el cual vamos a trabajar en este módulo, y por otro lado tenemos los lenguajes JavaScript, HTML y CSS:

- Un lenguaje podemos denominarlo lenguaje de servidor cuando este se ejecuta en la parte del servidor, valga la redundancia. Es capaz de realizar acciones sobre seguridad y sobre datos (principalmente) que de otra manera sería imposible realizar de forma segura.
- Un lenguaje lo denominamos de cliente cuando se ejecuta en la parte del navegador que lanza la petición, es decir en la parte de cliente. El lenguaje de programación que tenemos en el anterior ejemplo es JavaScript.



En el vídeo que encontrarás en el siguiente enlace podrás visualizar un ejemplo explicativo del funcionamiento cliente-servidor

https://youtu.be/xc5DoOqT0vI

En este punto del curso es complicado explicar el funcionamiento y el porqué de cada uno de ellos, pero es importante al menos tener una visión general como muestra la anterior imagen.

Caso práctico 1
Título: 20 lenguajes más usados

Foro 1
Título: lenguajes de programación en servidor

3. PHP

Como en el anterior apartado hemos nombrado, el lenguaje que vamos a utilizar este año en la parte de servidor es el lenguaje PHP, un lenguaje muy muy maduro y que a pesar de no tener buenas críticas, cuenta con una comunidad amplísima de desarrolladores.

Según la documentación de fabricante, PHP, acrónimo de "PHP: Hypertext Preprocessor", es un lenguaje de 'scripting' de propósito general y de código abierto que está especialmente pensado para el desarrollo web y que puede ser embebido en páginas HTML. Su sintaxis recurre a C, Java y Perl, siendo así sencillo de aprender. El objetivo principal de este lenguaje es permitir a los desarrolladores web escribir dinámica y rápidamente páginas web generadas; aunque se puede hacer mucho más con PHP. Fue creado originalmente por Rasmus Lerdorf en el año 1995 como solución a una problemática que encontró en el análisis de información a partir de código C.

En el siguiente enlace encontrarás un podcast fantástico sobre el PHP: http://wedevelopers.com/2015/12/23/we-developers-043-php/

Ebook recomendado (Hudson, 2015)
http://www.hackingwithphp.com/

Caso práctico 2
Título: PHP v5.6 vs 7

Foro 2
Título: PHP v5.6 vs 7

3.1. Versiones

PHP ha sido un lenguaje de programación que se ha caracterizado por la estabilidad de sus versiones, al contrario de muchos lenguajes actuales donde el cambio de versión se puede producir en menos de 12 meses. La segunda característica podemos destacar su intento de "compatibilidad hacia atrás" que ha provocado en muchas ocasiones no avanzar en la mejora del lenguaje.

Hasta su versión 4, y con la aparición de Zend, el lenguaje se construía de una forma más o menos caótica. Será a partir de esta versión donde el lenguaje alcanza otra dimensión más profesional.

Será en la versión 5 donde el lenguaje incorpora orientación a objetos y será entonces cuando se comienza a pensar en soluciones más grandes, flexibles y estructuradas. Será entonces cuando muchos frameworks comienzan a surgir teniendo como referencia el lenguaje de programación PHP.

La versión 6 nunca llegó a utilizarse realmente, ya que el desarrollo sobre la problemática Unicode resultó más compleja de lo que en un principio se planificó. Por lo que PHP pasó directamente a su versión 7 que es la que actualmente utilizamos.

En el siguiente artículo de uno de los blogs más reconocidos sobre desarrollo, encontrarás información sobre la versión 7
https://www.genbetadev.com/actualidad/php-7-ya-esta-aqui

3.2 Frameworks

Uno de los grandes éxitos actuales de los lenguajes de programación son los frameworks de desarrollos que nos permiten configurar determinados problemas ya que están ya desarrollados, como es el trabajo con Bases de Datos, la Seguridad, los Usuarios.

Quizá el FrameWork más famoso e importante en PHP es Symfony, a partir de éste se realizó Laravel, pero hay muchísimos frameworks MVC (Modelo Vista Controlador), CakePHP, Zend, ...

3.3 Aplicaciones

También son muchas las aplicaciones basadas en PHP, incluso la todopoderosa Facebook que basó su desarrollo en PHP (Hack). Una de las más famosas aplicaciones puede ser WordPress que es actualmente el CMS más utilizado en el mundo.

Resumen final:

Internet es el espacio de funcionamiento de un sistema de elementos, arquitecturas, protocolos y lenguajes de programación que hacen posible que una petición por parte de un cliente, de un navegador, llegue hasta un servidor que es capaz de comprender esta petición, devolviendo una respuesta hacia ese mismo cliente que lanzó la petición.

En la arquitectura cliente-servidor, tenemos elementos que están realizando consultas, pidiendo información, visitando páginas, también todos ellos generando tráfico y nueva información. Son los denominados clientes. En el otro lado tenemos los servidores. Estos reciben las peticiones y/o demandas de los clientes, y dependiendo de diferentes variables, como quien solicita la pregunta, desde dónde la solicita, el navegador que está utilizando u otros parámetros es capaz de procesarla y devolver una respuesta.

Centrados en el desarrollo de aplicaciones web, podemos clasificar los lenguajes de programación dependiendo de dónde se vayan a ejecutar según la anterior arquitectura (cliente o servidor), y en concreto en la parte

de servidor nos encontramos con una serie de leguajes que permiten realizar las tareas antes descritas.

PHP es un lenguaje de programación de uso general de código del lado del servidor originalmente diseñado para el desarrollo web de contenido dinámico, que a partir de su versión 5 incorpora en su totalidad la programación Orientada a Objetos y por lo tanto va a ser el lenguaje utilizado por nosotros para desarrollar nuestras aplicaciones de servidor.