UD1: Aplicacións web e linguaxes de marcas: HTML e CSS

ACTIVIDADE 1: Aplicacións web. Introdución as linguaxes de marcas.

Índice

1. Internet e a World Wide Web

A World Wide Web ten un gran impacto na nosa vida dende que apareceu en 1990. Nesta unidade didáctica, antes de aprender a manexar a linguaxe de marcas HTML, imos ver unha introducción a tecnoloxía Web.

1.1 ¿Qué é unha aplicación Web?

As aplicacións forman parte da nosa vida cotiá independentemente do hardware que empreguemos para executala (smartphones, tablets, ordenadores de escritorio,..). Distinguimos as ***aplicacións nativas,*** *que son* un tipo de software que se adapta a unha plataforma determinada, das ***aplicacións*** que se executan nun navegador web.

Unha web app (aplicación web en español) está basada en HTML, JavaScript e CSS. Posto que se carga no servidor web e se executa no navegador, non require ningunha instalación. Ademais, tamén se pode crear un acceso directo para ela no escritorio do ordenador ou na pantalla de inicio dos terminais móbiles mediante un marcador. Moitos programas e servicios preséntanse nas dúas *modalidades*: en forma de web app ou de native app.

As principais diferencias entre unha aplicación web e unha aplicación nativa son:

- Unha aplicación nativa prográmase tendo en conta as características da plataforma na que vai ser instalada, co cal só funcionan nesa plataforma en concreto.

  No caso de ter un fallo de seguridade é imprescindible descargar a nova versión ou facer a actualización para solucionalo.

- Unha aplicación web non se adapta tan ben ao hardware do dispositivo no que se usan pero, no mellor dos casos, unha única aplicación é suficiente para todas as plataformas (inda que non sempre é posible optimizar a aplicación para todos os navegadores.

  Cando hai un fallo de seguridade, este impleméntase directamente no software, co cal todos os usuarios desfrutan da versión máis segura.

Hoxe en día óptase en moitos casos por aplicacións híbridas, que son aplicacións móbiles deseñadas nunha linguaxe de programación web como HTML5, CSS ou JavaScript, xunto cun framework (entorno de traballo) que permite adaptar a vista web a calquiera vista dun dispositivo móbil.

As aplicacións web actuais teñen expectativas moi altas por parte do usuario. Espérase que estean dispoñibles 24 horas ao día os sete días da semana desde calquera lugar do mundo y que se podan usar desde practicamente calquera dispositivo ou tamaño de pantalla.

As aplicacións web deben ser seguras, flexibles e escalables para satisfacer os picos de demanda.

As principais características dunha aplicación Web son as seguintes:

- Unha aplicación Web permite distribuír información por Internet.
- Unha aplicación Web xestiona a concorrencia, permitindo o acceso a un mesmo recurso baseado na Web a múltiples usuarios.
- Unha aplicación Web pode xerar contido dinámico, construíndo páxinas Web sobre a marcha a partir de fontes de datos que poden ser alimentadas polos usuarios da mesma.

- Unha aplicación Web pode incluír seguridade baseada en declaracións ou en roles que nos permitan conceder ou denegar o acceso aos recursos específicos aos usuarios en función do seu rol.
- Unha aplicación Web pode conectarse a unha base de datos de xeito que os seus contidos poden almacenarse de xeito persistente e ser recuperados cando sexa necesario.
- Unha aplicación Web utiliza os servizos transaccionais dunha base de datos de modo que as súas actualizacións sexan fiables e consistentes.
  - Se o contido dunha aplicación Web se almacena nunha base de datos, temos que asegurar que, si se fan cambios polos usuarios, estes se xestionen de modo correcto, o que implica que teremos que dispor dun sistema de transaccións.
- Unha aplicación Web pode executarse en distintas máquinas, permitindo a escalabilidade do sistema.
  - En moitos casos é recomendable dispor de máis dunha copia da aplicación de modo que si varios clientes queren acceder ao mesmo tempo ou a máquina falla, podamos seguir dando o servizo necesario aos nosos clientes.

As aplicacións Web non poden funcionar de forma illada debido a que, a súa vez, empregan as tecnoloxías de Internet e da World Wide Web. A continuación, imos ver algunhas das características fundamentais de ditas tecnoloxías.

1.2 Servizos web 1.0, web 2.0, web 3.0 e web 4.0

A tecnoloxía avanzou máis nos últimos 30 anos que en toda a historia da humanidade; pero, o que realmente cambiou as nosas vidas por completo foi a web.

A Web 1.0 empezou nos años 60 con navegadores de só texto, como ELISA, despois apareceu HTML que fixo as páxinas máis agradables á vista e os primeiros navegadores visuales como Netscape e Internet Explorer.

Caracterízase principalmente por ser unidireccional e basada en ***contidos estáticos***. As primeiras páxinas publicadas en Internet constaban de contidos de texto que, unha vez publicados, non se actualizaban salvo que o "webmaster" modificase ditos contidos e volvese subir a web de novo a internet.

A web 1.0, que se desenvolveu a finais do século XX, tiña un carácter principalmente divulgativo, e empezaron a colgarse de internet documentos e información principalmente cultural.

A web 2.0, que aparece na primeira década deste século, supón unha importante evolución converténdose nun fenómeno social no que millóns de usuarios intercambian ideas e contidos en foros, blogs e redes sociais.

As páxinas web tradicionais ofrecían a mesma información para todos os usuarios, cos *servizos web 2.0 varían para cada usuario e incluso fan ao usuario partícipe do contido*.

A web 3.0 é a que empregamos a día de hoxe e é coñecida como *a "web semántica"*. Esta consiste nunha nova concepción do entorno web para un uso máis natural por parte dos usuarios, desaparecendo a busca por *palabras clave* para dar paso a buscadores que se

atopan máis preto de entender a linguaxe natural humana. Trátase de usar unha linguaxe similar á que se emprega na vida cotiá, o que conleva ademais a aparición de software que permita codificar correctamente esta información para devolver resultados o máis personalizados posible.

Outro cambio moi significativo é a **mobilidade** no seu uso. Na actualidade, unha grande parte do uso de Internet faise a través de dispositivos móbiles, e ferramentas como "a nube" diversifican as súas opcións de uso. Por elo, tamén poderiamos denominar á web 3.0 como a "web móbil" ou a "web flexible".

A Web 4.0 leva connosco dende 2016, pero terán que pasar varios anos para que se afiance e comece a mostrar todo o seu potencial. Esta nova versión da Rede basearase en explotar toda a información que agora mesmo conten, pero dunha forma máis natural e efectiva empregando Intelixencia Artificial.

A Web 4.0 propón un novo modelo de interacción có usuario máis completo e personalizado, comportándose como un espello máxico que de solucións concretas as necesidades do usuario.

Tecnoloxías como Siri, Cortana ou Alexa baséanse neste tipo de sistemas e podemos conseguir, mediante unha simple frase, que nos reproduza unha canción ou nos compre un billete de avión. O mellor, e o peor disto, é que os nosos dispositivos aprenden de todo o que dicimos, de xeito que coñecen por completo o noso estilo de vida, gustos, horario e demais aspectos do noso día a día.

Pero non podemos falar da web 4.0 sin falar do ***Metaverso***. a web 4.0, tamén coñecida como web simbiótica, busca desdibuxar a liña que separa ao se humano das máquinas, creando un mundo virtual onde podamos sumerxirnos por completo e vivir a diario sen saír de casa.

¿Chegará un día no que un dispositivo saiba o que queremos antes que nos?

1.3 Aplicacións na nube (cloud computing)

É outro dos termos fundamentais para definir os servizos ofrecidos polas páxinas actuais. Basease en ofrecer ao usuario servizos de modo que o usuario poda acceder a eles dende calquera dispositivo conectado a Internet.

Basease na programación distribuída de aplicacións, pero a tal nivel que as aplicacións distribúense incluso por centos de servidores situados en distintas partes do planeta e así poder responder a unha demanda inxente de peticións de servizo cunha altísima capacidade de tolerancia a fallos.

Ao usuario bástalle un navegador ou unha pequena aplicación (como una App dun dispositivo móbil) para acceder e empregar o servizo.

A computación na nube apareceu para responder tecnoloxicamente aos retos de empresas como Google ou Facebook que tiñan que atender a gran velocidade as peticións de millóns de usuarios. Hoxe en día o concepto permite que se utilice Internet como base de traballo, substituíndo así ao propio ordenador persoal que pode ser un equipo de menor potencia ao delegar na nube o proceso das tarefas e o almacenamento da información.

Exemplos de servizos na nube serían: discos duros virtuais, copias de seguridade en liña, aplicacións de ofimática web, redes sociais, bibliotecas multimedia como Youtube ,...

1.4 Tecnoloxías de Internet e a World Wide Web

As aplicacións Web baséanse tanto en Internet como na World Wide Web para funcionar, e algo que temos que ter en conta é que ambas non son a mesma cousa, xa que a primeira versión de Internet precede á World Wide Web en dez anos.

Internet é a rede de redes que xurdiu a partir do proxecto ARPAnet, que comezou nos anos setenta. Pola súa parte, a World Wide Web (de xeito abreviado Web ou W3) é unha colección de información multimedia baseada en hipertexto que é accesible a través de Internet e que data dos noventa (por hipertexto se entende ao contido na web e está enlazado de modo que se pode navega de xeito sinxelo entre as distintas páxinas Web, que poden estar situadas fisicamente en calquera parte do mundo).

Podemos dicir, que Internet é a autopista da información e a World Wide Web é o tráfico que circula por ela.

Tecnoloxías importantes de Internet

A WWW depende de certas tecnoloxías para o seu correcto funcionamento. Entre elas destacamos:

TCP/IP (Transmisión Control Protocol/Internet Protocol)

TCP/IP é na actualidade un conxunto de protocolos relacionados e ferramentas que axudan aos equipos a comunicarse entre eles.

Direccións IP

A dirección IP permite que unha máquina poda conectarse a outras sempre que se encontre na mesma rede física.

Nomes de dominio

A maioría dos equipos que albergan sitios Web empregan nomes de dominio en lugar de direccións IP. Ésto implica que os usuarios poden, por exemplo, acceder a www.edu.xunta.es en lugar de utilizar a dirección IP real. O sistema de dominios DNS (Domain Name Server) é o que se encarga de que o nome de dominio se traduza nunha dirección IP válida.

Tecnoloxías WWW importantes

HTTP (HyperText Transfer Protocol)

A WWW utiliza HTTP para enviar información. Cando se usa este protocolo, o nome de dominio (ou a dirección IP) vai precedida de http:// (por exemplo: http://edu.xunta.es)

Os navegadores web normalmente teñen en HTTP o seu protocolo por defecto e o inclúen por defecto nas direccións que introducen os usuarios. HTTP é un protocolo de tipo solicitude-resposta. Os clientes (normalmente un navegador) envían unha petición ao servidor Web, que é un software que contén información Web e que a proporciona como resposta ás peticións dos clientes normalmente en forma de páxina codificada en HTML para que o navegador a poda interpretar.

HTTPS é un protocolo que proporciona seguridade na transmisión de datos, impedindo que calquera poda lee ra información.

URL (Uniform Resource Locators)

URL (Localizador Uniforme de Recursos) é a especificación completa das ubicacións dos recursos de Internet. Unha URL comprende os seguintes elementos:

- O protocolo da petición (o protocolo por defecto do navegador normalmente é http://)
- A dirección IP ou o dominio do servidor
- O número de porto (porto 80 para HTTP e 443 para HTTPS).
- A ruta do subdirectorio a partir da raíz do documento (se procede)
- O nome do recurso (aínda que sempre hai unha páxina por defecto que se carga se non se especificou ningunha outra.

A seguinte URL inclúe todos estes elementos:

```
http://www.edu.xunta.es:80/fp/index.htm
```

Debido a que o protocolo e o porto son os que figuran por defecto no navegador, e que o recurso que figura é o que ten o dominio por defecto, isto sería equivalente a escribir:

```
www.edu.xunta.es/fp/
```

2. Tipos de páxinas web

Unha sinxela clasificación dos tipos de páxinas web podería ser a seguinte:

- Páxinas estáticas
- Páxinas dinámicas

Páxinas estáticas

Diremos que unha páxina é estática cando os seus contidos non poden ser modificados – nin dende o servidor que a aloxa (ordenador remoto) nin dende o cliente (navegador)– mediante ningunha intervención do usuario nin tampouco a través de ningún programa.

Un exemplo de páxina estática sería:

```
<html>
  <head></head>
  <body>
      Hoxe é 4-2-2019 e son as 14:23:57 horas
  </body>
</html>
```

Calquera usuario que acceda a esta -xa sexa en *modo local*, ou a través dun *servidor remoto*– visualizará sempre a mesma data: *4 de febreiro de 2019*.

Páxinas dinámicas

Chamaremos dinámicas ás páxinas que poden ver modificados os seus contidos – de forma automática ou mediante a intervención do usuario – ben sexa dende o cliente e/ou dende o servidor.

Para que esas modificacións poidan producirse é necesario que *algo ou alguén* especifique: ***qué*, *cómo*, *cando*, *ónde* e *de qué xeito*** deben realizarse, e que exista outro *algo o alguén* capaz de acceder, interpretar e executar tales instruccións no momento preciso.

Scripts

Chámase *script* a un conxunto de *instruccións* escritas nunha *linguaxe* determinada que van **incrustadas** dentro dunha *páxina WEB* de modo que o seu *intérprete* poda acceder a elas no momento no que se requira a súa *execución* (por exemplo, cando se carga a páxina, ao mover o ratón, ao pulsar un botón,...)

Cando se **incrustan** *scripts* nunha *páxina WEB* empezan a *convivir* nun mesmo documento informacións destinadas a distintos *intérpretes*. Por unha parte, o *código HTML* que vai ser *interpretado* polo *navegador*, e pola outra, os *scripts* que van ser *executados* –dependendo da linguaxe na que foron escritos– polo **seu** *intérprete* correspondente.

O xeito de diferenciar os contidos é delimitar os scripts *marcando* o seu comezo cunha etiqueta de *apertura* <script> e sinalando o final cunha etiqueta de *peche* </script>. O que non está contido entre esas etiquetas considerarase *código HTML*.

Pódese **inserir** nun mesmo documento *scripts* desenvolvidos en distintas *linguaxes*. indicandoo nun atributo language: <script language="JavaScript"> </script>

Se non hai este atributo enténdese que a linguaxe é javascript . Isto permite para que no momento en que vaian ser *executados* se invoque o *intérprete* adecuado.

Por exemplo:

```
<script>
  ......
  ...... instruccións..
  ......
</script>
```

estaríamos sinalando que nas instruccións contidas no script usase a sintaxe de JavaScript.

Para o caso concreto de PHP, existe unha sintaxe alternativa:

```
<?php
......
...... instruccións..
......
?>
```

No seguinte exemplo verás que a data que aparece na páxina é a *data actual* do teu sistema, e ademáis, cada vez que premas o botón *Actualizar* do teu navegador poderás comprobar que *se actualiza* a hora.

Unha *intervención* do usuario *modifica os contidos*.

```
<html>
 <head>
     <script>
       var son= new Date();
       var data=son.getDate()+" - "+(son.getMonth()+1)+" - "+son.getFullYear();
       var hora=son.getHours()+":"+son.getMinutes()+":"+son.getSeconds();
       document.write('Hoxe é '+data+' e son as '+hora+' horas');
     </script>
 </head>
 <body>  </body>
</html>
```

3. Servidores e clientes

Podemos comparar un **host,** ou servidor, cun servizo de comida a domicilio.

Cada empresa pode **atender peticións dun único** ou de **varios** servizos distintos (*pizzas*, *xeados*, ou *pratos rexionais*, por citar algúns exemplos), pero a oferta de *cada un*

deses servizos **require** unha infraestructura *adecuada a cada caso*. A oferta de *pizzas* esixirá dispoñer dun *forno*, e a de *xeados* necesitará dunha *instalación frigorífica*.

Pois ben, algo moi similar ocorre cos **host**. Tamén estes poden ofrecer *un* ou *varios* servizos (páxinas web, correo electrónico, transferencias FTP, noticias, etcétera) e tamén é necesario que cada servizo dispoña da súa propia infraestructura, que neste caso sería **un programa distinto** (*software de servidor*) para cada un deles.

Como podes ver, **non basta** con falar de **servidores**. É necesario especificar tamén *qué é o que serven*, para o cal habería que dicir: **servidor de páxinas web**, **servidor de correo**, ... e ter presente que –inda que *convivan* no mesmo **host**– cada un deles require o seu propio software e a súa propia configuración.

Ao programa que utilizamos para realizar cada **petición** chamarémoslle **cliente**.

¿Qué é unha petición?

Unha **petición** é un conxunto de datos que un **cliente** (recorda que o cliente sempre é un dos programas instalados no teu ordenador) envía a través de Internet solicitando unha resposta determinada por parte dun ordenador remoto.

Cada tipo de **petición** terá contidos distintos. Por exemplo, cando se trata de *ler mensaxes de correo*, a petición realizada polo **cliente** (*Correo*) contería, entre outros, moitos dos datos da configuración da conta, tales como: o **protocolo** (forma de comunicación) – no caso do correo o habitual sería o protocolo **POP** (**P**ost **O**ffice **P**rotocol)–, o nome de **host** onde está aloxado o *buzón* (servidor POP ou servidor de correo entrante), o nome da conta, a contrasinal, e algunhas outras informacións relativas á xestión desa conta tales como se deben conservarse ou non as mensaxes no servidor, etcétera.

¿Qué ocorre con esa petición?

Calquera **petición** pasa en primeira instancia por un *servidor de nomes de dominio* (Domain Name Server) **DNS**, unha especie de guía telefónica que contén os nomes dos servidores e as direccións IP a través das cales están conectados a Internet.

Unha vez *resolta* esa petición polo *servidor* **DNS** (*direccionamento* da *petición* á IP correspondente) comprobarase se esa IP está activa (se existe un ordenador conectado a través dela) e, en caso de estalo, determinarase se ese ordenador ao que estamos accedendo *é capaz* de **atender a petición** para o que deberá ter **instalado e funcionando** o **software de servidor adecuado ao protocolo da nosa petición**.

Cando o *ordenador remoto* **acepta a petición** o *software de servidor* e/o as **aplicacións do lado do servidor resolven a petición** (*comproban que o nome da conta e a contrasinal son correctas*, *comprobar se existen mensaxes*, *borralos do buzón ei así o especifica a petición*, etc.) e **devolven ao cliente** (recorda que o *cliente* era noso *Correo*) a información requirida.

So falta que unha vez recibida a **resposta** *Correo* (**cliente**) **interprete** a información recibida e nos permita visualizar ou imprimir o contido das mensaxes *descargadas* do servidor.

Servidor e cliente nunha mesma máquina

Ata agora -ao referirnos a servidores e clientes– fixemos alusión a dúas *máquinas*: o noso propio ordenador (*ordenador local*) no que estarían instaladas as aplicacións **cliente** e un *ordenador remoto* no que se aloxarían as aplicacións de **servidor**. Eso é o máis *habitual*, pero *non é a única posibilidade*. Dado que **servidor** e **cliente** son unicamente **aplicacións** é perfectamente posible que ambas *convivan* dentro da mesma *máquina*.

A diferencia substancial sería que agora non é necesario o servidor de DNS para buscar a dirección IP. Utilizaríamos unha IP (habitualmente a **127.0.0.1**) **reservada** para estes casos –preestablecida na configuración do servidor– e a través dela se canalizarían as *peticións* ao *noso propio servidor*.

3.1 Servidores de aplicacións web

Os servidores web son os encargados de recibir as peticións referidas a páxinas ou elementos da web a través do protocolo http. Normalmente é un software aloxado nun ordenador servidor.

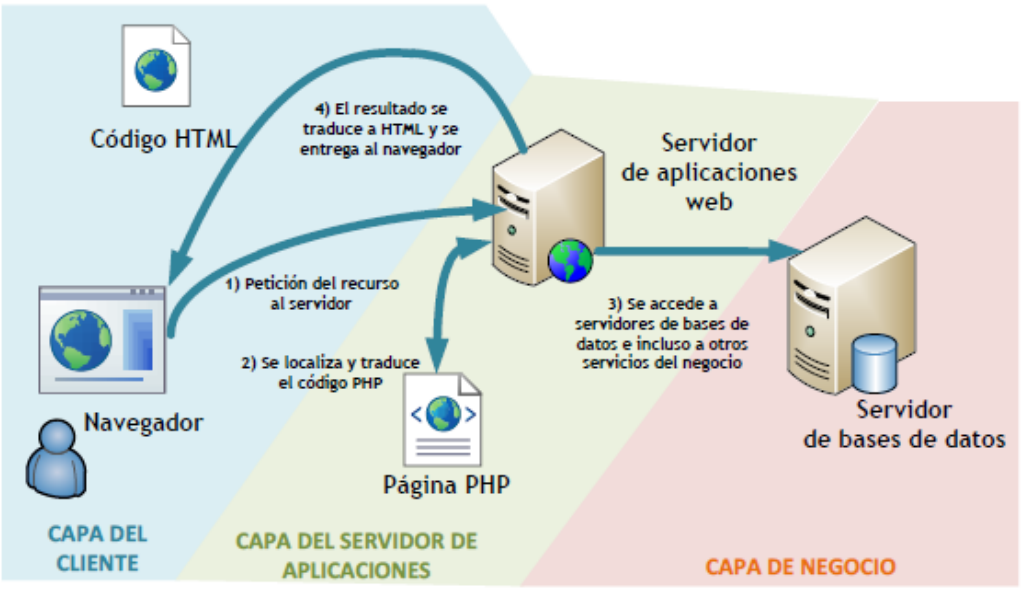
Os servidores de aplicacións podemos velos como unha ampliación dos anteriores. Son servidores web, pero que teñen capacidade de almacenar e xestionar aplicacións web. Entendendo que unha aplicación web é un servizo ao que os usuarios acceden a través da web.

Este tipo de servidores non só serven para atender peticións http, senón que ademais son capaces de entender instrucións de linguaxes avanzados da web e traducilas ou ben son capaces de acceder a recursos de outros servidores. Ese proceso faise de forma transparente ao usuario, é dicir, o usuario pide o servizo a través do seu navegador e o servidor de aplicacións atende a petición e interpreta o código da aplicación co fin de traducilo e mostrarlle ao usuario o resultado de forma entendible polo seu navegador (é dicir en formato HTML).

Ao xeito de traballar dun servidor de aplicacións, coñéceselle normalmente como arquitectura de ***tres capas*** (as veces fálase de máis capas). Unha primeira capa é a do navegador que é capaz de traducir código do lado do cliente (HTML, JavaScript, CSS, Flash,…). Para elo esa capa debe dispor de todos os compoñentes necesarios para facer esa labor no ordenador do usuario.

A segunda capa fórmana o servidor de aplicacións na súa labor de traducir código no lado do servidor (JSP, PHP, ASP, Perl…) e convertilo ao formato entendible polo navegador.

A terceira capa son todos os servizo aos que accede o servidor de aplicacións para poder realizar a tarefa encomendada á aplicación (por exemplo o acceso á base de datos):

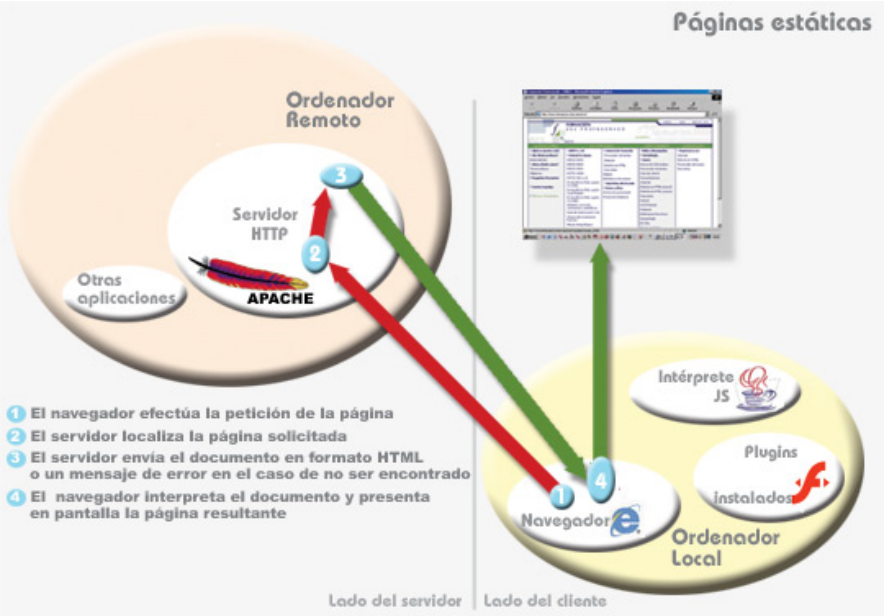


Exemplos de servidores web son Apache, que é o servidor web máis empregado; Nginx y Microsoft IIS.

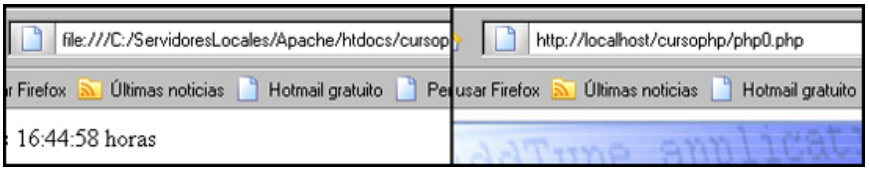
3.2 Esquemas de diferentes *peticións* de páxinas WEB

Intentaremos resumir de forma esquemática os procesos dalgúns dos diferentes tipos de *peticións* de páxinas WEB.

Aquí temos a máis sinxela delas:



Se observas con detemento o esquema la parte superior é posible que encontres algo *que non te cadre*... porque no esquema *hai un servidor* que *parece imprescindible* para atender as peticións e sen embargo ti –**sen ter instalado ningún servidor**– és capaz de visualizar as túas propias páxinas web facendo *dobre click* sobre a súa icona. Eso é certo, pero fíxate nas dúas direccións que aparecen nesta outra imaxe:

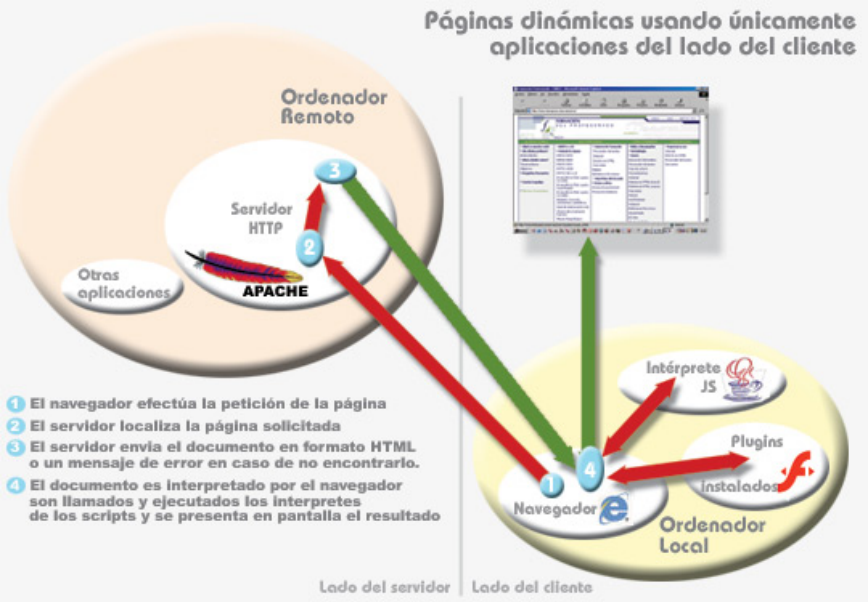


A da esquerda –consecuencia de ter feito dobre click sobre a icona do documento– contén como dirección unha *ruta* (o *path* que conduce ata o documento) mentres que na da dereita aparece o sintagma **http** ao principio da dirección.

No primeiro caso non fixemos ningunha *petición de páxina web,* senón que abrimos un documento con extensión (html) asociada con Internet Explorer na nosa configuración de Windows. O proceso é o mesmo que se houberamos feito dobre click sobre a icona dun documento con extensión **txt**, coa única salvedade de que neste último caso abriríase o *bloc de notas* (pola asociación de extensións e aplicacións na configuración de Windows).

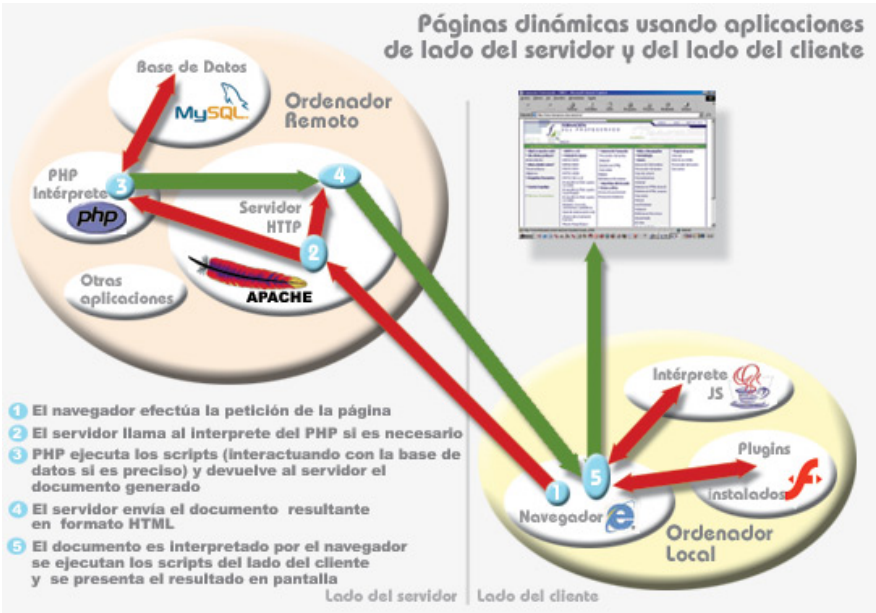
No segundo caso as cousas son distintas. Inclúese o *sintagma* **http** – acrónimo de **H**iper**T**ext **T**ransfer **P**rotocol– para indicar que ese é o **protocolo** que debe ser utilizado e que será preciso que o servidor que reciba a **petición** *sexa capaz de interpretalo*. Por iso aos servidores que aloxan páxinas web adoitáselles chamar **servidores HTTP** e requireselles que *soporten* este protocolo.

Seguindo cos esquemas, o seguinte é o correspondente a unha petición de páxina na que hai *incrustados scripts* escritos en *linguaxes do lado do cliente*:



Como podes observar non require *nada distinto* ao do suposto anterior. A diferenza sería que neste caso se f*arían chamadas* ao *intérprete de JavaScript* –incluído nos navegadores e/ou a eventuais *plugins* necesarios para interpretar outros tipos de *script*.

E por último, o esquema máis complexo: un exemplo de *convivencia* nun mesmo documento de varios *scripts* e varios tipos de linguaxe:



Aquí xa é preciso que, ademais dun servidor capaz de soportar o protocolo **HTTP**, estea instalado –do lado do servidor– un intérprete PHP, un **servidor de bases de datos MySQL** e que, ademais, estean configurados de xeito que poidan interactuar entre eles.

A linguaxe **PHP** dispón de funcións que lle permiten acceder a moi diversos tipos de *servidores de bases de datos* poidendo: **crear**, **consultar**, **borrar** e **modificar** tanto bases de datos como táboas e rexistros das mesmas.

4. Linguaxes

Hai múltiples posibilidades en cuanto a linguaxes de *script*. Pero antes de facer mención a algúns deles é convinte facer unha clasificación previa.

As linguaxes de *script* poden clasificarse en dous tipos: linguaxes do lado do cliente e

do lado do servidor.

Linguaxes do lado do cliente

Un linguaxe é *do lado do cliente* cando o *intérprete* que vai *executar* os seus *scripts* é **accesible** dende este –o *cliente*– sen que sexa necesario facer ningunha *petición* ao *servidor*.

Seguramente terate ocorrido algunha vez que ao intentar acceder a unha *páxina web* aparece unha mensaxe dicindo que *a correcta visualización da páxina require un* ***plug-in*** *determinado*, e que, á vez, ofréceche a posibilidade de *descargalo* nese momento.

Eso ocorre porque cando o *navegador* –que no caso das *páxinas web* é o *cliente*– trata de *interpretar* a páxina, encontra **incrustado** nela *algo* (un ficheiro de son, unha animación *Flash*, etcétera) que –de forma moi similar ao que ocorre cos *scripts*– require un *intérprete adecuado* do que non dispón nese momento.

Linguaxes do lado do servidor

Unha linguaxe é *do lado do servidor* cando a execución dos seus *scripts* se efectúa **antes** de *dar resposta á petición*, de xeito que o *cliente* non recibe o documento orixinal senón o resultante desa interpretación previa.

Cando se usan estes tipos de linguaxe o *cliente* recibe un documento no que cada *script* contido no orixinal foi *substituído* polos resultados da súa execución.

Isto é algo a ter moi en conta, porque, neste caso, os usuarios **non terán** a posibilidade de **visualizar o código fonte**, mentres que cando se trata de *linguaxes do lado do cliente* sempre é posible visualizar os *scripts*, ben sexa de forma directa –mirando o código fonte da páxina recibida– ou *lendo* o contido de ficheiros externos –vinculados a ela– que son bastante fáciles de encontrar na *cache* do navegador.

¿Cómo sabe o que ten que facer o servidor?

Dado que nuns casos o servidor debe *entregar* o **documento orixinal** –páxinas estáticas ou páxinas dinámicas nas que se usan *linguaxes* do *lado do cliente*– mentres que noutros casos –páxinas dinámicas usando linguaxes do *lado do servidor*– ten que devolver o **resultado** da execución dos *scripts*, é razoable que te preguntes: ¿cómo *sabe* o servidor o que debe facer en cada caso?

A resposta é sinxela: *indícaselle* simplemente poñendo unha extensión ao documento.

Se na *petición* se alude a un documento con extensión **.htm** ou **.html** o servidor entenderá que esa páxina non require a intervención previa de ningún *intérprete* do *seu lado* e entregará a páxina *sen manipular*.

Se nesa petición se alude a unha extensión distinta –**.php**, por exemplo– o servidor entendería que *antes de servir a páxina* debe *lela* e requirir ao intérprete de PHP que execute os scripts desenvolvidos nesa linguaxe (en caso de que os contivera) e devolvería ao cliente o **documento que resultara** da execución de ditos scripts.

4.1 Exemplo dunha páxina PHP

Observemos este código fonte. Trátase dunha páxina web moi simple que **non contén ningún script PHP**.

Se gardamos esta páxina co nome **exemplo4.html** e a volvemos gardar –sen modificar nada nos seous contidos– como **exemplo4.php** e visualizamos ambos exemplos veremos que o resultado é idéntico.

```
<html>
 <head> <title>A miña primeira páxina PHP</title> </head>
 <body>
      Esta é unha páxina sinxelísima
 </body>
</html>
```

Se modificamos a páxina anterior (*exemplo4.php*) e lle engadimos a nosa primeira etiqueta PHP (exemplo5.php) teremos o noso primeiro script PHP. Este sería o *código fonte*:

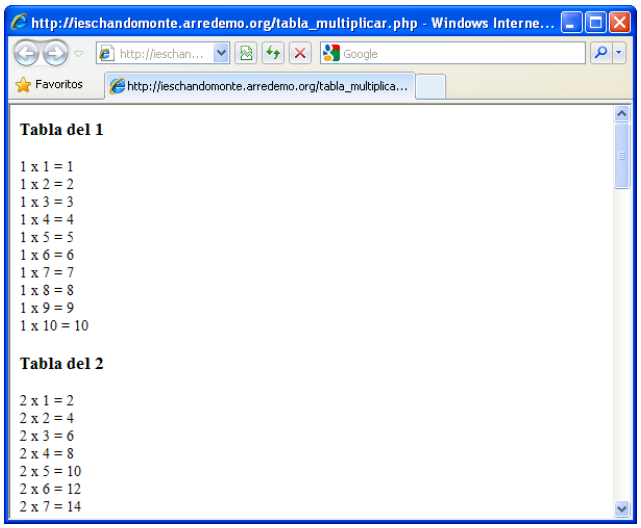
```
<html>
  <head> <title> O meu primeiro script PHP</title> </head>
  <body>
      Esta é unha páxina sinxelísima
    <?
      echo "¿Aparecerá esta liña?";
   ?>
  </body>
</html>
```

Outro exemplo de script PHP que visualiza a táboa de multiplicar é (Taboa_multiplicar.php):

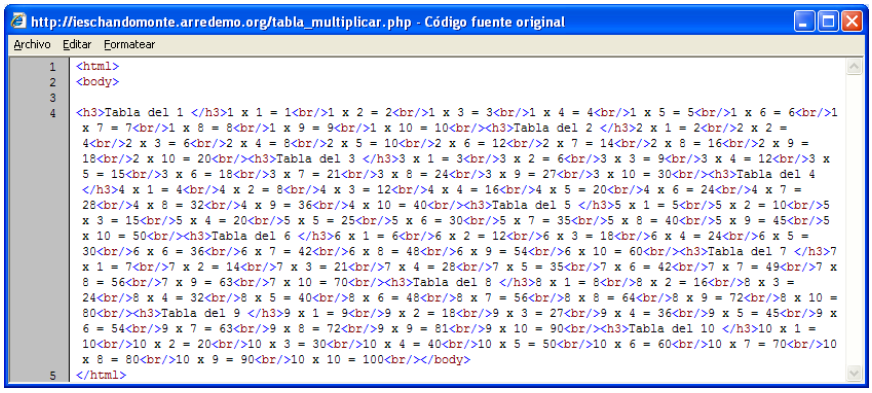
```
<html>
 <body>
     <?php
        /* Programa para imprimir las táboas de multiplicar desde la 1 a la 10 */
       for($tabla=1; $tabla<=10; $tabla++) //<-un ciclo de 10 (uno para cada tabla)
       {
                  echo "<h3>Tabla del $tabla </h3>";

                   // generamos la tabla
                   for($i=1; $i<=10; $i++)
                  {
                     echo "$tabla x $i = ". ($tabla*$i) . "<br/>";
                  }
     }
   ?>
  </body>
</html>
```

Visualizarase a táboa de multiplicar como se ve na seguinte imaxe:



Nembargantes, se visualizamos o código fonte non vemos o código escrito en linguaxe PHP, senón o resultado do procesamento de instruccións que se levou a cabo no servidor para transformalo nunha páxina capaz de visualizarse no navegador do cliente:

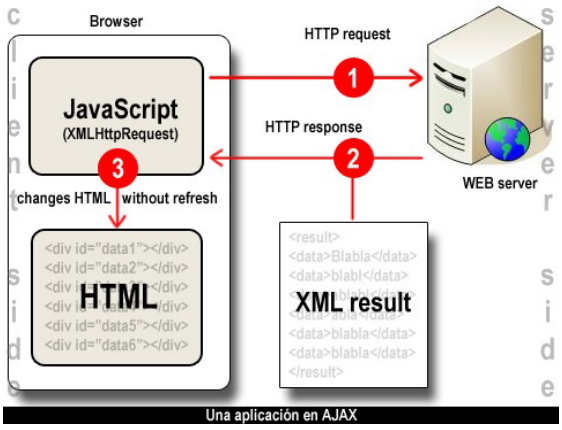


5. AJAX

AJAX é unha técnica de desenvolvemento web para crear aplicacións interactivas que se executan no **cliente**, é dicir, no navegador dos usuarios mentres se mantén a comunicación *asíncrona* col servidor nun segundo plano. Deste xeito é posible facer cambios sobre as páxinas sen necesidade de recargalas, o que supón unha maior interactividade, velocidade e facilidade de uso das aplicacións.

AJAX significa Asynchronous Javascript And XML.

A meirande parte das comunicacións de datos en AJAX entre un servidor e un cliente web adoitan realizarse en formato XML. E a maior parte das veces non se accede aos datos XML dun xeito directo, senón que estes son xerados de forma dinámica no servidor, normalmente mediante un script PHP. Por exemplo, a información podería obterse dunha base de datos que ten información que pode variar co paso do tempo.



A vantaxe principal que proporciona AJAX fronte ao xeito tradicional de obter e mostrar información é que con AJAX non se fai un cambio de páxina, senón que a información se visualiza na mesma páxina, o que permite seguir traballando coa mesma páxina como se non sucedera nada, actualizándose a información cando sexa necesario.

O uso de AJAX fará que o usuario interactúe dun xeito máis fluído co servidor web, o que dará as nosas páxinas un deseño máis profesional.

6. Linguaxes de marcas

6.1 ¿Que é o marcado?

En documentos electrónicos, o marcado son códigos, tamén chamados ***etiquetas*** que se introducen no texto electrónico para definir a estrutura e o formato no que teñen que aparecer.

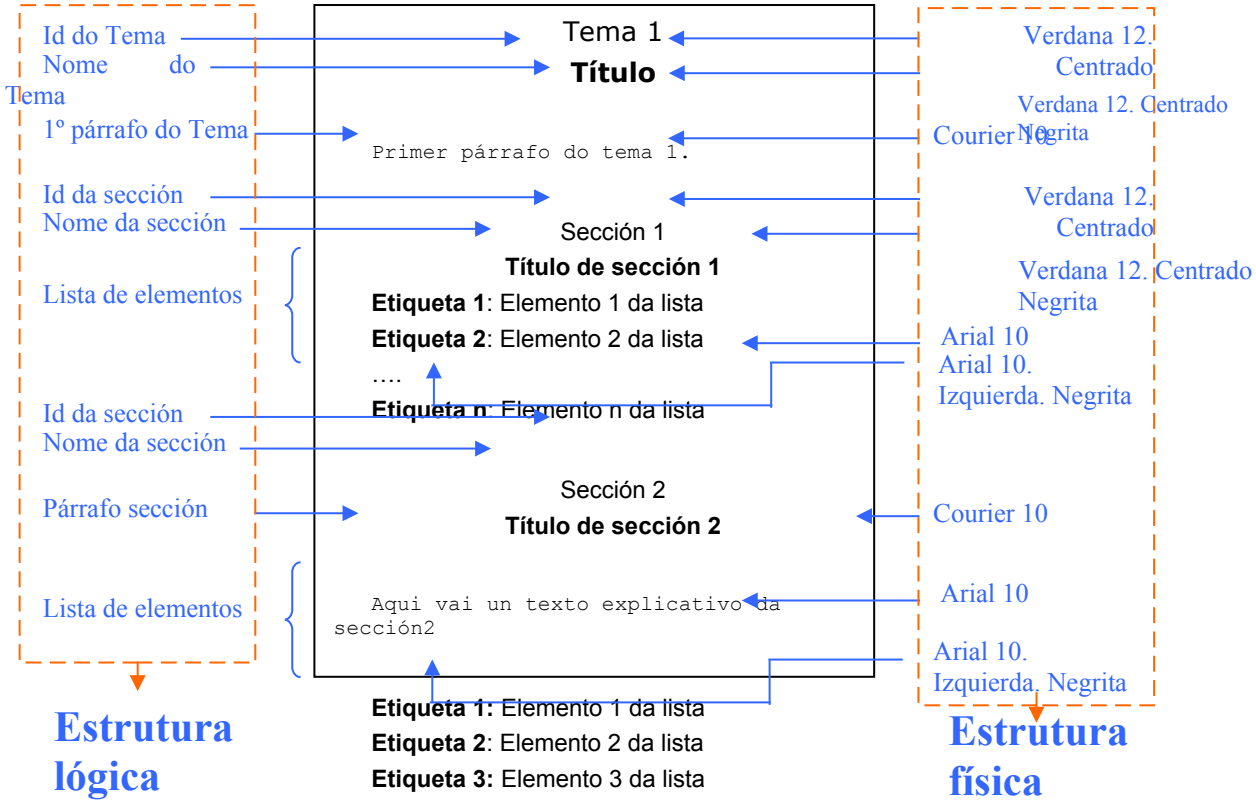
O marcado electrónico define dúas estruturas:

- Estrutura lóxica: formada polas partes que o compoñen e as súas relacións.

- Estrutura física: indica a aparencia do documento, incluíndo os seus compoñentes físicos, posicionamento de elementos e tipografía empregada.

  → isto sería a **Presentación** do documento

Un exemplo sería o seguinte:



6.2 Definición de linguaxe de marcas

Unha *Linguaxe de marcas* ou *Linguaxe de* marcado, pódese definir como unha forma de codificar un documento onde, xunto co texto, se incorporan etiquetas, marcas ou anotacións con información relativa á estrutura do texto, a súa presentación, ...

As linguaxes de marcas permiten facer explícita a estrutura dun documento, o seu contido semántico, ou calquera outra información lingüística ou extralingüística que se quere facer patente.

Exemplos:

```
<dataCelebración>O día<date>10/11/2010</date>tivo lugar ...</dataCelebración>

<subr>Santiago de Compostela</subr>
```

6.3 Clasificación das linguaxes de marcas

As linguaxes de marcado pódense clasificar en:

- **Procedemental:** Describen operacións tipográficas. → *Estrutura física*
- **Estrutural:** Describen a estrutura lóxica dun documento, pero non a súa tipografía
- **Híbrido:** Combinación dos dous anteriores.

Tamén se poderían clasificar polo seu uso. Algúns linguaxes de marcas específicos son:

- Documentación electrónica: RTF, TeX, Wikitexto
- Tecnoloxías de internet: HTML, XHTML, RDF, RSS
- Linguaxes especializados: MathXML, VoiceXML, MusicXML

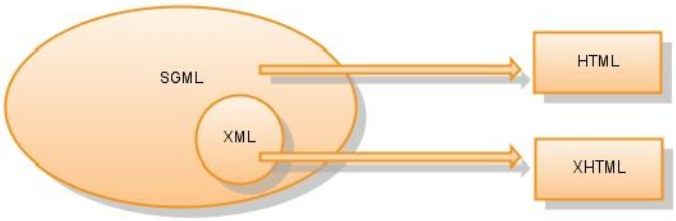
6.4 SGML: Onde empezou todo

IBM creou a linguaxe GML, que posteriormente en 1986, se estandarizaría pola Organización Internacional de Estándares (ISO) no SGML (*Standard Generalized Markup Language*). A linguaxe SGML sirve para especificar as regras de etiquetado de documentos e non impón en sí ningún conxunto de etiquetas en especial

A importancia do SGML é tal, que serviría de base para a creación de multitude de linguaxes específicas de marcas. O máis importante deles, con diferenza, creouse a principios dos anos 90, e foi a primeira gramática da linguaxe HTML (e ata a versión 4.01).

O problema do SGML é que, debido ós ambiciosos obxectivos suscitados á hora da súa creación, se converteu nunha linguaxe moi potente e complexa, e polo tanto moi difícil de aprender e de utilizar. Ademais tiña moitas características pensadas para optimizar a escritura manual dos documentos no ordenador (por exemplo, a posibilidade de non pechar ou omitir etiquetas), o cal complicaba a súa interpretación ás aplicacións.

Por iso se creou o XML (*eXtensible Markup Language*), que non é máis que un subconxunto do SGML creado para simplificar a creación das gramáticas de linguaxes de marcado.



Os deseñadores do XML deixaron fóra as partes menos utilizadas do SGML, conseguindo que a especificación do XML ocupe 30 páxinas, fronte ás 500 páxinas do SGML. Segundo Richard Ligth, no seu libro "*Presenting XML*", XML ofrece o 80% das vantaxes do SGML cun 20% da súa complexidade.

Igual que HTML se creou en base a SGML, XHTML (*eXtensible HyperText Markup Language*) creouse en base a XML.

6.5 Conceptos comúns as linguaxes de marcas

Antes de entrar a ver en detalle as linguaxes de marcado, convén ter claros algúns conceptos básicos para estas linguaxes.

Etiquetas

Inda que existe certa flexibilidade á hora de empregar as etiquetas cando se traballa con SGML, a sintaxe estándar utiliza parénteses angulares para indicar o comezo e fin dunha etiqueta. O nome aparecerá entre estes como se ve a continuación.

```
<etiqueta>
```

A anterior é unha etiqueta de comezo, que indica que a continuación irá o contido. Ao final do contido, haberá unha etiqueta de fin, que é similar á de comezo, excepto que o nome da etiqueta está precedido por unha barra inclinada:

```
</etiqueta>
```

Elementos

Un par de etiquetas de comezo e de fin xunto co contido que aparece entre elas é o que se coñece como *elemento.* O formato xeral dun elemento é o seguinte:

```
<etiqueta>        → Etiqueta de comezo
  Contido         → Elemento
</etiqueta>       → Etiqueta de fin
```

As características definidas pola etiqueta se aplican ao contido do elemento. Os elementos poden ter elementos aniñados. Un elemento aniñado é coñecido como elemento fillo e comeza e termina dentro dos elementos pai.

```
<etiqueta_pai>
      <etiqueta_filla>
       Contido                    } Elemento aniñado
      </etiqueta_filla>
</etiqueta_pai>
```

Atributos nas etiquetas

Algúns elementos teñen atributos. Estes atributos son os encargados de configurar o elemento dalgunha maneira. Os atributos aparecen dentro das etiquetas de inicio e consisten nun ou máis pares nome-valor co seguinte formato:

```
nome_atributo="valor_atributo"
```

Por exemplo, se temos un elemento chamado `documento`, podería ter un atributo chamado `linguaxe` cun valor que utilice o código dun idioma estándar, como pode ser o francés (fr):

```
<documento linguaxe="fr">
```

Poderían utilizarse as comillas simples para rodear os valores dos atributos en lugar das dobres.

Os atributos son os encargados de proporcionar os metadatos aos elementos. Noutras palabras, utilízanse para proporcionar información extra sobre un elemento ou para aplicar certa configuración adicional aos mesmos. A linguaxe en que se escribe un documento é información do mesmo, non parte do seu contido, así que especificar a linguaxe como un atributo dota de maior sentido ao elemento que o utiliza.

Un elemento debe ter máis dun atributo. Neste caso, deben aparecer todos dentro da etiqueta de apertura, non importando a orde na que aparecen.

```
<documento linguaxe="fr" tipo="manuscrito">
```

sería equivalente a:

```
<documento tipo="manuscrito" linguaxe="fr">
```

Sintaxe correctamente formada

As principais regras dunha sintaxe ben formada son as seguintes:

- Todas as etiquetas debe estar equilibradas de modo que un elemento teña tanto unha etiqueta de apertura como de peche:

```
<etiqueta></etiqueta>
```

- As etiquetas deben estar correctamente aniñadas de xeito que unha etiqueta debe pecharse antes de que se peche a etiqueta pai:

```
<etiqueta_pai>
      <etiqueta_filla>
        .....
      </etiqueta_filla>
</etiqueta_pai>
```

- Un documento debe ter un elemento ***raíz*** que rodee a todo o documento de modo que a súa etiqueta de comezo sexa a primeira etiqueta do documento e a súa etiqueta de fin sexa a última.
- Todos os valores dos atributos deben ser escritos entre comillas. Son válidas tanto as comillas dobres como as simples, pero deben emparellarse correctamente (un mesmo elemento non pode combinar comillas simples e dobres no mesmo elemento.

```
<documento tipo="manuscrito" linguaxe='fr'>
```