1.1.- INTRODUCCIÓN

Luis Enrique González de la Fuente

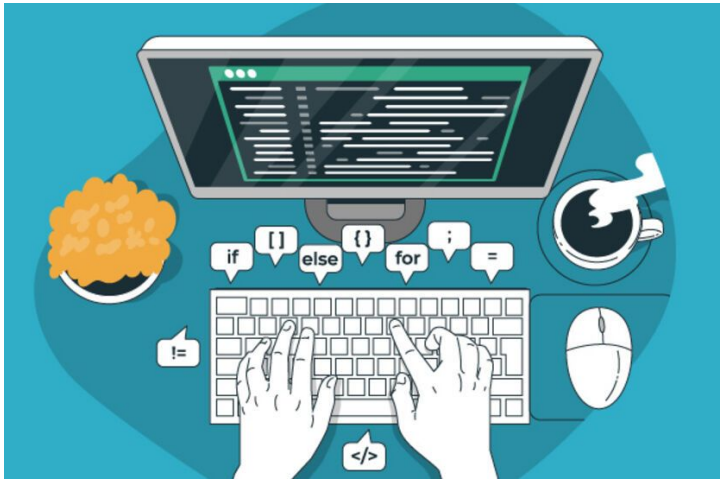
CONTENIDOS

1. Modelos de programación: entornos cliente/servidor.
2. Tecnologías implicadas
3. Tipos de páginas Web.
4. Servidores web.
5. Integración con los lenguajes de marcas.
6. IDEs y herramientas de programación.
7. Código en las páginas Web.

OBJETIVO DEL PROGRAMADOR

Un programador o desarrollador web genera una solución que procesa las peticiones de un cliente y devuelve una respuesta adecuada a las necesidades y requerimientos.

Pero también debe preocuparse por crear aplicaciones dinámicas, conectar con los datos oportunos, utilizar elementos multimedia y seguir siempre los estándares marcados.

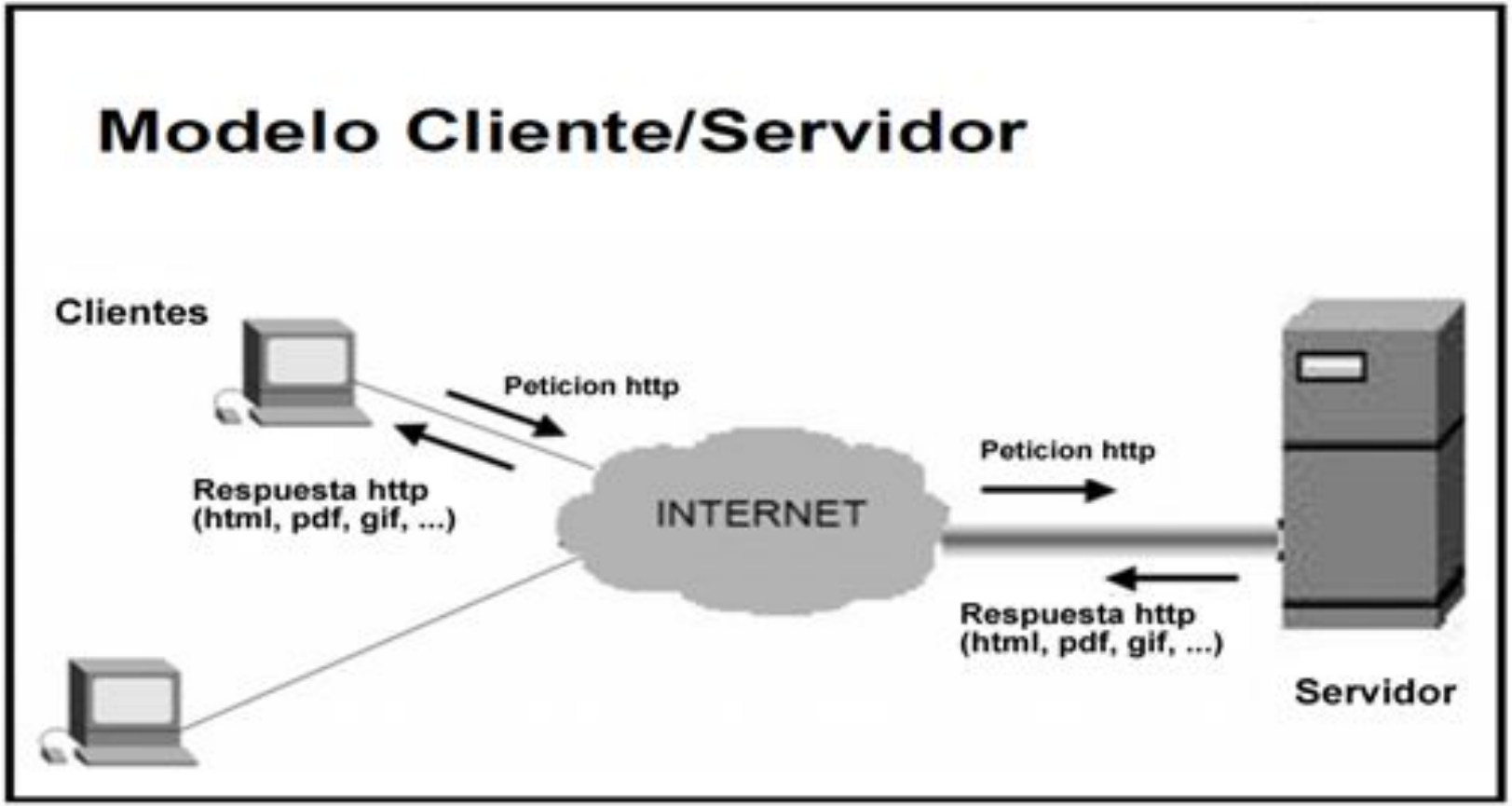


1.- MODELOS DE PROGRAMACIÓN

Elementos implicados en la comunicación web:

- World Wide Web
- Protocolos: (como por ejemplo TCP, IP, HTTP, FTP, SMTP…)
- Resolución de nombres: DNS

¿Qué más?

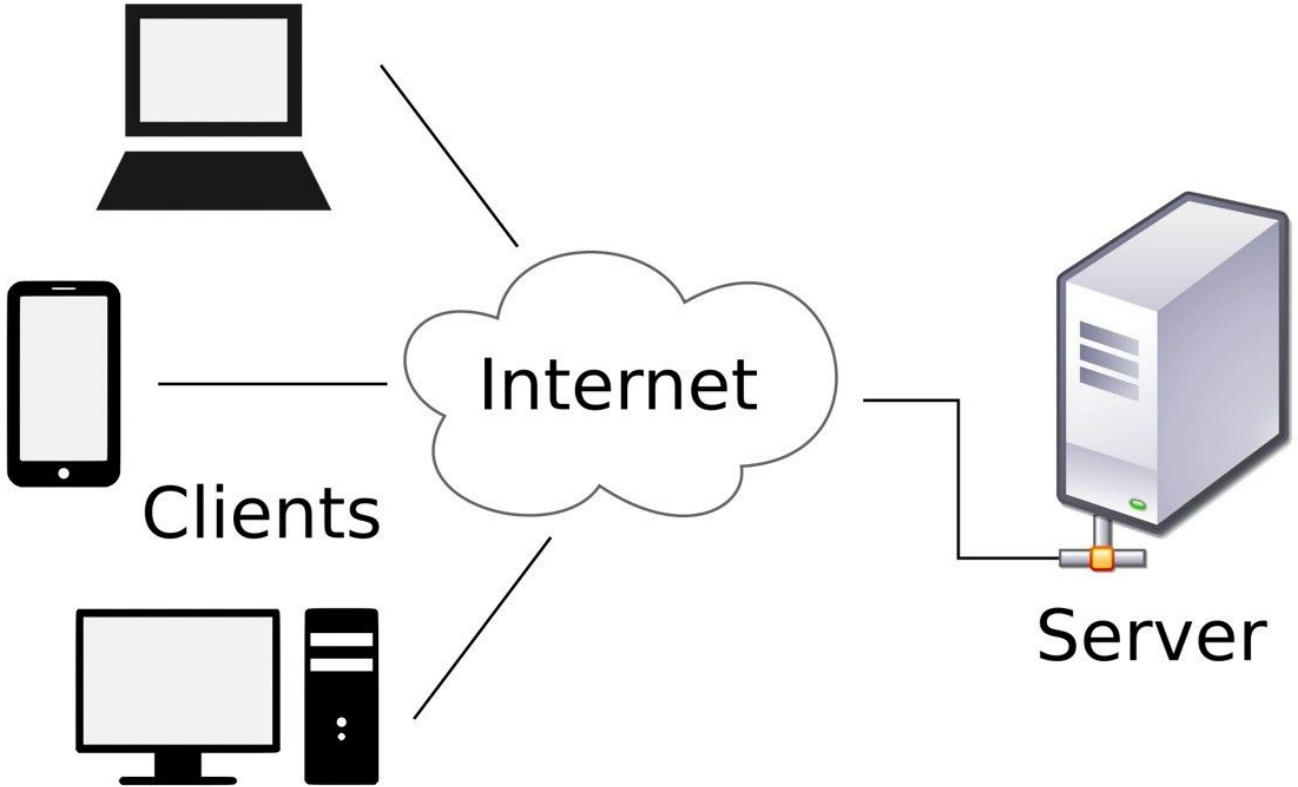

Modelo Cliente/Servidor
Clientes
Peticion http
Respuesta http
(html, pdf, gif, ...)
INTERNET
Peticion http
Respuesta http
(html, pdf, gif, ...)
Servidor

MODELOS DE PROGRAMACIÓN

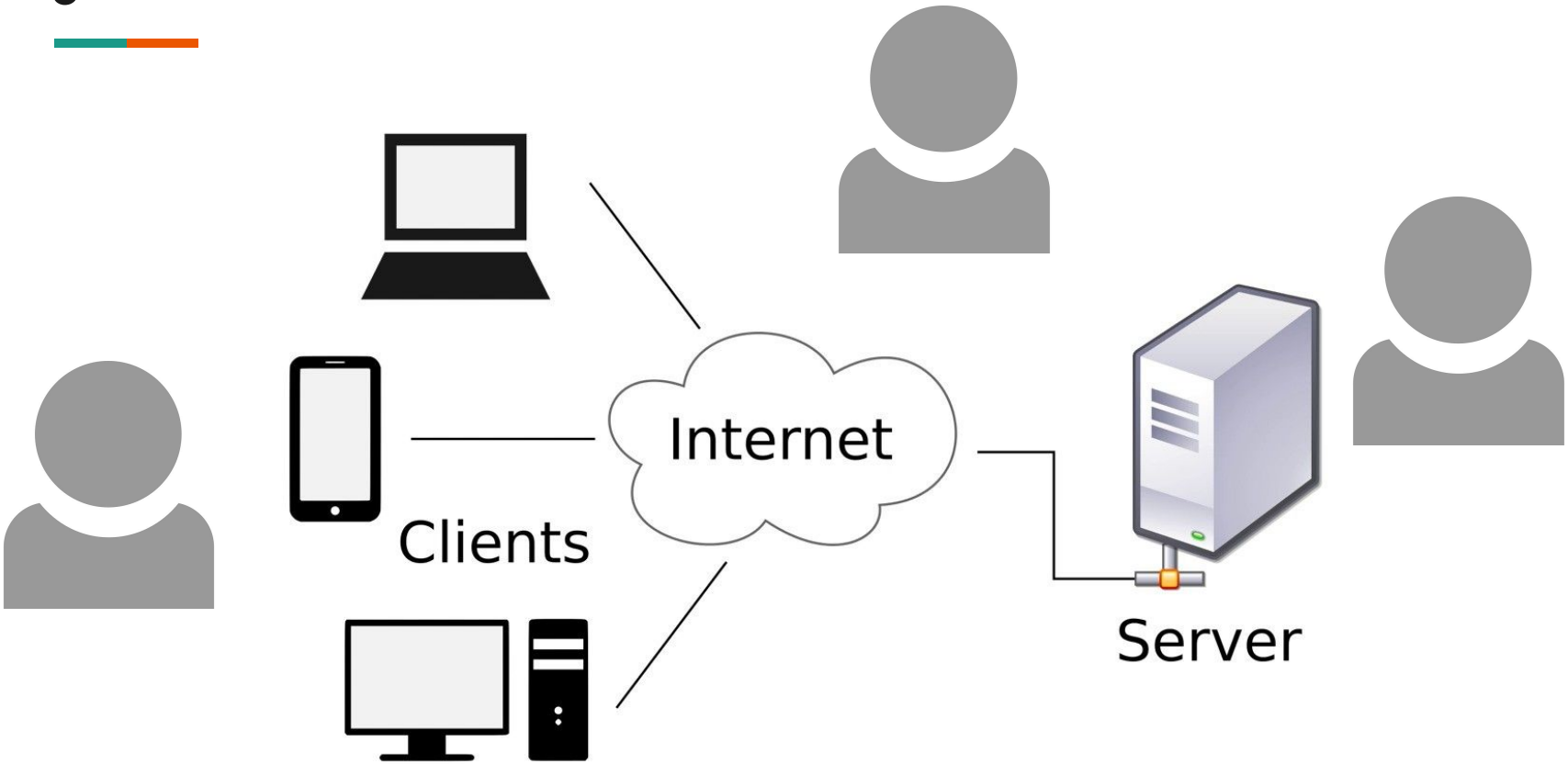
CONFIGURACIÓN ARQUITECTÓNICA DE LOS ELEMENTOS:

- Distribución de los elementos y la función que tiene cada uno de ellos.
- La configuración más habitual se basa en el modelo denominado Cliente/Servidor.
- Se basa en la idea de servicio: un cliente es un consumidor de servicios y el servidor es el proveedor de dichos servicios.

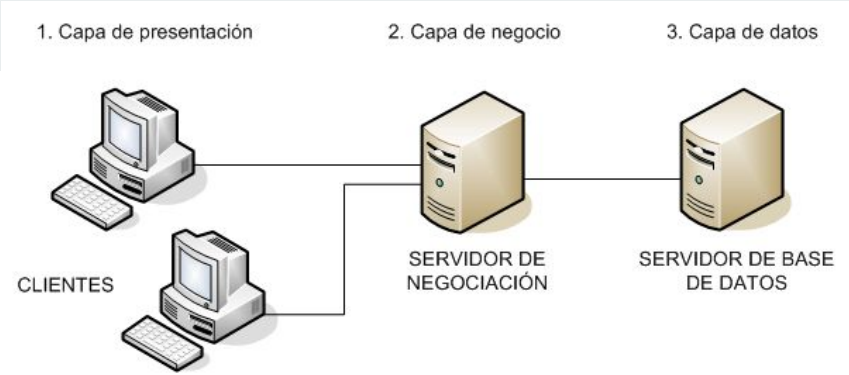
ESQUEMA BÁSICO



¿USUARIOS?



CAPAS DE ESPECIALIZACIÓN



Las funcionalidades que deben existir en la relación cliente/servidor se pueden separar en tres capas:

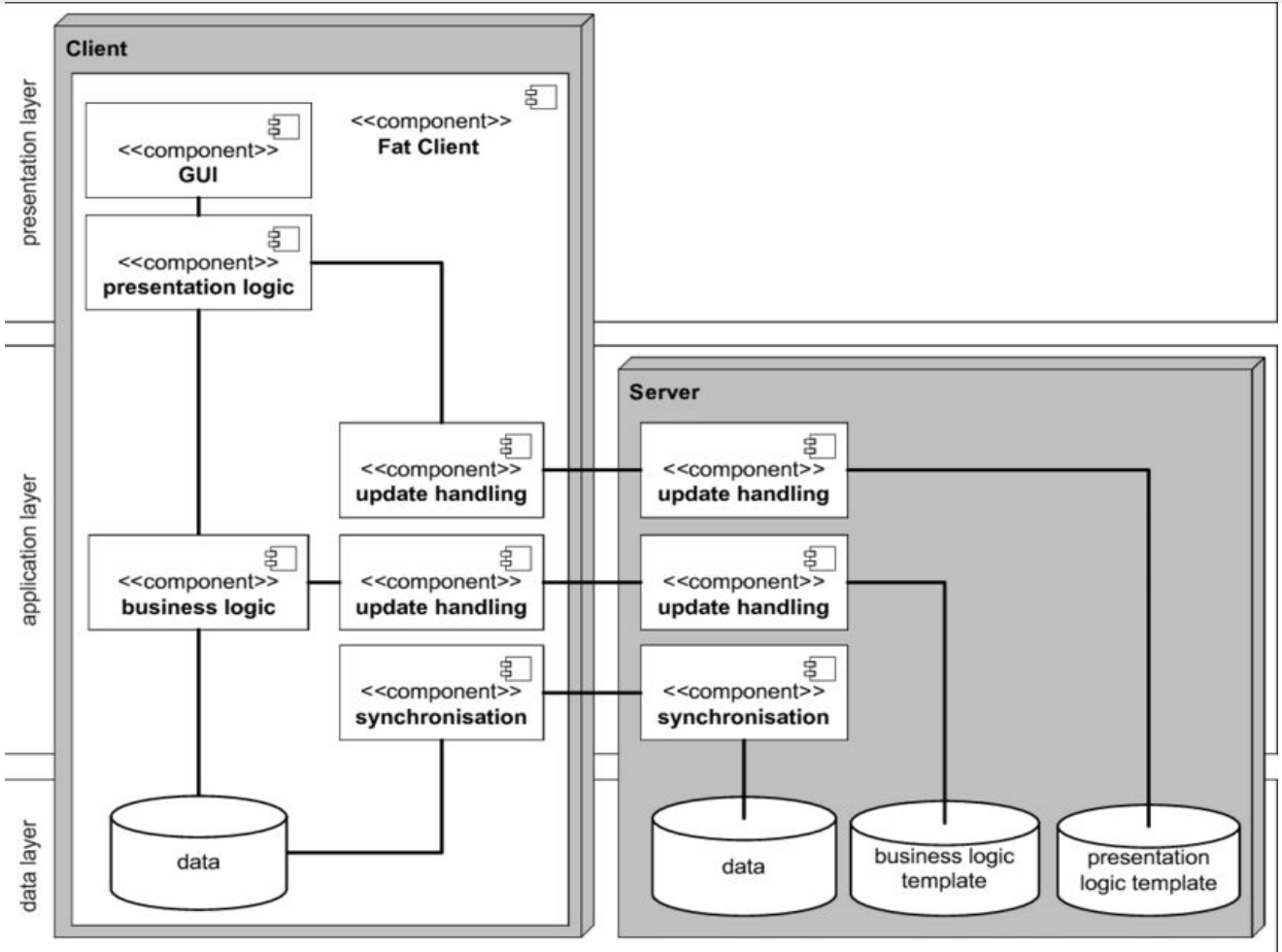
- Capa de presentación: la que ve el usuario. Interfaz gráfica que interactúa con el usuario. Normalmente está en el cliente
- Capa de la lógica de negocio: conoce y gestiona la funcionalidad de nuestro sistema (reglas de negocio). Recibe peticiones y envía las respuestas. Puede estar en el cliente o en el servidor.
- Capa de persistencia o de almacenamiento de datos: para acceder y guardar los datos.

Los modelos arquitectónicos de programación se clasifican según el lugar donde encontremos cada capa y según ello se elegirán unas u otras tecnologías.

CONFIGURACIONES ARQUITECTÓNICAS

- Según reparto de **funciones**: Modelo de 2 capas (se juntan lógica y datos) o de 3 capas.
- Según el **tamaño** de los componentes: Fat client (thin server) vs Fat server (thin client): si la capa lógica se encuentra en el cliente o en el servidor.
- Según el **servicio** ofrecido: Puede haber tipos de servidores, como de ficheros, de bases de datos, de transacciones, de objetos o web.

FAT CLIENT


Client
presentation layer
application layer
data layer
<<component>>
GUI
<<component>>
Fat Client
<<component>>
presentation logic
<<component>>
update handling
<<component>>
business logic
<<component>>
update handling
<<component>>
synchronisation
data
Server
<<component>>
update handling
<<component>>
update handling
<<component>>
synchronisation
data
business logic
template
presentation
logic template

2.- TECNOLOGÍAS IMPLICADAS

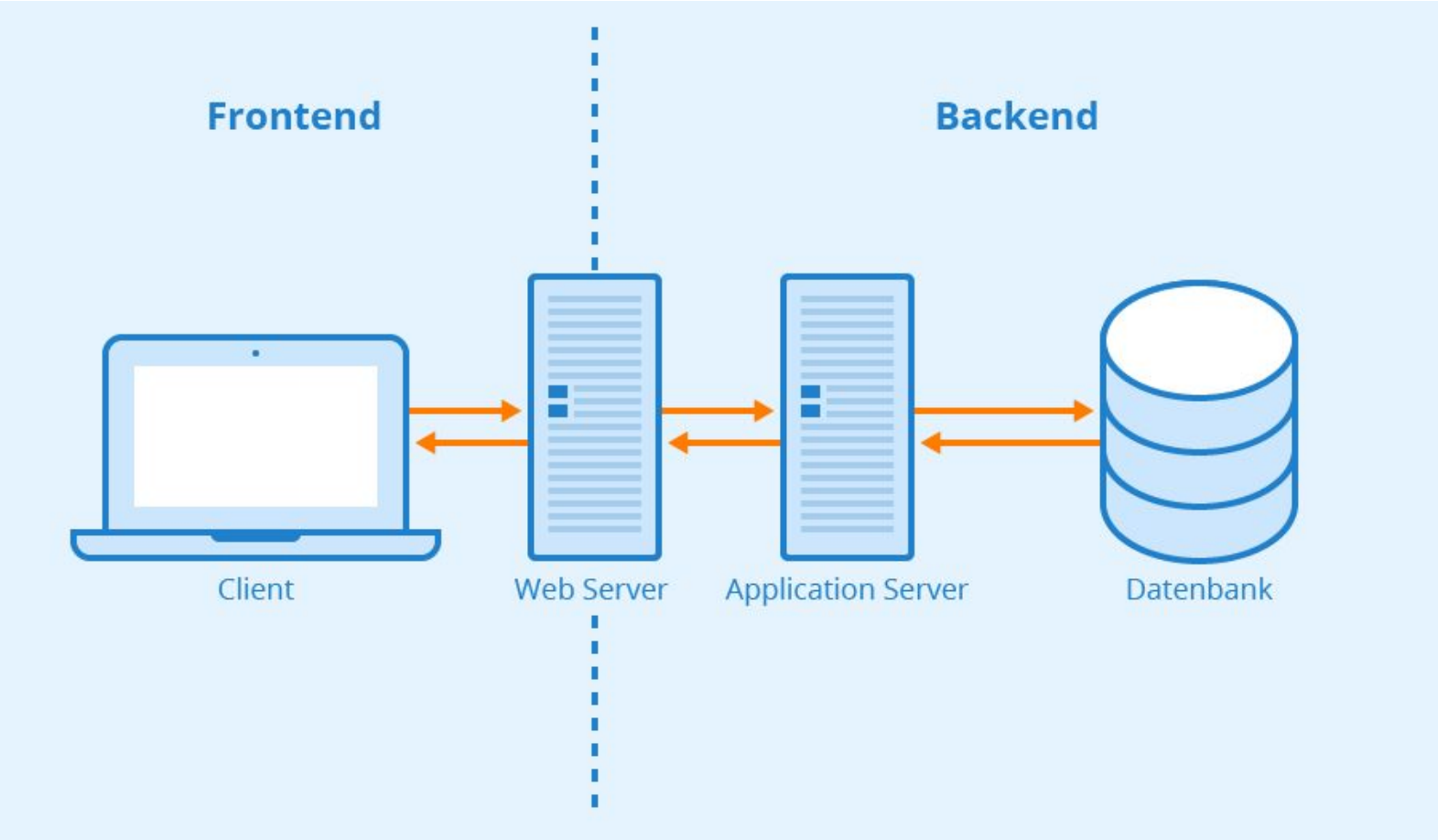
¿ES LO MISMO?

CLIENTE y SERVIDOR

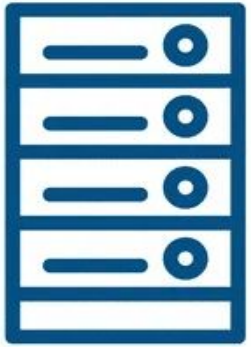
vs

FRONT-END y BACK-END

Luis Enrique González de la Fuente


Frontend
Backend
Client
Web Server
Application Server
Datenbank

FUNCIONES

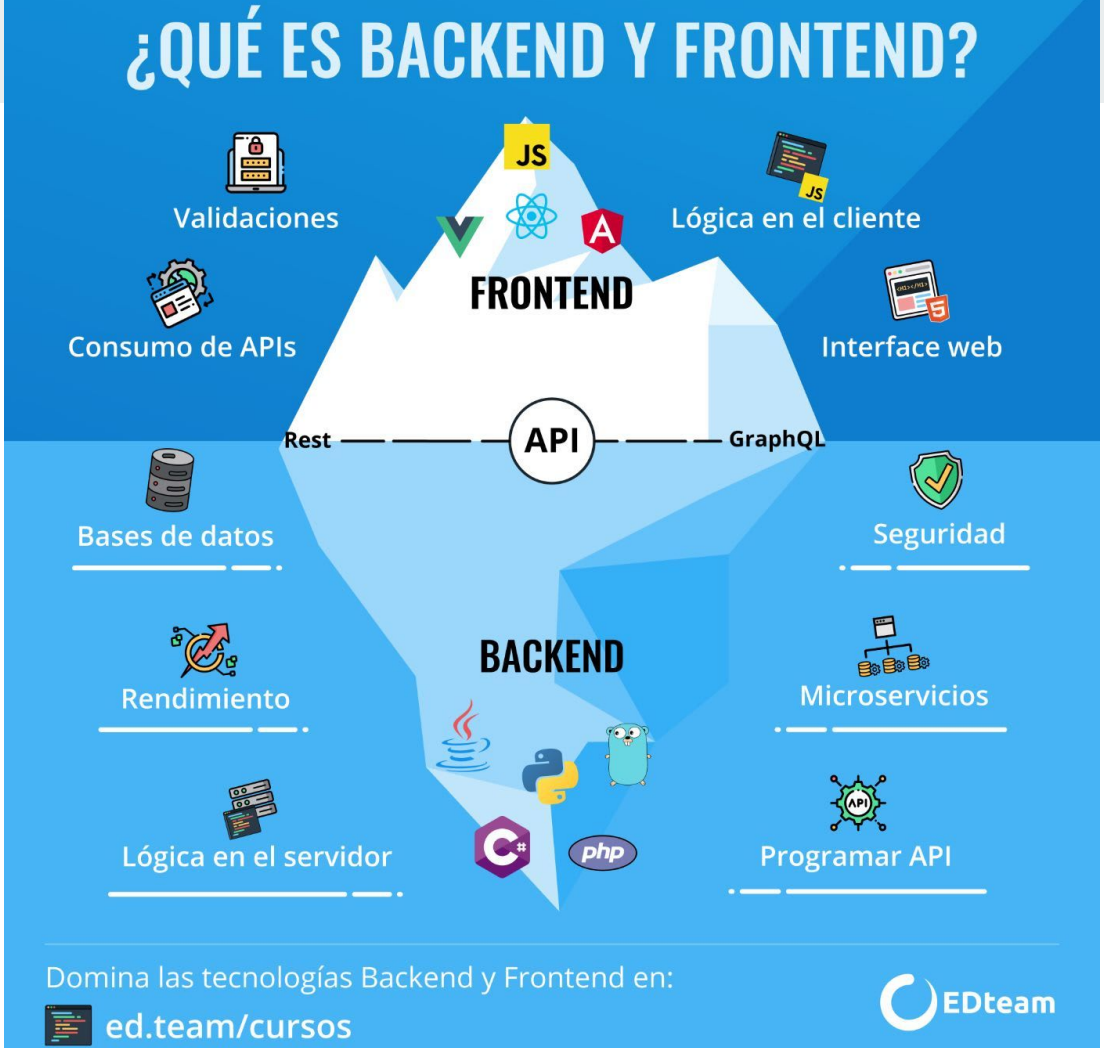


Front End

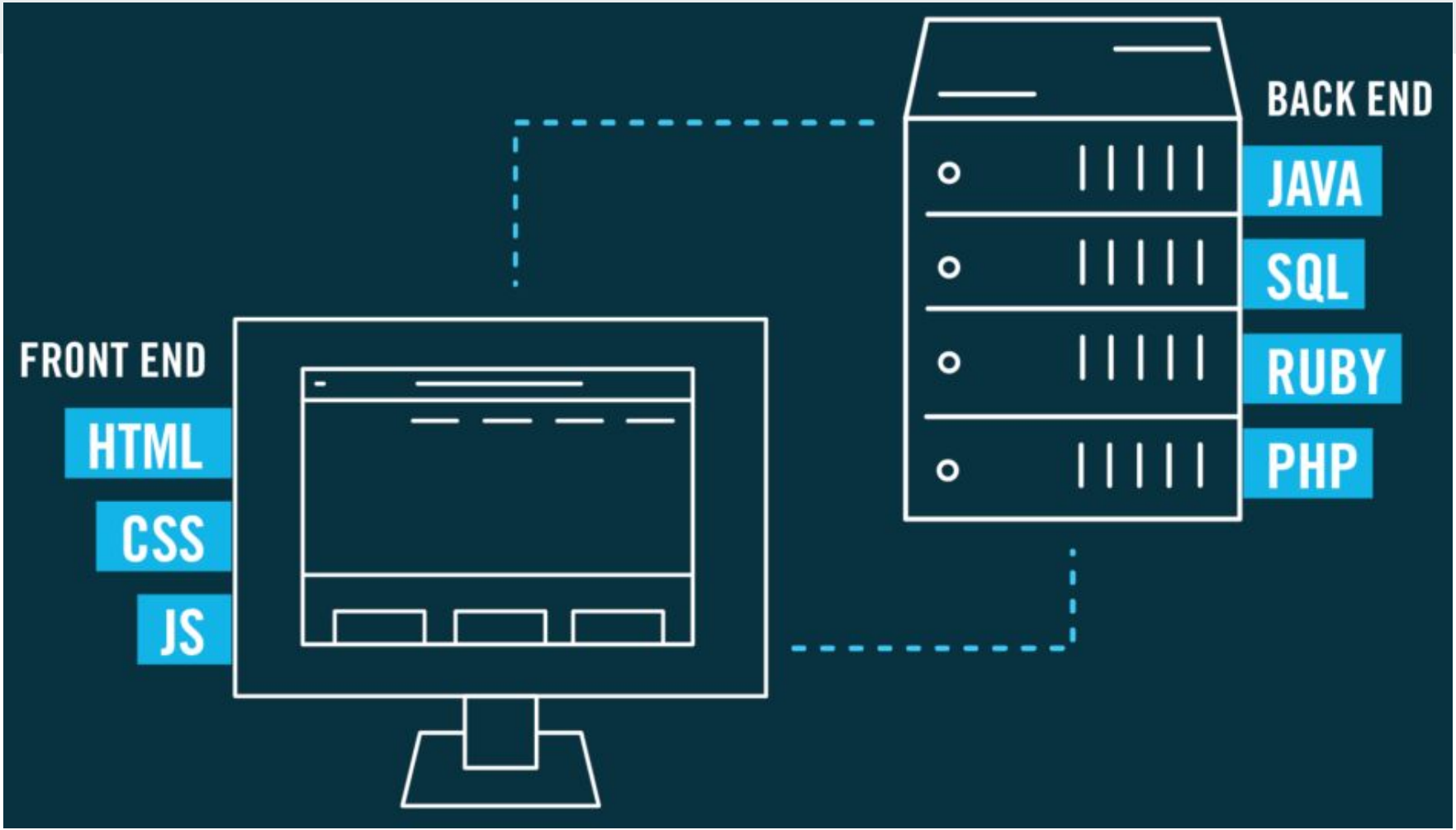
- Markup and web languages such as HTML, CSS and Javascript
- Asynchronous requests and Ajax
- Specialized web editing software
- Image editing
- Accessibility
- Cross-browser issues
- Search engine optimisation

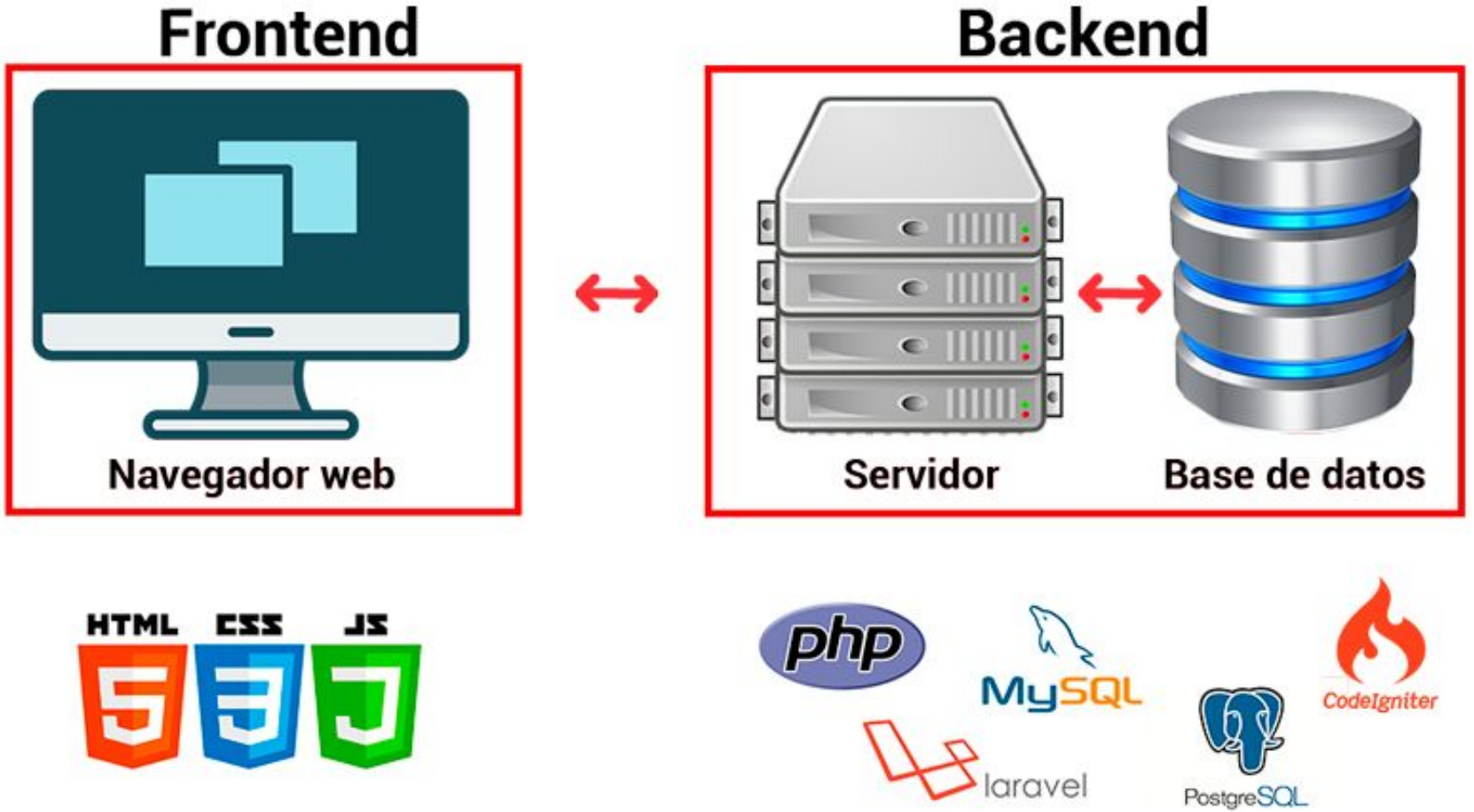
Back End

- Programming and scripting such as Python, Ruby and/or Perl
- Server architecture
- Database administration
- Scalability
- Security
- Data transformation
- Backup

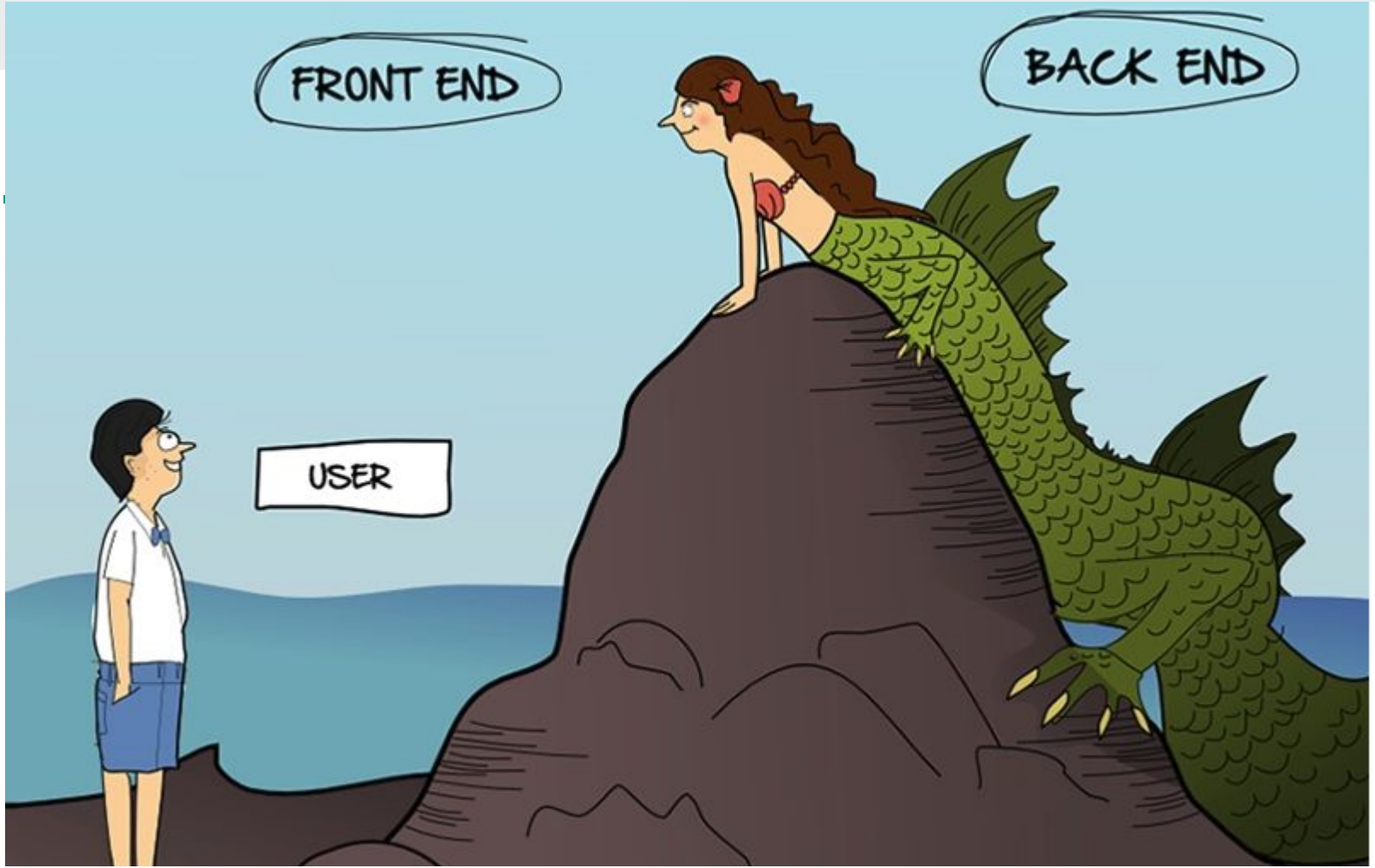

¿QUÉ ES BACKEND Y FRONTEND?
JS
Validaciones
Lógica en el cliente
FRONTEND
Consumo de APIs
Interface web
Rest
API
GraphQL
Bases de datos
Seguridad
BACKEND
Rendimiento
Microservicios
Lógica en el servidor
Programar API
Domina las tecnologías Backend y Frontend en:
ed.team/cursos
EDteam

TECNOLOGÍAS




Frontend
Backend
Navegador web
Servidor
Base de datos
HTML
CSS
JS
php
MySQL
CodeIgniter
laravel
PostgreSQL

¿DÓNDE NOS DEJAMOS AL USUARIO?



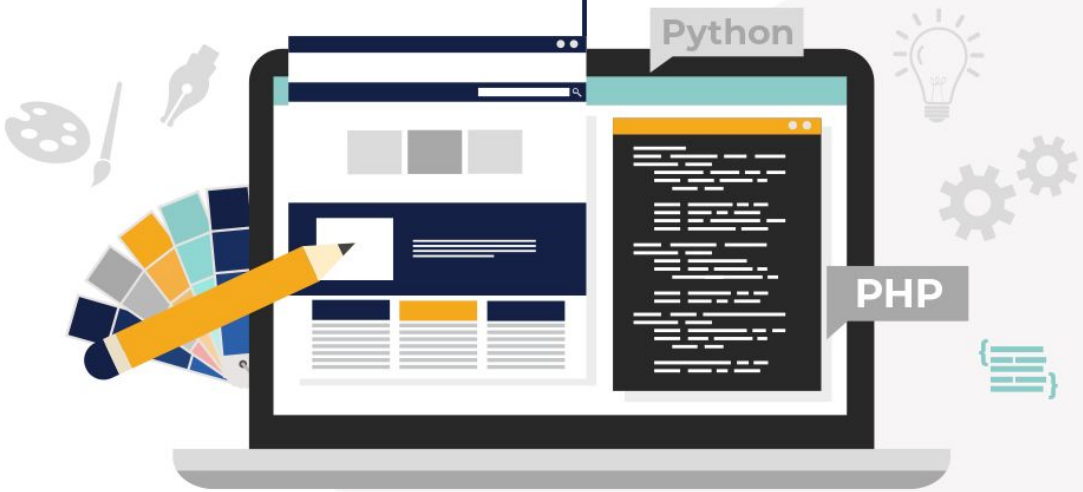
¿Y EL PROGRAMADOR?

¿EN QUÉ DEBE PENSAR CADA UNO?

Frontend vs. Backend

- Más relacionado con el diseño, lo que un usuario ve.
- La navegación es un aspecto fundamental; y esto es, que la página sea intuitiva.
- Tiempo de carga es el adecuado.
- Adaptabilidad a los distintos soportes.

- Aquella que los usuarios no ven, más interno.
- Engloba el servidor, las bases de datos y las aplicaciones.
- La seguridad supone un elemento central.
- Sumamente importante que la información sea funcional.



¿QUÉ DIFERENCIA HAY ENTRE CREAR UNA PÁGINA WEB Y PROGRAMAR UNA APLICACIÓN WEB?

Luis Enrique González de la Fuente

3.- TIPOS DE PÁGINAS WEB

Según la forma en la que el servidor envía la página Web podemos tener:

- Aplicaciones Web **estáticas**: el servidor envía una página en la que no hay ningún tipo de interacción ni en la página ni como respuesta del servidor. Normalmente son HTML básicos.
- Aplicaciones Web **dinámicas**: codificadas mediante HTML dinámico o DHTML (HTML, CSS y Javascript) son aquellas en las que la página cambia su visualización según alguna interacción con el usuario.
- Aplicaciones Web **interactivas**: en este caso existe una comunicación continua entre cliente y servidor, lo que hace que haya procesamiento en ambas partes.

TECNOLOGÍAS ASOCIADAS

A lo largo de la historia se han utilizado diversas tecnologías diferentes en ambos extremos:

- En el **cliente** encontramos tecnologías para la creación de aplicaciones interactivas como HTML (como un formulario), controles ActiveX (certificados), los ya extintos objetos Flash (animaciones interactivas) o los applets de Java y otras tecnologías como AJAX (carga dinámica de contenidos).
- En la parte del **servidor** ha sido más habitual utilizar lenguajes embebidos en código HTML (como PHP, ASP, ASP.Net de Microsoft o JSP), o realizar ejecutables que se cargaban en la página de tipo CGI o servlets.

APLICACIONES WEB DINÁMICAS

Las aplicaciones web son herramientas digitales pensadas para que sean accesibles desde un navegador. En este contexto, una página web **dinámica** es una aplicación informática que utiliza bases de datos para cargar su información. El contenido de estas páginas cambia siempre que el usuario entra, pudiendo variar la información **según la interacción** del visitante de la web.

GENERACIÓN DINÁMICA DE PÁGINAS WEB. VENTAJAS

Como definición simple podríamos decir que una página web **dinámica** permite actualizar el contenido (personalizar) cuando se desee sin la necesidad de saber lenguaje HTML

Las principales **ventajas** son que permiten una mayor interacción y mejorar en gran medida la experiencia del usuario, su diseño tiene múltiples posibilidades.

- Se amolda a cualquier objetivo de negocio ya que las posibilidades que te ofrece una página web dinámica son prácticamente infinitas.
- Facilidad para añadir contenidos y modificarlos incluso mediante la interacción con el usuario.
- Mejora el posicionamiento en los buscadores, ya que éstas obtienen mejor posicionamiento si existe una actualización constante de los contenidos.
- Se administran desde un CMS (content management system).
- Gracias al panel de control no es necesario ir al servidor.
- Diseño web responsive.

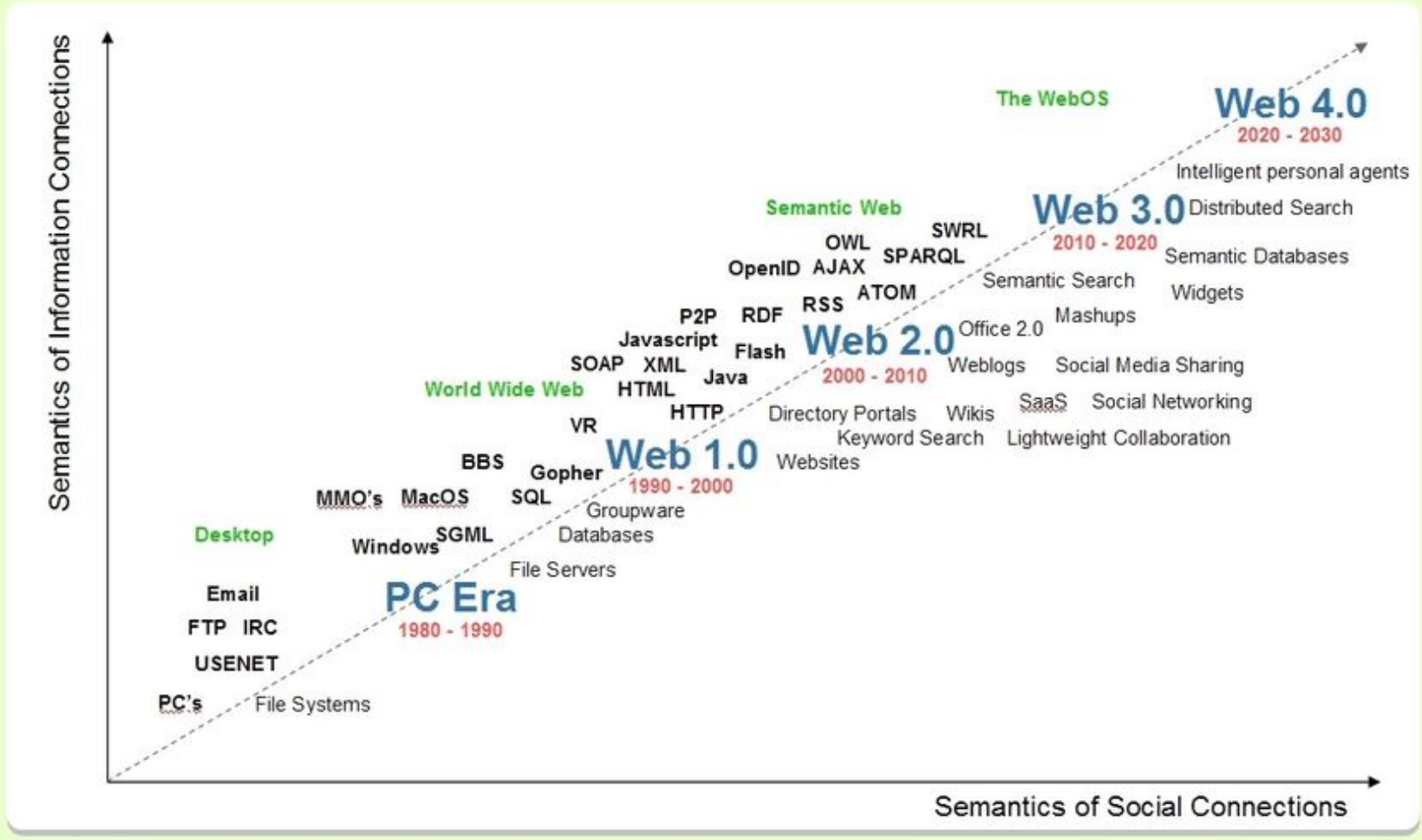
APLICACIONES WEB ENRIQUECIDAS

- Una rich Internet application (**RIA**), "aplicación de Internet enriquecida" o "aplicación rica de internet" (ARI), es una aplicación web con características similares a las aplicaciones de escritorio tradicionales aunque usando un navegador para ejecutarse.
- Las RIA surgen como una combinación de las ventajas que ofrecen las aplicaciones web y las aplicaciones tradicionales. Buscan mejorar la **experiencia** y **productividad** del usuario.
- Evitan el problema de que en las aplicaciones web hay una **recarga** continua de páginas web cada vez que el usuario pulsa sobre un enlace. De esta forma se produce un tráfico muy alto entre el cliente y el servidor, llegando muchas veces a recargar la misma página con un cambio mínimo.
- En los entornos RIA, en cambio, no se producen recargas de página, ya que desde el principio se carga toda la aplicación, y solo se produce comunicación con el servidor cuando se necesitan datos externos como datos de una base de datos o de otros ficheros externos.

APLICACIONES CGI Y DERIVADOS

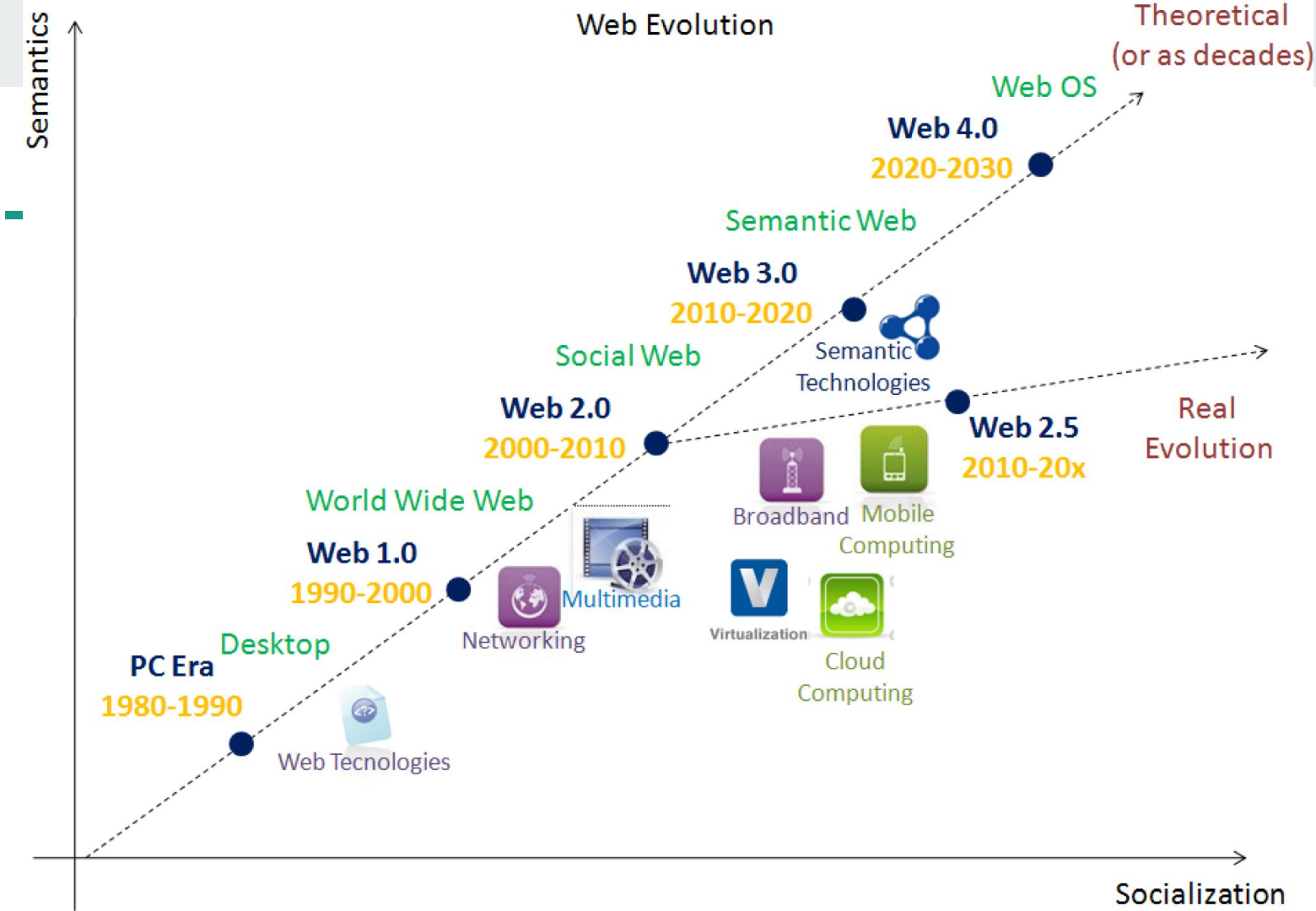
- Los **CGI** (Common Gateway Interface) es un estándar que define la forma en la que un programa externo a la aplicación Web (programado en cualquier lenguaje de programación) se encarga de, mediante unos parámetros de entrada, devolver el contenido que se va a visualizar.
- Al ser un programa externo, por cada petición se lanza una nueva instancia del programa, lo que satura el servidor.
- Evolucionó a **FastCGI** y también hay alternativas similares como los servlets de Java o los JavaBeans

PREVISIÓN DE EVOLUCIÓN WEB EN 2007



Source: Radar Networks & Nova Spivack, 2007 – www.radarnetworks.com

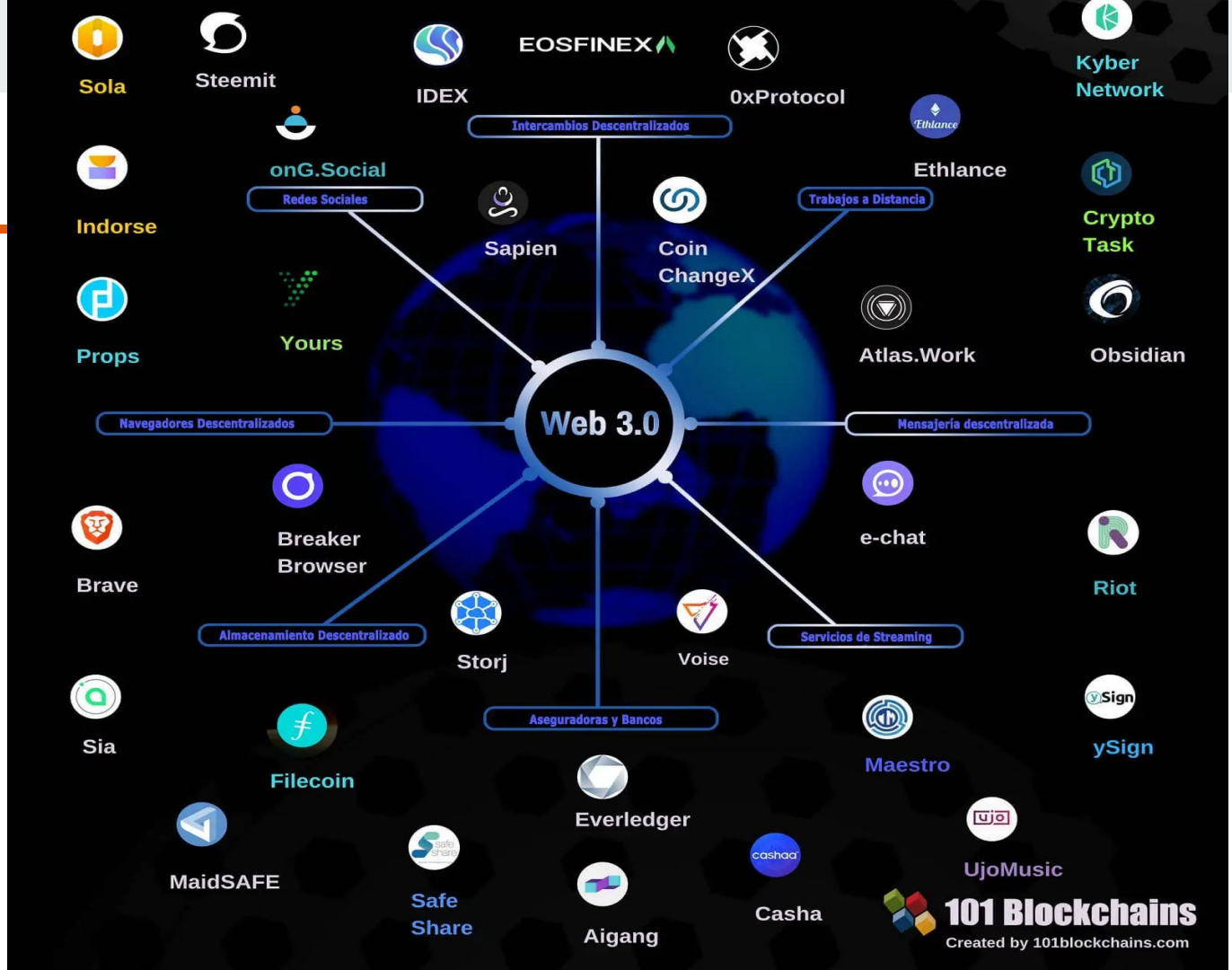

The WebOS
Web 4.0
2020 - 2030
Intelligent personal agents
nantic Web
Web 3.0
Distributed Search
2010 - 2020
SWRL
OWL
SPARQL
AJAX
Semantic Databases
Semantic Search
ATOM
Widgets
RSS
Mashups
Web 2.0
Office 2.0
Weblogs
Social Media Sharing
2000 - 2010

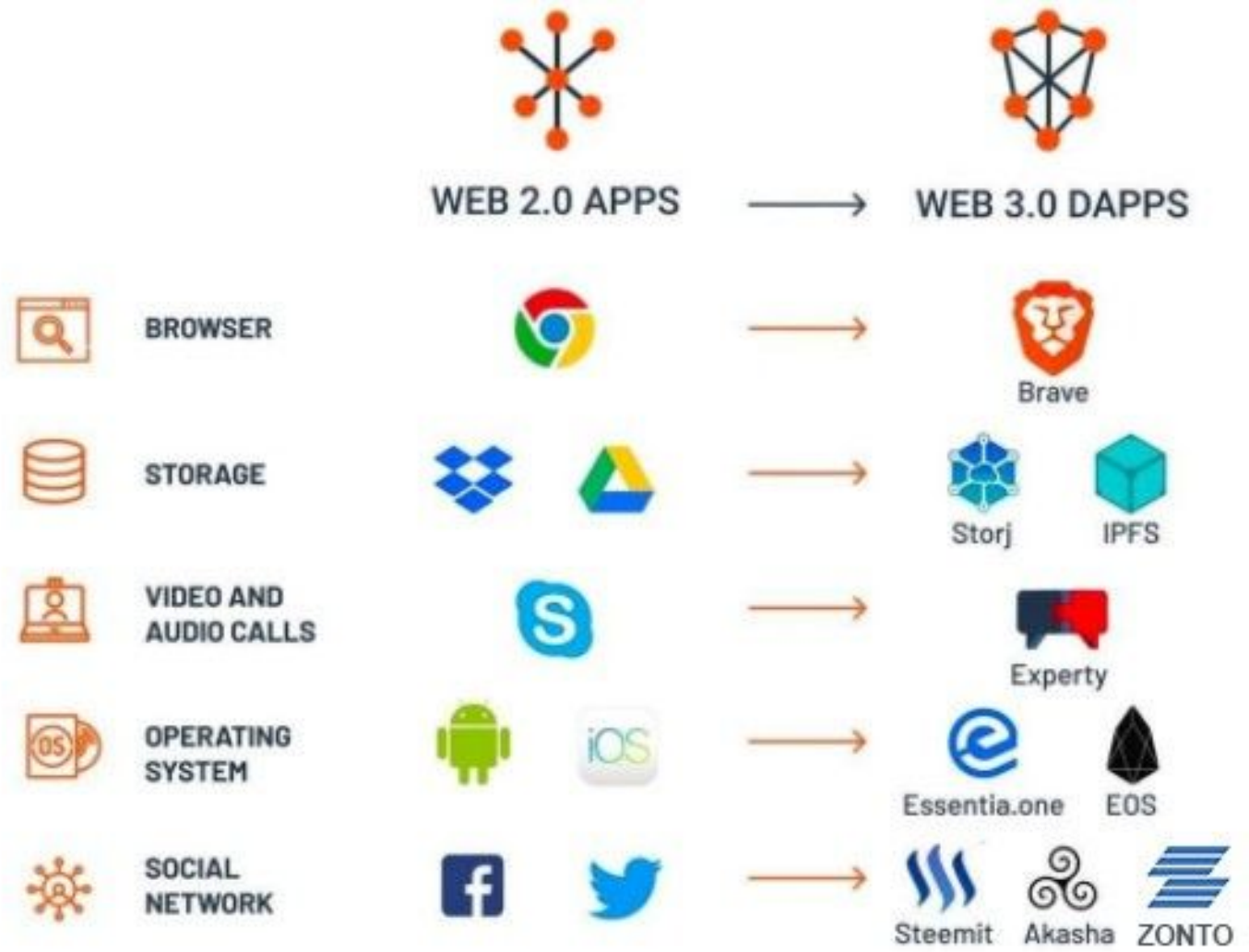

Web Evolution
Semantics
Socialization
Theoretical
(or as decades)
Web OS
Web 4.0
2020-2030
Semantic Web
Web 3.0
2010-2020
Semantic
Technologies
Social Web
Web 2.0
2000-2010
Web 2.5
2010-20x
Real
Evolution
Broadband
Mobile
Computing
World Wide Web
Web 1.0
1990-2000
Multimedia
Networking
Virtualization
Cloud
Computing
Desktop
PC Era
1980-1990
Web Tecnologies

EVOLUCIÓN REAL

- Descentralización.
- Blockchain.
- Criptomonedas.
- NFT
- …


Yahoo!
Google
msn
Web 1.0
"Read Only",
Decentralized
aws
Uber
Web 2.0
Participatory,
Centralized
IPFS
sia
Web 3
No Intermediaries,
Decentralized


Web 3.0
Intercambios Descentralizados
Redes Sociales
Trabajos a Distancia
Navegadores Descentralizados
Mensajería descentralizada
Almacenamiento Descentralizado
Servicios de Streaming
Aseguradoras y Bancos
Sola
Steemit
IDEX
EOSFINEX
0xProtocol
Kyber Network
onG.Social
Ethlance
Indorse
Sapien
Coin ChangeX
Crypto Task
Props
Yours
Atlas.Work
Obsidian
Brave
Breaker Browser
e-chat
Riot
Storj
Voise
Sia
Filecoin
Maestro
ySign
Everledger
MaidSAFE
Safe Share
Aigang
Casha
UjoMusic
101 Blockchains
Created by 101blockchains.com


WEB 2.0 APPS
WEB 3.0 DAPPS
BROWSER
Brave
STORAGE
Storj
IPFS
VIDEO AND AUDIO CALLS
Experty
OPERATING SYSTEM
iOS
Essentia.one
EOS
SOCIAL NETWORK
Steemit
Akasha
ZONTO

LENGUAJES EN EL SERVIDOR EN LA ACTUALIDAD

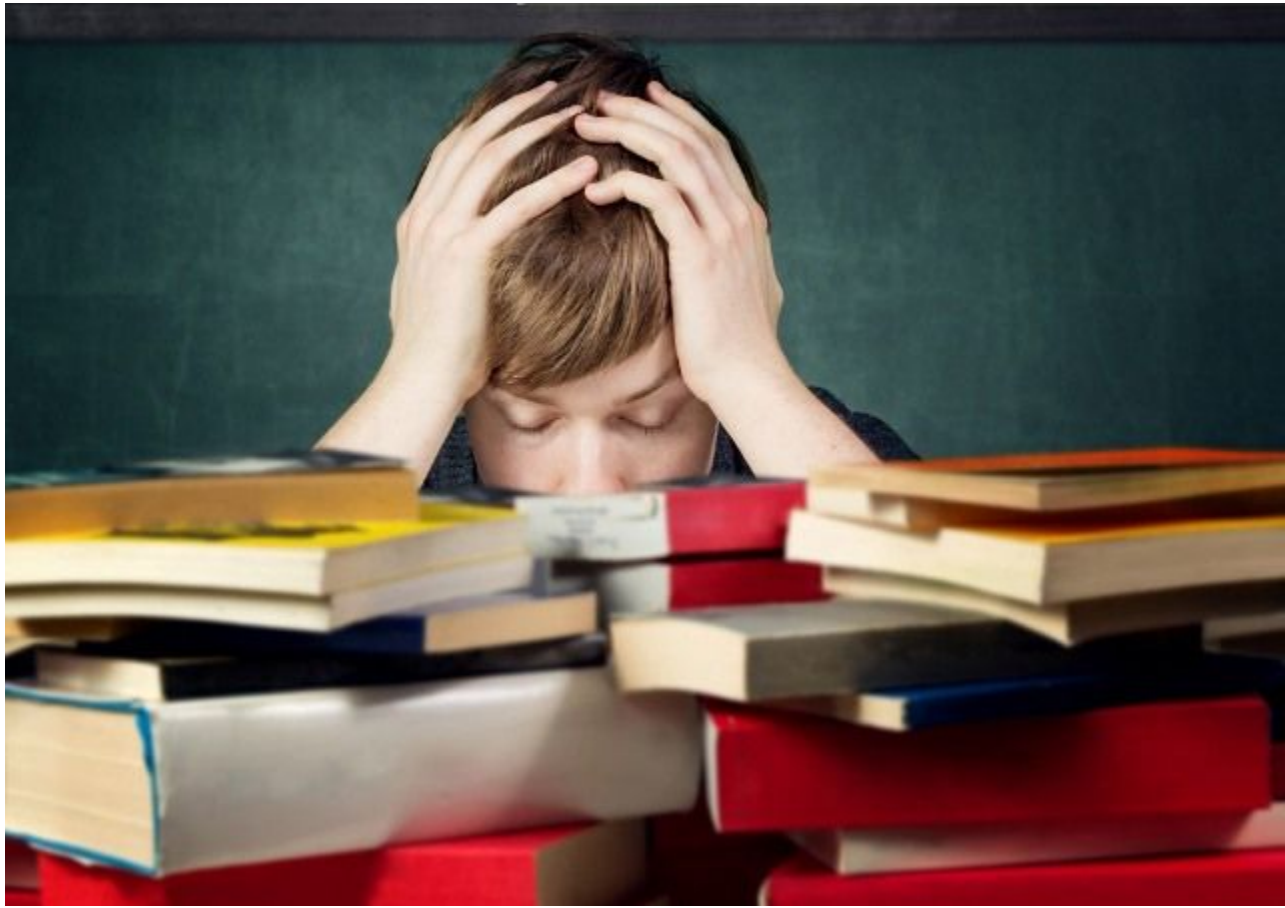
LENGUAJES DE SCRIPTING (intercalado en el HTML):

- **PHP**
- ASP y ASP.Net
- JSP
- Perl
- Python
- ColdFusion
- Ruby
- ...

TECNOLOGÍAS UTILIZADAS POR LAS “GRANDES” PÁGINAS (9-2022)

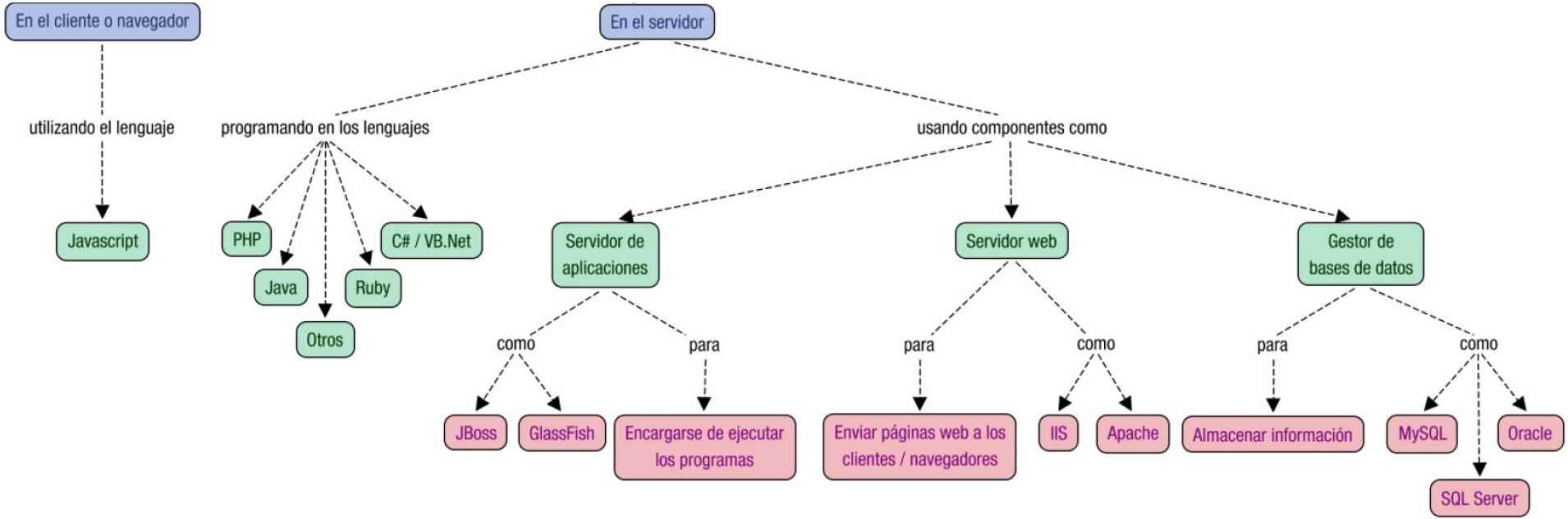
Página web	Lenguaje de programación del lado del cliente	Lenguaje de programación del lado del servidor
Google	JavaScript	C, C++, Go, Java, Python, PHP (HHVM)
Facebook	JavaScript	Hack, PHP (HHVM), Python, C++, Java, Erlang, D, XHP, Haskell
YouTube	JavaScript	C, C++, Python, Java, Go
Yahoo	JavaScript	PHP
Amazon	JavaScript	Java, C++, Perl
Wikipedia	JavaScript	PHP, Hack
Twitter	JavaScript	C++, Java, Scala, Ruby

RESUMIENDO



Luis Enrique González de la Fuente

PROGRAMACIÓN WEB



4.- SERVIDORES WEB

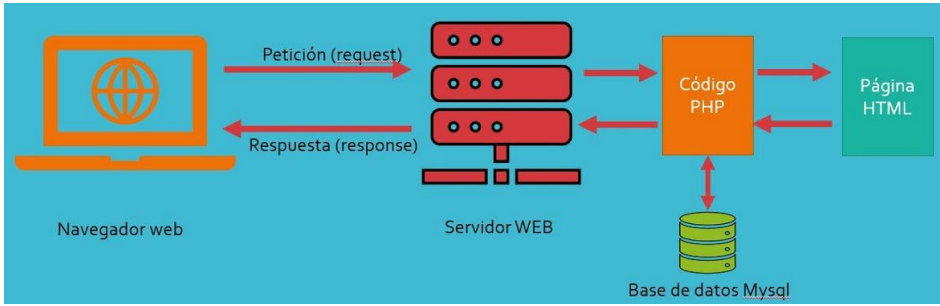
- Apache
- Microsoft IIS (Internet Information Services)
- Nginx (Nginx comercial)
- GWS (Google Web Server)
- Plataforma AWS (Amazon Web Services)

FUNCIONALIDADES DEL SERVIDOR WEB

- La principal función de un servidor Web es **almacenar** los archivos para poder transmitirlos a través de Internet y poder ser visitado por los usuarios. Cuando un usuario entra en una página de Internet, su navegador se comunica con el servidor enviando y recibiendo datos que determinan qué es lo que ve en la pantalla.
- Cada servidor Web posee una IP única, irrepetible, que lo identifica en la red.
- La capacidad de un servidor depende de su hardware, es decir, de los componentes que tenga la máquina en cuestión.

5.- INTEGRACIÓN CON LOS LENGUAJES DE MARCAS.

- PHP es un lenguaje de script del lado de servidor, multiplataforma y con una integración inmediata con HTML (aunque podría usarse con otros lenguajes de marcas como XHTML) y junto a cualquier fichero XML.
- Para utilizar PHP es necesario disponer de un navegador web, un servidor web que soporte PHP e instalar un intérprete PHP en la máquina en la que se desarrollará el código.
- Cuando el fichero PHP (con extensión .php) es solicitado al servidor web, el intérprete del mismo ejecuta las sentencias PHP transformándose en código HTML que luego será procesado por el navegador del cliente.
- Además PHP es capaz de autogenerar archivos HTML y guardarlos en el sistema de archivos en lugar de mostrarlos por pantalla.



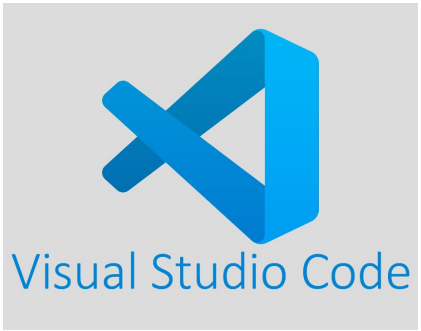
```php
<?php
  echo "|| 1 :> Aquí también puedo incluir PHP.";
?>
<html>
  <head>
    <title>Embebiendo PHP 1</title>
  </head>
  <body>
        <h2>EJEMPLOS DE COMO EMBEBER PHP EN HTML</h2>
        <h3>Primer caso: líneas de script PHP junto con código HTML.</h3>
        <p>Una forma de embeber sentencias de PHP dentro de HTML es<br />
        incorporando el código directamente entre las líneas de HTML.</p>
<?php
  echo "Ahora incluyo el código PHP entre las líneas de HTML.<br />\n";
  // Creo la variable $a y le asigno el valor 3
  $a = 3;

  if (!empty($a)){
        echo "La variable a guarda el valor: ".$a."<br />\n";
  }
  else{
        echo "La variable está vacía.<br />\n";
  }
?>
        <p>Esta es una forma!!<br />
        Puedo incluir sentencias de PHP fuera de las etiquetas <b>html</b>, como en 1 y 2!</p>
  </body>
</html>
<?php
  echo "|| 2:> Aquí también puedo incluir PHP.";
?>
```

6.- IDEs Y HERRAMIENTAS DE PROGRAMACIÓN.

Editores PHP:

- Sublime text.
- Notepad++.
- Visual Studio Code.



IDE PHP:

- Netbeans.
- Eclipse.
- PHPStorm.

IDEs Y HERRAMIENTAS DE PROGRAMACIÓN.

Entorno elegido:

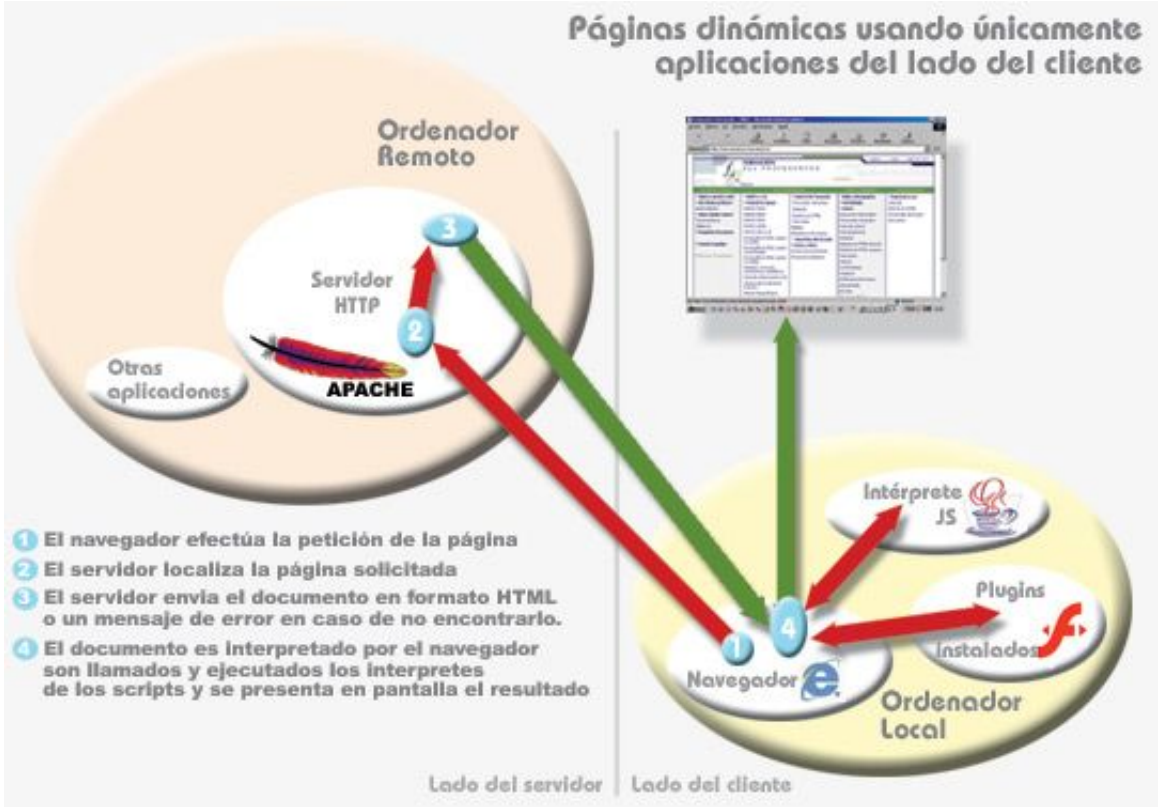
- PHPStorm.
- XAMPP.
- XDebug.

Un vídeo de preparación.

7. CÓDIGO EN LAS PÁGINAS WEB

- Comunicación entre extremos.
- Lenguajes embebidos en HTML: PHP, ASP, JSP, Servlets entre otros…
- Tecnologías asociadas (Javascript, BBDD, etc)
- Obtención del lenguaje de marcas a mostrar en el cliente.
- Etiquetas para inserción de código.
- Bloques de código.

COMUNICACIÓN ENTRE EXTREMOS

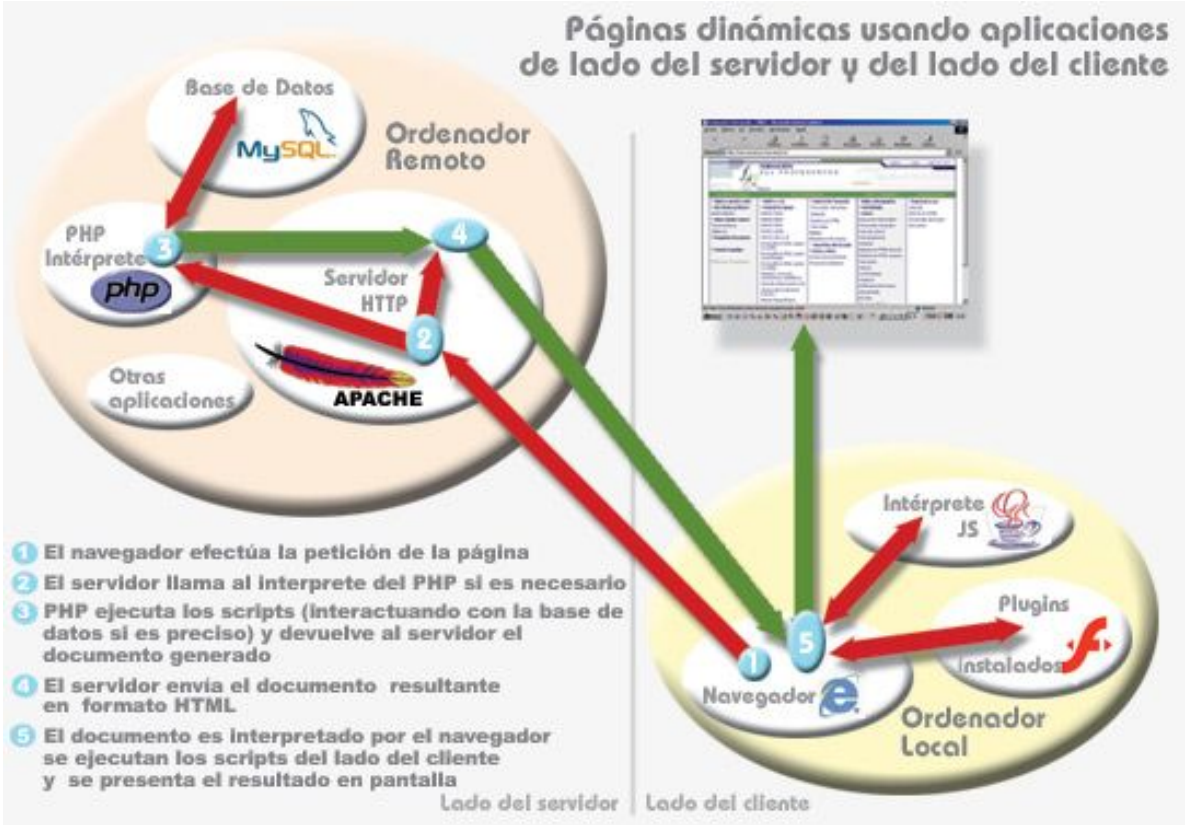


COMUNICACIÓN ENTRE EXTREMOS

Cuando nosotros pinchamos sobre un **enlace** hipertexto, en realidad lo que pasa es que establecemos una **petición** de un archivo HTML residente en el servidor (un ordenador que se encuentra continuamente conectado a la red) el cual es enviado e interpretado por nuestro navegador (en el cliente).

Por eso, podemos hablar de lenguajes de lado servidor que son aquellos lenguajes que son **reconocidos, ejecutados e interpretados por el propio servidor** y que se envían al cliente en un formato comprensible para él. Por otro lado, los lenguajes de lado cliente son aquellos que pueden ser directamente "digeridos" por el navegador y no necesitan un pretratamiento.

COMUNICACIÓN ENTRE EXTREMOS



COMUNICACIÓN ENTRE EXTREMOS

Tenemos claro que la comunicación entre cliente y servidor generará peticiones y respuestas entre ellos. La posibilidad de hacerlo depende de las capacidades que tenga el servidor web y los módulos o extensiones que tenga instalados.

El funcionamiento habitual es que el servidor, tras recibir la petición, cargue un archivo HTML estático y lo envíe hacia el cliente a través de la red.

Para comenzar el proceso, el cliente tiene que hacer una **petición** formal (con unas normas establecidas) a una dirección **URL** (Localizador Uniforme de Recurso) bajo un **protocolo** (FTP, HTTP) que también podrá ir acompañada de un **puerto** y una **ruta**.

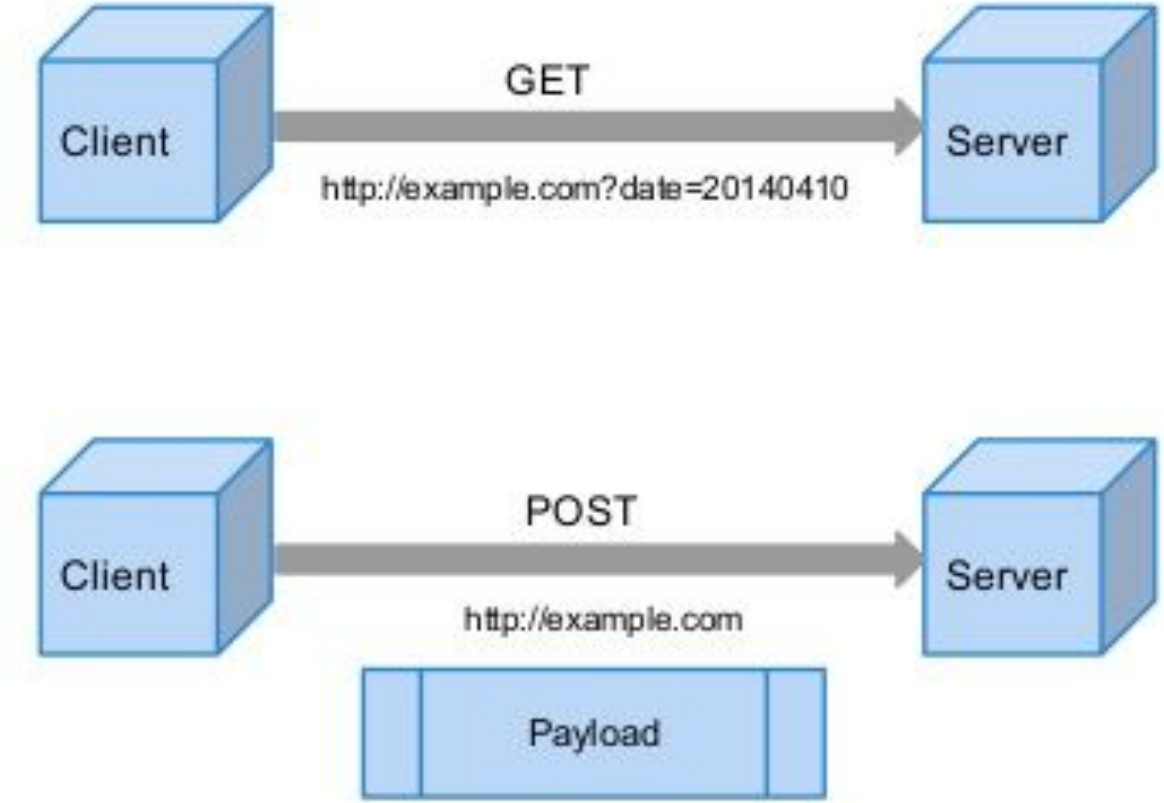
Dichas peticiones podrán seguir 2 métodos, GET y POST.

COMUNICACIÓN ENTRE EXTREMOS

Método **GET**: es un método de invocación en el que el cliente le solicita al servidor web que le devuelva la información identificada en la propia URL. Lo más común es que las peticiones se refieran a un documento HTML o a una imagen, aunque no es exclusivo. En tal caso, el servidor realiza las operaciones oportunas y le devuelve al cliente el resultado generado tras esa petición.

Método **POST**: mientras que el método GET se utiliza para recuperar información, el método POST se usa habitualmente para enviar información a un servidor web. Estos casos suelen darse al enviar el contenido de un formulario de autenticación, así como entradas de datos o especificar parámetros para algún tipo de componente ejecutado en el servidor.

COMUNICACIÓN ENTRE EXTREMOS. GET VS POST



LENGUAJES EMBEBIDOS EN HTML: PHP, ASP, JSP...

Lo primero es entender que un servidor web es un programa cuya misión última es servir datos en forma de documentos HTML (HyperText Markup Language) codificados en este lenguaje.

Por defecto, un servidor web se mantiene a la **espera de peticiones** HTTP realizadas por parte de un cliente (a través de un navegador web) “escuchando” en un puerto de comunicaciones (normalmente el 80).

En función de cómo se procesan las peticiones realizadas por un cliente web podemos distinguir distintos tipos de servidores web. Las diferentes estrategias de optimización de la ejecución y de diseño del servidor dan lugar a diferentes tipos de servidores web.

LENGUAJES EMBEBIDOS EN HTML: PHP, ASP, JSP...

- Servidores **basados en procesos**: al recibir una petición se clona un proceso que la atiende mientras el original sigue esperando más peticiones. (Apache)
- Servidores **basados en hilos**: consumen menos recursos porque se clonan en lugar de procesos completos y comparten el espacio de memoria (problema de seguridad con dicho espacio). (Apache)
- Servidores **dirigidos por eventos** (basados en sockets no bloqueantes): sockets en cliente y servidor que se comunican.
- Servidores implementados en el **núcleo del sistema** (kernel): el servidor web se ejecuta a nivel de sistema operativo para darle prioridad.

Alguna estadística: https://news.netcraft.com/

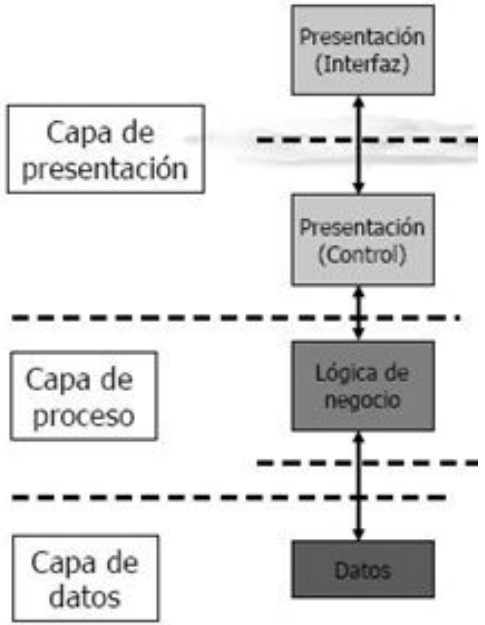
TECNOLOGÍAS ASOCIADAS (JAVASCRIPT, BBDD, etc)

Desde el punto del servidor, encontraremos una ejecución de código escrito en un lenguaje concreto que será responsable de generar la información necesaria en el cliente antes de enviarle la respuesta a la petición realizada..

Entre estos lenguajes podemos encontrar PHP, Python, ASP, JSP y objetos CGI y servlets entre otros.

PHP (“Hypertext Preprocessor” nacido en 1994) es un lenguaje de scripting de propósito general y de código abierto que ha sido especialmente diseñado (desde la versión 4) para el desarrollo de aplicaciones web y que puede ser embebido (intercalado) en código HTML.

El núcleo propiamente dicho de PHP maneja la configuración del entorno de ejecución (fichero php.ini) ofreciendo interfaces de programación (API PHP).



OBTENCIÓN DEL LENGUAJE DE MARCAS A MOSTRAR EN EL CLIENTE

Una vez recibida una petición de un cliente:

- El servidor Web analiza el código del documento que tiene que responder (analizador léxico y sintáctico) para generar instrucciones con el código intermedio.
- El intérprete PHP se encarga de traducir las instrucciones oportunas en código resultado.
- El servidor web envía el resultado al cliente en forma de HTML.

ETIQUETAS PARA LA INSERCIÓN DE CÓDIGO

Como ya hemos comentado, la programación en servidor mediante los lenguajes embebidos implica **insertar código** propio en páginas HTML a través de una serie de etiquetas especiales que delimitan los fragmentos de código que han de ser procesados por el servidor web.

Independientemente del lenguaje utilizado (sea PHP, ASP, JSP u otros) el componente del servidor encargado de procesar el código (intérprete para PHP) ignorará el código HTML que se encuentra fuera de dichas etiquetas.

EJEMPLO “HOLA MUNDO”:

```
<html>
	<head>
		<title>Primer ejemplo</title>
	</head>
	<body>
		<h1>Primer ejemplo: <br/></h1>
		<?php
			echo “Hola mundo.”;
		?>
	</body>
</html>
```

ETIQUETAS PARA LA INSERCIÓN DE CÓDIGO

CARACTERÍSTICAS:

- Todo script comienza y termina con una **etiqueta** de **inicio y** otra de **fin**. En PHP, estas etiquetas son “<?php” y “?>”, en ASP se utilizan <% y %> y en JSP <%= y %>.
- Los espacios en blanco que escribamos dentro del código embebido no tienen ningún efecto salvo para mejorar la legibilidad del código escrito.
- El código de servidor embebido en páginas HTML está formado por un conjunto de sentencias separadas. En PHP y JSP esta separación se realiza con un punto y coma (;) mientras que en ASP esta separación se hace mediante retornos de carro.
- Los scripts embebidos pueden situarse en cualquier parte del recurso web ejecutado y puede ser intercalado en cualquier fragmento de HTML. El número de scripts que podemos tener dentro de un fichero HTML es indefinido.
- Cuando se ejecuta un código embebido, el script entero se sustituye por el resultado de dicha ejecución, incluidas las etiquetas de inicio y fin.

BLOQUES DE CÓDIGO

Por **ejemplo** en este código PHP podemos observar diferentes partes de un fichero HTML con PHP intercalado.

En este caso podemos ver cómo hay tres fragmentos de código. El primero define el contenido de una variable ($salida) y los otros dos lo que hacen es escribir el valor de dicha variable entre el código HTML para indicar tanto el título de la página como el contenido de la misma.

```
<html>
    <?php
        $salida = “Contenido PHP”;
    ?>
    <head>
        <title>
            <?php
                echo $salida;
            ?>
        </title>
    </head>
    <body>
        <h1>Segundo ejemplo: <br /></h1>
        <?php
            echo $salida;
        ?>
    </body>
</html>
```

EJERCICIOS

- 1. Crea un fichero que contenga las etiquetas HTML mínimas para mostrar un mensaje de texto (<html>, <head>, <title> y <body>) y guárdelo con extensión PHP. A continuación escribe una sentencia echo o print que permita mostrar por pantalla el mensaje “Este es el resultado correcto del primer ejercicio”. Abra un navegador y compruebe el resultado.
- 2. Crea un fichero PHP que muestre por pantalla el mensaje “Segundo ejercicio: visualización del contenido de variables”. A continuación, define dos variables $nombre y $edad y asígnales un valor correcto. Después, cree una sentencia que muestre un mensaje que contenga dichas variables, similar a “Mi nombre es: valor_de_la_variable_$nombre y mi edad es valor_de_la_variable_$edad”.

EJERCICIOS

- 3. Crea un fichero PHP que permita comprobar las capacidades aritméticas de PHP. Para ello, crea dos variables $n1 y $n2. Asígnale los valores 13 y 4, respectivamente. Defina una tercera variable $resultado y escriba un código que permita hacer y mostrar las siguientes operaciones: 13 – 4, 13 + 4, 13 * 4, 13 / 4 y 13 % 4
- 4. Crea un fichero de PHP que permita conocer toda la información de una variable (función var_dump()) y también el tipo de dicha variable (función gettype()) probando a asignar diferentes valores numéricos, letras, “false” y null.