

UT1 – SELECCION DE ARQUITECTURAS Y HERRAMIENTAS DE PROGRAMACION

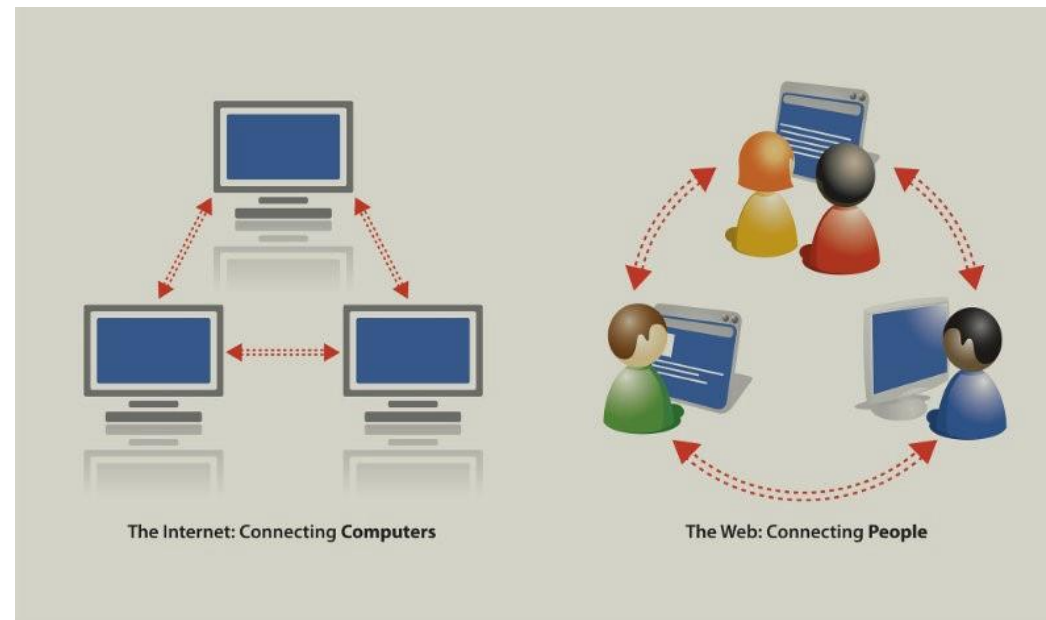
Desarrollo Web en Entorno Servidor
Curso 2022-23



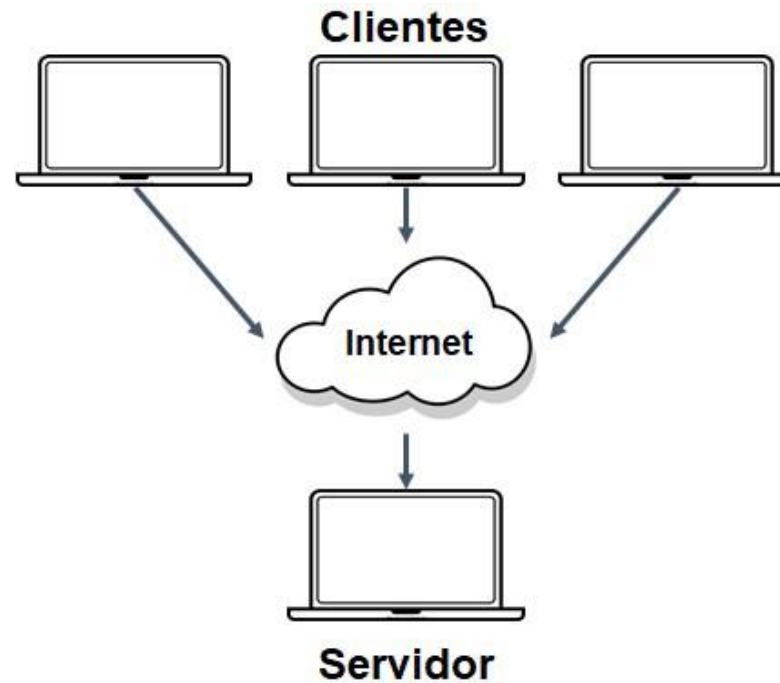
Objetivos

1. Conocer el camino de una petición web y su posterior respuesta, desde que el usuario introduce una dirección web hasta que visualiza el contenido.
2. Conocer la arquitectura cliente-servidor.
3. Diferenciar de forma clara que es un cliente y que es un servidor.
4. Conocer los distintos modelos de ejecución de un servidor web.
5. Conocer los principales lenguajes de desarrollo que son ejecutados en el servidor y las principales características de los mismos.
6. Entender como el servidor recibe la petición, la procesa y sirve el contenido.
7. Ser capaz de escoger una buena tecnología para resolver un problema concreto dentro del ámbito del desarrollo web.
8. Conocer las herramientas necesarias para poder programar en entornos web de forma eficiente.

- **Internet** es un conjunto de dispositivos digitales, conectados entre sí para compartir información a través de un medio físico.
- La **WWW** (la world wide web , o red mundial) es el sistema de distribución de documentos de hipertexto o hipermedia a través de internet.



- En una arquitectura cliente-servidor, un **servidor** es el encargado de mandar la información a los denominados **clientes** (los que la consumen) en base a unos parámetros que los clientes utilizan para determinar exactamente qué parte de la información requieren

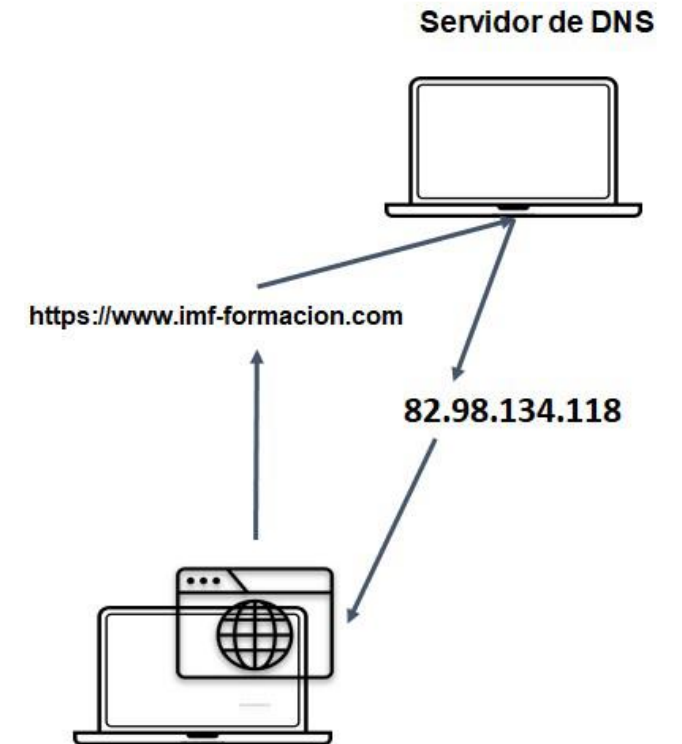


Protocolos WWW

- En la WWW se utilizan protocolos, más concretamente dos: el **HTTP** y el **HTTPS**.
- La diferencia entre ellos está únicamente en la encriptación de los segundos frente a los primeros.
 - Si la información va encriptada a través de internet, aunque sea interceptada por un agente externo a la comunicación, no puede ser interpretada de forma correcta y es por ello que la denominamos segura.
- Hasta hace poco tiempo, el protocolo HTTPS solo se utilizaba para comunicaciones críticas como procesos de compra o accesos de usuarios.
- Tras la aparición de organizaciones como Let's Encrypt (que permiten una conexión segura de forma gratuita, mientras que antes era de pago) este tipo de comunicaciones han proliferado mucho.

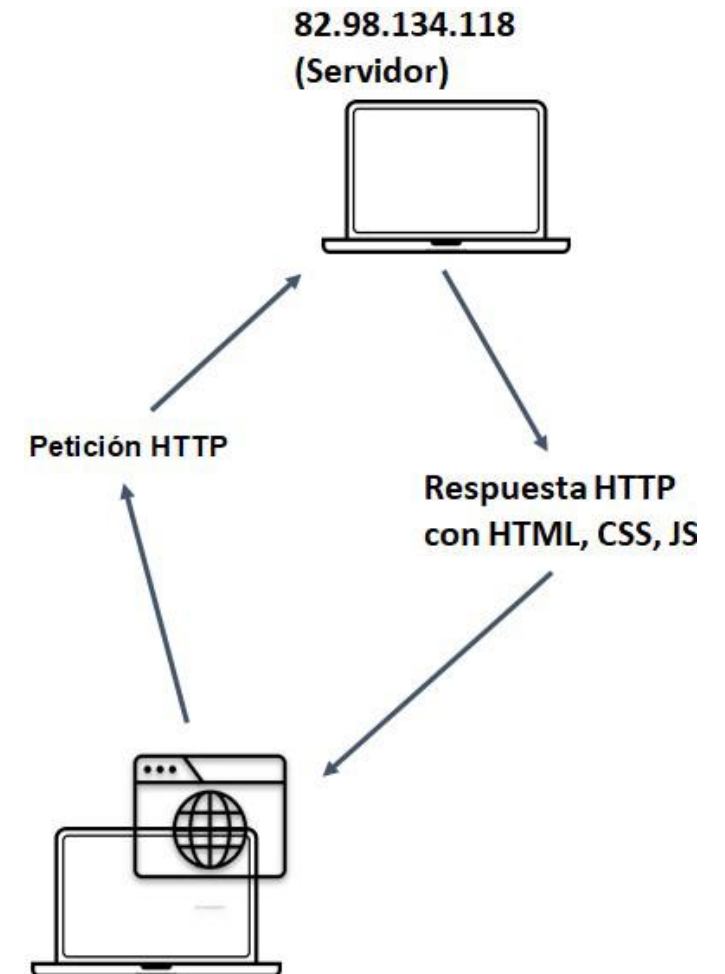
- Para que la petición llegue correctamente del servidor al cliente, en primer lugar, hay que localizar al servidor.
- Este servidor no es más que un ordenador, más o menos potente, conectado a internet.
- Para localizar ese servidor de entre los millones de dispositivos conectados a la red de redes se crearon las direcciones IP.
- Una dirección IP es un número que se asigna a cada dispositivo conectado a internet.
- Cada router del mundo tiene una de forma única durante el tiempo que permanece conectado, pudiendo variar si se desconecta de la red.
- Los servidores tienen siempre la misma, dado que los operadores telefónicos lo ofrecen como servicio.

- Para poder localizar la IP del servidor que se está buscando, en base a su dominio se utilizan los servidores de DNS.
- Un servidor DNS es un directorio que contiene un listado de todos los dominios y sus IP asociadas.
- En la figura de la derecha se puede observar el flujo de información entre el cliente y el servidor DNS, que permite al navegador localizar el servidor de manera inequívoca dentro de los millones de dispositivos conectados a internet.



- Una vez el navegador ha localizado el servidor contra el que se quiere lanzar la petición:
 - Se le envía una petición a través de un protocolo conocido (HTTP o HTTPS en el caso de peticiones web).
 - El servidor mandará una respuesta.
 - La petición HTTP contiene, además del dominio, la información necesaria para ser procesada correctamente por el servidor:
 - La ruta del recurso a servir,
 - Datos del dispositivo que realiza la petición,
 - Datos de sesión (si es que se están utilizando cookies para mantener sesiones de usuario)
 - Otros datos.

- Dicha petición llega al servidor que la procesa según las reglas que tenga estipuladas en su configuración y la sirve al cliente.
- La respuesta puede ser un fichero de cualquier tipo.
- Si el objetivo es que un navegador lo interprete, tendrá que ser necesariamente un fichero HTML.
- Además de ficheros HTML, el servidor puede ofrecer otros tipos de archivos auxiliares que sean de utilidad para enriquecer el contenido HTML mostrado:
 - CSS (hojas de estilos)
 - JS (JavaScript)
 - Elementos multimedia (imágenes, vídeos, audios).
- El cliente, a través del navegador web, interpreta esta respuesta y muestra al usuario el contenido.



- Dentro de la comunicación cliente-servidor, como se ha visto en el apartado anterior, el cliente es el sistema que inicia la comunicación.
- En desarrollo web el cliente es habitualmente un navegador, pero no es así necesariamente. Se determina cliente a todo aquel que realiza peticiones al servidor (entendiendo peticiones legítimas y esperadas, obviando ataques informáticos, inspecciones de seguridad, etc.).
- Como se ha explicado en el apartado anterior, un dispositivo móvil o cualquier otro elemento conectado a internet puede actuar como cliente en un momento dado.
- Por ejemplo, en una aplicación móvil, lo usual es que consulte al servidor información de una base de datos (por ejemplo, el listado de pedidos de un usuario de una tienda) y el móvil recibe la información en formato JSON o XML y la muestra de la forma que le sea mejor.
- La gran diferencia entre el HTML y los otros formatos citados, es que el HTML está pensado para mostrar información, no para procesarla, es decir, contiene elementos relacionados con la semántica y los estilos. El JSON (y el XML), sin embargo, únicamente transportan información, dejando de lado del cliente toda la responsabilidad.

- En el primer ejemplo se puede observar un bloque de código HTML que podría interpretar el navegador. En este caso se tendría un título, un cuerpo formateado como un párrafo y una imagen lista para ser mostrada.
- En el segundo ejemplo se puede observar una respuesta JSON, donde claramente se ve la falta de elementos de estilos. Esta información está pensada para que se procese y se utilice en un cliente y no para ser mostrada directamente al usuario.

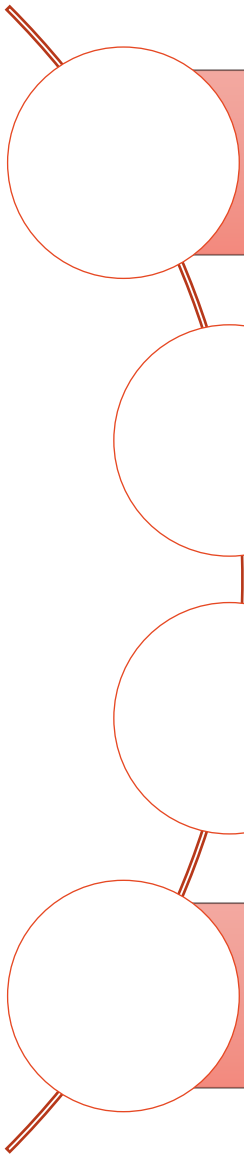
```
<html>
  <body>
    <article class="main">
      <h1>Título</h1>
      
      <p>Cuerpo del artículo, párrafo de ejemplo.</p>
    </article>
  </body>
</html>
```

```
[
  {
    "title": "Título",
    "body": "Cuerpo de la noticia",
    "image": {
      source: 'http://example.com/image.jpg',
      alt: "Texto alternativo de la imagen"
    }
  }
]
```

- Este módulo se centra en los **navegadores web** como clientes (hasta llegar al apartado de servicios web), por lo que no se profundizará más en JSON y XML y sus características especiales.
- Lo más importante de un navegador web es conocer con profundidad qué parte del trabajo realiza él y qué parte el servidor.
- El navegador web es el encargado de renderizar HTML, interpretar y aplicar los estilos descritos por el CSS y ejecutar el código JavaScript.
- Es importante tener claro que **el navegador web nunca podrá ejecutar** PHP, Python o cualquier otro lenguaje de servidor, de la misma forma que un servidor no podrá interpretar HTML o CSS, ni ejecutar JavaScript (salvando alguna excepción como Node.js).

- El servidor es el dispositivo encargado de **mandar la información requerida** por los clientes. Esta se puede pasar en formatos preparados para ser vistos e interpretados por un navegador (HTML, CSS) o por algún otro programa (JSON, XML).
- Es importante tener claro que en el ámbito de hardware no tiene por qué ser un equipo distinto a un ordenador personal. **Cualquier ordenador, debidamente configurado, puede actuar como servidor.** No obstante, los servidores web que se utilizan de forma profesional, en la realidad, suelen tener más potencia que los ordenadores personales. Dependiendo de los requisitos del sistema utilizado y de la cantidad de tráfico que tenga que soportar se necesitarán servidores más o menos potentes.
- En el caso de que el sitio web tenga que realizar tareas muy pesadas, muchas tareas o soportar mucho tráfico, entonces se pueden utilizar varios servidores de forma simultánea. Esta estrategia (la de incrementar el número de servidores) se llama **escalado horizontal** (dado que todos los servidores sirven para lo mismo).

- Cuando la arquitectura de los servidores consta de muchas máquinas pequeñas, cada una contratada de forma individual, configurada de forma independiente y fácilmente escalable, se le llama **alojamiento en la nube o Cloud hosting**.
- Esta forma de arquitecturar sistemas se hizo popular a raíz de ofrecerlo Amazon como servicio de alojamiento. Actualmente, otras grandes compañías como Microsoft (con Microsoft Azure) o Google (con Google Cloud) tienen sistemas con prestaciones similares.
- Los servidores web, debido a sus requisitos especiales (gran ancho de banda, RAM alto, disponibilidad 24/7, etc.) no suelen tenerse en las oficinas o centros de datos de las empresas y lo habitual es alquilarlas a empresas especializadas.
- Atendiendo al modelo de alquiler que se utilice se pueden diferenciar cuatro grandes grupos de hostings.



Hosting compartido. Si se escoge hosting compartido, como su propio nombre indica, se comparte el servidor (el ordenador) con otros portales web, de forma que los recursos que está utilizando un sitio web no están disponible para el resto y viceversa.

Servidor dedicado. Si se escoge un servidor dedicado los recursos no se comparten y, a todos los efectos, es como alquilar un ordenador concreto conectado a internet.

Servidores privados virtuales. Los servidores privados virtuales son un término medio entre hosting compartido y servidor dedicado. Una misma máquina se divide en fracciones con su propia asignación de recursos. Se comparte un mismo equipo, pero cada partición (virtual) es independiente, tiene su propio sistema operativo y su propia configuración.

Hosting en la nube. Y, por último, el hosting en la nube es, como se detalló anteriormente, un conjunto de configuraciones donde se contratan por separado los elementos, de una red de elementos disponibles. Para contratar este tipo de hosting se elige el tamaño del disco duro, la cantidad de CPU disponibles, la RAM, la duplicidad necesaria, la cantidad de espacio para copias de seguridad, etc.

- Se puede definir un **servidor web** como el sistema encargado de devolver contenidos estáticos ante peticiones de HTTP mientras que un **servidor de aplicaciones** es capaz de albergar toda la lógica de negocio de un sistema y, además, gestionar las peticiones HTTP si así se configurase. No obstante, a través de extensiones los servidores web pueden servir contenido dinámico (por ejemplo, PHP) y no dependen para ello de un servidor de aplicaciones como tal (como sí era necesario anteriormente).
- Para comprender lo difuso de los términos hay que remontarse a los años en los que el contenido servido era principalmente HTML plano (no dinámico) e imágenes. Tras la evolución de la web, empresas como Kiva o NetDynamic crearon **JSP** (Java Server Pages) así como Microsoft lanzó **ASP** (Active Server Pages).

Los servidores web cada vez eran capaces de hacer más tareas y los servidores de aplicaciones fueron añadiendo la capacidad de cambiar su lógica en base a cambios en las peticiones HTTP. Básicamente, se equipararon.

- Actualmente, son términos sinónimos o cuasi sinónimos dado que **se usan indistintamente**, y quizá se puede decir que se usa servidor web cuando el sistema está enfocado a interfaz de usuario (un portal web) y servidor de aplicaciones como sistemas más pesados con comunicaciones multicanal, transacciones más pesadas, etc.

- En cualquier caso, lo más importante es conocer los distintos sistemas capaces de servir contenido, con más o menos lógica detrás – o quizá ninguna – hacia internet, y aquí sí que hay consenso sobre las diferentes opciones disponibles:
- Tanto **Nginx** como **Apache** son servidores que corren bajo el **entorno Linux** y que son capaces de integrarse con la lógica de negocio y servir páginas y aplicaciones web complejas.
- Nginx consume menos recursos que Apache, aunque la diferencia no es significativa en la mayoría de los casos.
- Por otro lado, la gestión de los mismos varía siendo el primero un software enfocado a eventos mientras que el segundo es orientado a procesos. Apache lleva más tiempo funcionando por lo que la documentación, localización y resolución de errores suele ser más sencilla, aunque cualquiera de los dos es robusto y se utiliza en entornos profesionales y de alto rendimiento.

- Como se ha comentado anteriormente, las páginas web son un conjunto de ficheros estáticos (HTML, CSS, JavaScript...) que entiende el navegador. Sin embargo, la mayoría de las páginas web modernas (y entiéndase moderna como cualquiera posterior al año 2000) no tiene los ficheros HTML en el servidor de manera única. Lo normal es que esos ficheros se generen de forma dinámica utilizando un lenguaje de programación.
- Cabe destacar que, a pesar de que muchas veces se confunda como tal, el HTML no es un lenguaje de programación, sino de marcado. Con HTML no se puede programar ni establecer una lógica y lo único posible es etiquetar la información.
- Para generar HTML se puede usar cualquier lenguaje de programación existente o cualquier estrategia que se desee, sin embargo, lo más cómodo y habitual es usar un lenguaje preparado para hacerlo.
- A excepción de Java, **todos los lenguajes preparados para generar HTML son interpretados**, esto quiere decir que no se requiere una compilación previa para ejecutarse y que el intérprete es capaz de ejecutar el código escrito directamente. Los principales lenguajes para programación web son PHP, ASP, Java, Ruby, Scala, Python y JavaScript (aunque este último tiene particularidades).

PHP

- PHP (acrónimo recursivo que, en inglés, corresponde a hypertext preprocessor) es sin duda el lenguaje más usado para programación web. La gran mayoría de sitios están generados con PHP o incluye PHP entre sus tecnologías. Tiene licencia GPL (software libre) y fue de los primeros en desarrollarse y ser capaz de embeberse dentro del código HTML.
- Es un lenguaje interpretado, débilmente tipado (es decir, las variables no se tienen por qué declarar de un tipo concreto de datos y además pueden mutar durante la ejecución) y compatible con todos los sistemas operativos.
- Ampliamente usado junto con el servidor web Apache y en entornos Linux, usando MySQL como base de datos. A este conjunto de tecnologías se les llama LAMP (Linux-Apache-MySQL-PHP) y es el stack más usado en desarrollo web.
 - Ejemplos: sitios webs como Wikipedia o Facebook están desarrollados con PHP.
- Por último, hay que destacar un elemento muy importante de cara a la elección de tecnologías para web, y son las herramientas disponibles. En PHP están escritos los principales **CMS** que existen para la web como WordPress, Joomla, Drupal, PrestaShop o Magento y **Framework** como Symfony, Laravel o Phalcon.

ASP.NET

- ASP.NET (antes ASP) es una tecnología creada por Microsoft para desarrollar sitios web dinámicos. Dado que es una tecnología de Microsoft, para poder ejecutarse necesita una máquina con Windows, principalmente Internet Information Server, aunque también puede utilizarse Personal Web Server para desarrollos locales.
- A diferencia de PHP (y de otros lenguajes interpretados), **ASP es un lenguaje compilado**, es decir, para ejecutarse en un equipo ha de compilarse primero para generar un fichero ejecutable compatible con la máquina en la que se está corriendo. En teoría, los lenguajes compilados, al tener ejecutables ya en código entendible por el ordenador, son más rápidos que los interpretados y dan una eficiencia mejor. En la práctica, esta diferencia no es demasiado grande y la mayoría de los lenguajes interpretados ya tienen herramientas para paliar esta pequeña desventaja.

ASP.NET

- Como principales puntos fuertes, se puede destacar:
 - Una **buena velocidad de respuesta**.
 - Una **arquitectura robusta** y que **separa correctamente las capas de presentación y lógica**.
Quizá las ventajas más importantes vienen de la mano de su propio nacimiento, tiene gran parte propietaria (el core es software libre, pero no el resto) y los costes de alojamiento suelen ser superiores.
- En cuanto a **Frameworks** disponibles, ASP.NET se podría definir como un Framework en sí mismo, no obstante, Microsoft tiene dos versiones diferenciadas de su propio Framework:
 - **ASP.NET Core y ASP.NET Full Framework**, siendo el segundo una versión más potente que el primero (también más pesada).
- En cuanto a CMS, lo cierto es que **no existe ningún CMS en ASP.NET** que sea usado de manera extensiva, aunque quizá se pueda citar Umbraco, Kentico o DNN como los más relevantes, están lejos del nivel de uso de los CMS escritos para, por ejemplo, PHP.

Java

- Uno de los lenguajes reyes no solo para desarrollo web, sino para desarrollo en general es, y presumiblemente lo seguirá siendo aún bastantes años, Java.
- Java es un lenguaje de propósito general, orientado a objetos, concurrente y muy extendido; pero muy particular, dado que su forma de ejecutarse así lo es.
- Mientras que los lenguajes interpretados son independientes del entorno (se pueden ejecutar, con el intérprete adecuado en cualquier arquitectura) y los compilados han de compilarse para cada arquitectura particular, Java es un lenguaje compilado, pero compatible y ejecutable en todas las arquitecturas hardware.
- Esto es así porque para ejecutarse siempre se necesita levantar una máquina virtual de Java que es la que realmente ejecuta el programa, es decir, no se está ejecutando realmente sobre el sistema operativo sobre el que está corriendo la máquina, sino sobre esta virtualización.
- Adaptando la virtualización a cada sistema operativo, cualquier ejecutable Java es compatible con todos los sistemas y arquitecturas.

Java

- Esta capa de abstracción, conjuntado con una biblioteca enorme de Plugins y funcionalidades ya desarrolladas y listas para usar, le dio a Java una fama que aún hasta el día de hoy le dura, siendo el lenguaje elegido por multitud de profesionales y, sin duda, una de las principales tecnologías dentro del mundo Enterprise y de los grandes proyectos públicos.
- El éxito de Java como lenguaje de programación de propósito general hizo que fuese utilizado como lenguaje de programación web, en lugar de otros más específicos para esta tarea, sin embargo, con las bibliotecas adecuadas, es un lenguaje más que apto para desarrollar cualquier web o aplicación web sin ningún problema.
- Como dato a tener en cuenta, se puede destacar que es el lenguaje de programación utilizado por los móviles con sistema operativo Android (no así iOS).
- Al igual que con los lenguajes anteriormente estudiados, se pueden citar grandes portales de internet escritos en Java, como Amazon, eBay o LinkedIn.

Ruby

- Muy lejos de los niveles de uso y antigüedad de Java o PHP, Ruby es un lenguaje interpretado, reflexivo y orientado a objetos, cuya principal característica es la simplicidad. Ruby está escrito para que se pueda leer y aprender con facilidad, acercándose lo máximo al lenguaje natural (en inglés).
- Una característica particular de Ruby es que en este lenguaje todo es un objeto, por lo que todo admite propiedades y tiene (o puede tener) métodos definidos. Tanto es así, que hasta los tipos simples como los números enteros (o simples en otros lenguajes) pueden comportarse como objetos.
- En general, se le suele definir como un lenguaje moderno y estructurado y fue adoptado rápidamente por multitud de startups y empresas de nueva creación, así como consultoras para gran cantidad de proyectos de todos los tamaños. Tras el bum sufrido en 2015, su uso se estabilizó convirtiéndose en una herramienta más de la caja de herramientas de los desarrolladores.

Ruby

- Una de las cosas que hizo que Ruby resultase muy utilizado fue la potencia y simplicidad de su Framework por antonomasia, Rails (denominándose al conjunto Ruby on Rails) que proporcionaba una buena base para desarrollo web.
- Algunos ejemplos de grandes proyectos escritos con estas tecnologías pueden ser GitHub, la herramienta para soporte a usuarios Zendesk o el portal para subir diapositivas SlideShare (que, posteriormente, se fusionó con LinkedIn).

Scala

- Scala es quizá el lenguaje más particular de los analizados en esta unidad, dado que inicialmente no fue concebido para desarrollo web y tampoco es usado con ese propósito con demasiada frecuencia.
- Es un lenguaje compilado que corre sobre una máquina virtual de Java, está orientado a objetos, aunque incorpora características de lenguajes funcionales como técnicas de emparejamiento de patrones (Matching Patterns) para modelar tipos algebraicos. Es quizá esta característica particular la que lo hace ser muy usado en entornos relacionados con Big Data y tratamiento de datos, además de su interoperabilidad con los programas escritos en Java (Hay que recordar que Java tiene multitud de complementos disponibles y código escrito listo para ser utilizado, además de un conjunto enorme de desarrolladores que lo conocen).
- No hay grandes proyectos web como tal escritos utilizando esta tecnología, pero sí herramientas ampliamente usadas como Apache Spark (para procesar gran cantidad de datos), Salding (de Twitter) o Apache Kafka.

Python

- Python es uno de los lenguajes más demandados en la actualidad y es debido a su versatilidad. Es un lenguaje interpretado muy similar a PHP en concepto y funcionalidad, pero al contrario que PHP (el cual ha evolucionado, prácticamente, en exclusiva para creación web) ha evolucionado con ramas muy diversas usándose para scripting en servidores, tratamiento de datos a gran escala (Big Data) y, por supuesto, creación de sitios web.
- Al igual que Ruby, Python fue concebido para tener una gran legibilidad y transparencia (de hecho, Ruby se inspiró en Python, como confirmó su creador, Yukihiro Matsumoto).
- De cara a la creación web, Python cuenta con un Framework (Django) y un CMS (Django CMS) muy versátiles y potentes, además de bien estructurados.

Python

- Quizá, la única gran desventaja de Python con respecto a PHP en cuanto a desarrollo web se refiere es la gran masa de gente, código y herramientas, que existen para PHP, siendo mucho menor que las de Python. Este hecho, que además es complicado de paliar, hace que aún a día de hoy siga siendo un lenguaje utilizado mucho menos para desarrollo web que para otros menesteres.
- A pesar de no usarse con tanta frecuencia, gran cantidad de sitios web enormes y muy populares han sido escritos en esta tecnología, como YouTube, Google, Instagram, Spotify o Dropbox.

JavaScript

- JavaScript ha sido tradicionalmente un lenguaje ejecutado en el navegador, es decir, en la parte del cliente y no del servidor. No obstante, tras la aparición de Node.js (un intérprete de JavaScript para servidores) se comenzó a utilizar también para algunas funcionalidades o aplicaciones en servidores.
- Node.js es un entorno de ejecución de JavaScript orientado a eventos asíncronos. Este punto fue su principal novedad y es su punto fuerte. Normalmente, cuando se realiza un desarrollo web, es el cliente el que inicia la interacción (con una petición HTTP) y el cliente responde. Con Node.js, sin embargo, este patrón se cambia y el servidor, de forma sencilla, puede mandar eventos al navegador de manera reactiva. Este tipo de funcionalidad resulta tremendamente útil para algunas aplicaciones como chats, aplicaciones en tiempo real, gestión de tráfico y estadísticas, etc. Además, es un entorno pensado desde el principio para el alto rendimiento y para ser escalable, por lo que tiene herramientas para que se pueda usar en proyectos exigentes desde el punto de vista del rendimiento.
- Debido a sus características, Node.js no es un lenguaje que se use con frecuencia como único lenguaje de desarrollo web (excepto para algunas aplicaciones concretas), pero es muy utilizado como complemento para desarrollar características en las que puede aportar mucho valor. De esta forma, grandes sitios como Netflix, LinkedIn, PayPal o Booking utilizan Node.js en alguna parte de sus portales de servicios.



IMR