Material didáctico para exposición teórica

Familia profesional	IFC	Informática e comunicacións
Ciclo formativo	CSIFC03 CSIFC02 CSIFC01	Desenvolvemento de aplicacións web Desenvolvemento de aplicacións multiplataforma Administración de sistemas informáticos en rede
Grao		Superior
Módulo profesional	MP0373	Linguaxes de marcas e sistemas de xestión de información
Unidade didáctica	**UD02**	**XML**
Actividades	**A01** **A02**	**XML** **Espazos de nomes**
Profesoras		María del Carmen Fernández Lameiro María Elena Goy López

Índice

1. A01. XML

1.1 Introdución

XML (*eXtensible Markup Language*) deriva da linguaxe SGML e é unha linguaxe de marcas extensible. Utilízase para estruturar, almacenar e transmitir información. Que sexa extensible significa que é mais cunha linguaxe, é unha metalinguaxe ao poder ser utilizada para crear novas linguaxes.

XML non dispón dun grupo limitado de etiquetas, se non que se poden crear, ampliar e adaptar etiquetas a medida e crear con elas novas linguaxes, sempre que as etiquetas e atributos cumpran un número moi pequeno de normas. Exemplos de linguaxes XML:

Linguaxe	Utilización
3DML	para presentación de obxectos en 3D
Xforms	para definición de formularios
MathML	para definición de fórmulas e expresións matemáticas
SMIL	para creación de presentacións e contidos multimedia
VML e SVG	para definición de gráficos vectoriais

Unha avantaxe dos documentos XML é que son arquivos de texto e por tanto permiten mover información entre aplicacións en distintas plataformas, como por exemplo, mover información dende a base de datos Postgres nun sistema Linux á base de datos MS-SQL Server nun sistema Windows.

Sobre os documentos XML pódense facer operacións específicas que poñen en valor traballar con este tipo de documentos para transmitir información. Algunhas delas son:

- Crear documentos ben formados, é dicir, que cumpra a sintaxe básica XML.
- Validar documentos segundo as especificacións detalladas nun DTD ou nun esquema.
- Localizar elementos que compoñen o arquivo XML mediante XPath.
- Transformar a estrutura do arquivo XML para conseguir outro documento mediante XSLT. Por exemplo converter un documento XML noutro HTML, XHTML ou PDF.
- Establecer relacións entre os elementos mediante XLink e Xpointer.

1.2 Analizador XML

Un documento XML pode ser lido por persoas pero o seu obxectivo é que sexa procesado por aplicacións e procesos. Un analizador XML é un programa de baixo nivel que funciona entre unha aplicación e os arquivos XML. A súa función é obter información do arquivo XML, controlando a sintaxe; algúns tamén poden validar o arquivo XML contra un documento (DTD ou Schema).

A interface entre o analizador e a aplicación baséase na lectura que o analizador fai do documento XML e na interpretación que fai a aplicación desa lectura. Pode estar baseada en:

- Eventos

  Neste caso, o analizador xera eventos a medida que vai lendo o arquivo XML e encontrando etiquetas de apertura ou peche. A aplicación que utilice este tipo de analizadores terá que manexar eses eventos e reaccionar ante eles mediante controladores de eses

eventos. Este tipo de analizadores é utilizado por aplicacións que teñen a súa propia estrutura de datos como por exemplo, as aplicacións que importan documentos XML a bases de datos. O estándar para esta interface é SAX (Simple API for XML).

As vantaxes deste tipo de analizadores son que:

- Non necesitan gran cantidade de memoria xa que só cargan nela a sección de datos XML coa que estea traballando.
- É rápido e sinxelo escribir aplicacións baseadas neles.

As desvantaxes son que:

- Dependen totalmente da estrutura do documento, de tal xeito que se cambia algo no documento XML, hai que volver a reescribir a aplicación.
- Só se poden escribir accións simples xa que ao ir procesando o documento a medida que se lee non se ten información sobre o que vén a continuación.

- Obxectos. Neste caso, o analizador lee por completo o arquivo XML e xera en memoria unha estrutura de árbore cos elementos. As aplicacións que utilicen este analizador son do tipo exploradores e editores. O estándar para esta interface é DOM (Document Object Model).

  As vantaxes deste tipo de analizadores é que son moi rápidos xa que traballan con tódolos datos en memoria e non nun arquivo como nos analizadores tipo SAX e ao ter toda a estrutura en memoria pódense facer operación complexas movéndose por toda a estrutura de árbore (retroceder, avanzar, facer buscas....).

  As desvantaxes son que precisa máis memoria cós SAX e que para procesar todo o documento unha soa vez é máis lento; pola contra se hai que recorrer o documento máis dunha vez será moito máis rápido xa que a estrutura xa está cargada en memoria a partir da primeira lectura.

Algúns analizadores poden utilizar as dúas interfaces.

1.3 Ámbitos de aplicación de XML

As aplicacións que poden xerar, almacenar, transportar ou interpretar información en arquivos XML poden ser de case calquera ámbito, como por exemplo:

- Un navegador web como IE ou Mozilla.
- Un procesador de textos como OpenOffice Writer que utiliza XML para estruturar os documentos.
- Un IDE como NetBeans que utiliza XML para estruturar os proxectos.
- Clientes gráficos como MySQL Query Browser ou Workbench, ou clientes web como PHPMyAdmin, que permiten importar e/ou exportar datos dunha base de datos MySQL en XML.
- Programas para a xestión da formación de traballadores como no caso da Fundación Tripartita para la Formación en el Empleo, que utiliza documentos XML validados por un Schema para unir a información que subministra cada empresa participante.
- Predición do tempo por localidades en Aemet.
- Páxinas web escritas en XHTML.
- Programas escritos en Java, C ou PHP.
- Unha ferramenta como Speccy que proporciona información sobre o sistema dun PC.

- Un programa de debuxo como Adobe Illustrator, que interpreta coordenadas de dúas dimensións en XML.
- Unha folla de cálculo como Gnumeric, que analiza sintacticamente XML para buscar números e funcións usadas nun cálculo.
- Unha aplicación como Google Earth que utiliza unha gramática XML denominada KML para crear modelos e almacenamento de función xeográficas como por exemplo puntos, liñas, imaxes, ou polígonos e que permitirá compartir lugares e información sobre eles.
- Protocolos estándar para servizos web: WSDL (Web Services Description Language) e SOAP (Simple Object Access Protocol).

1.4 Versións

O Consorcio World Wide Web (W3C) é unha comunidade internacional formada por organizacións e persoal a tempo completo e público en xeral, que traballan conxuntamente para desenvolver estándares Web. A súa páxina oficial en español é : http://www.w3c.es/. W3C define as versións:

- *XML 1.0 Fifth Edition W3C Recommendation 26 November 2008* (http://www.w3.org/TR/REC-XML/) que modifica o orixinal publicado o 10 de febreiro de 1998
- *XML 1.1 Second Edition W3C Recommandation 16 August 2006* (http://www.w3.org/TR/XML11/) que modifica o orixinal publicado o 13 de decembro de 2001.

As diferenzas entre elas en canto ao contido dos documentos XML é que a versión 1.0 é moito máis restritiva nos carácteres permitidos para os nomes das etiquetas, dos atributos e dos elementos identificados de forma única.

1.5 Estrutura do documento XML

O documento XML está formado por unha serie de carácteres que teñen que cumprir unhas normas de sintaxe básica para estar ben formado, almacenados segundo un sistema de codificación. Consta de dúas partes: prólogo e corpo.

1.5.1 Sistema de codificación

Usaremos sempre UTF-8

O sistema de codificación empregado nun documento é o xogo de carácteres empregado para representar internamente os símbolos que contén. Pódense utilizar os seguintes sistemas de codificación:

- ASCII (*American Standard Code for Information Interchange*). Emprega os primeiros 7 bits dun byte (entre 0 e 127). Definía como se debían representar 128 carácteres entre números, letras (fala inglesa soamente), signos de puntuación e 32 carácteres de control. O oitavo bit usábase para controlar erros na transmisión (control de paridade).
- ASCII estendido que utiliza os 8 bits para representar carácteres, é dicir, 128 máis có ASCII. Existen moitas páxinas de código para representalos entre as que se destacan:
  - CP437. É o xogo de carácteres orixinal do IBM PC (o que viña na ROM dos adaptadores de vídeo). Inclúe vocais minúsculas con til, eñes e apertura de exclamacións e interrogacións. Non inclúe, por exemplo, vocais maiúsculas con til, salvo a É.

  - CP850. É unha adaptación do CP437 que se empregou en MS-DOS en países de Europa occidental. Inclúe, entre outros, todas as vocais maiúsculas con til. Unha pequena modificación desta páxina de código incluíndo o símbolo do euro é a CP858.
  - Windows Code Pages. Son xogos de carácteres empregados polos sistemas operativos Windows na década dos 80. Existen varios, cada un pensado para ser empregado nun alfabeto determinado. En Europa occidental e países de lingua inglesa empregouse o xogo de carácteres Windows-1252.
  - ISO-8859 moi empregados en sistemas Linux. O máis usado para o español é o ISO-8859-1 ou latin-1 e o ISO-8859-15 que é unha pequena variante que inclúe o símbolo do euro (€). Segue a usarse amplamente hoxe en día.
- Unicode que naceu para evitar a complicación anterior, consiste nun repertorio de máis de 100.000 carácteres que abranguen a maior parte de alfabetos empregados na actualidade. Existen diversas formas de especificar a codificación de cada carácter. As máis utilizadas son:
  - UTF-8. Codificación orientada a byte con símbolos representados con lonxitude variable de un a catro bytes. Os 128 carácteres do ASCII orixinal codifícanse nun byte que sempre comeza por cero. Para representar outro carácter de ampla utilización (vocais con til, eñe, alfabeto grego, árabe, etc.), empréganse 2 bytes. Os demais carácteres usarán 3 ou 4 bytes. É moi utilizado pois posibilita a codificación de calquera carácter Unicode reducindo considerablemente o espazo ocupado polos documentos se o comparamos coas súas alternativas.
  - UTF-16. Emprega tamén codificación de lonxitude variable. Neste caso, divide os carácteres en dous grupos: os de máis ampla utilización, que codifica en 2 bytes, e o resto, que ocupan 4 bytes cada un.
  - UTF-32. é una codificación de lonxitude fixa que utiliza 4 bytes por carácter polo que en xeral, os documentos con este sistema de codificación ocuparán máis espazo que se utilizaran UTF-8.

Precaucións a ter en conta cos documentos XML con relación aos sistema de codificación:

- É moi importante especificar no documento o xogo de carácteres que se utilizou ao gravalo. De no ser así, a aplicación que o vaia a utilizar traducirá o código segundo o sistemas de codificación que teña por defecto e o resultado pode ser ilexible. Os navegadores web, por exemplo, permiten consultar e cambiar o xogo de carácteres utilizado para representar cada documento.
- Débese gardar o documento XML co mesmo sistema de codificación que o indicado no documento.
- Aínda cando se estea representando un documento no mesmo xogo de carácteres no que foi creado, isto non garante que a representación en pantalla sexa correcta. Necesítase ademais que o tipo de letra que esteamos usando conteña tódolos carácteres que utiliza o documento. A gran maioría de tipos de letra conteñen soamente os carácteres máis habituais que utilizamos ao escribir documentos.

1.5.2 Prólogo

O prólogo é a parte na que se declara o documento XML, na que tamén se pode declarar o DTD utilizado para validar, e na que se colocan as posibles instrucións de procesamento.

1.5.2.1 Declaración XML

Sintaxe:

```
<?xml version="versionXML" encoding="codificacion" standalone="yes|no"?>
```

- *version*: a versión do XML utilizada, habitualmente (e o seu valor por defecto) 1.0.
- *encoding*: refírese ao sistema de codificación utilizado. Aconséllase indicar sempre o sistema de codificación aínda que non sexa obrigatorio.
- *standalone="yes | no"* : informa de se o documento é independente de declaracións externas como por exemplo un DTD. O seu valor por defecto é "no".

Exemplo:

```
<?xml version="1.0" encoding="UTF-8"?>
```

Isto será o prólogo mínimo e obrigatorio que empregaremos

Na versión 1.1 é obrigatorio indicar polo menos a versión XML. Por defecto usarase a versión 1.0.

1.5.2.2 Tipo de documento

Verémolo cando estudemos os DTD's

Declara as normas de sintaxe para validar o documento XML mediante un DTD interno ou externo. Este tema verase con máis detalle máis adiante. Sintaxe para DTD interno:

```
<!DOCTYPE elementoraiz[
...definición do DTD...
]>
```

- *elementoraiz* é o nome do elemento raíz.
- *definición do DTD* está composto polas normas de sintaxe.

Exemplo:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE alumnos[
<!ELEMENT alumnos (alumno)*>
<!ELEMENT alumno (#PCDATA)>
]>
<alumnos>
        <alumno>Pepe</alumno>
        <alumno>María</alumno>
</alumnos>
```

Sintaxe para DTD externo:

```
<!DOCTYPE elementoraiz SYSTEM "urldtd"
```

- *elementoraiz* é o nome do elemento raíz
- *urldtd* é a url do arquivo dtd externo

Exemplo:

```
<!DOCTYPE alumnos SYSTEM 'alumnosExterno.dtd'>
```

1.5.2.3 Instrucións de procesamento

No prólogo e en xeral en calquera lugar do documento XML, poden aparecer instrucións de procesamento que pasan información ás aplicacións que van ler o documento.

Sintaxe:

```
<?aplicación instrucións?>
```

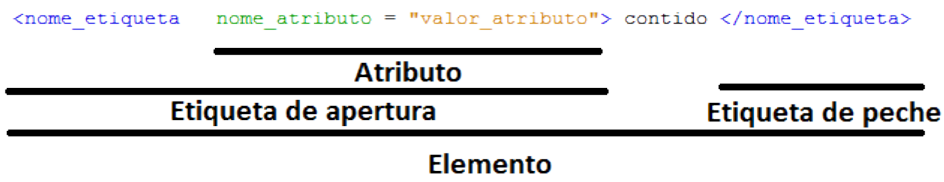
- *aplicación* é o nome da aplicación á que van dirixidas as instrucións. Se empeza por xml serán instrucións de procesamento de XML.
- *instrucións* son as ordes para esa aplicación e estarán no seu formato.

Exemplo:

```
<?xml-stylesheet type="text/css" href="principal.css"?>
```

1.5.3 Corpo

O corpo está composto por elementos e os elementos están formados por etiquetas. As etiquetas son texto entre os símbolos < e > e poden ter atributos e/ou contido. Os atributos están pensados para gardar información adicional ou descrición acerca das etiquetas, especifícanse na etiqueta de inicio e teñen que ter un único valor entre comiñas simples ou dobres. As etiquetas poden almacenar varios datos e incluso estenderse mediante estruturas aniñadas de varios elementos. O contido pode non existir ou estar formado por elementos aniñados.



Exemplo de etiquetas:

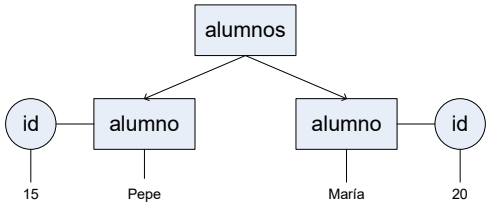
```
<root>
        <titulo>O meu primeiro libro de pesca</titulo>
        <cantidade unidade="kg">7</cantidade>
        <asignatura numHoras="12">Programación en Java</asignatura>
        <becario></becario>
</root>
```

1.5.4 Documento ben formado

Un documento XML está ben formado se cumpre as seguintes normas sintácticas:

- Todos os elementos do documento aníñanse en forma de árbore, existindo un único elemento raíz, do que descenden todos os demais.

```
<?xml version="1.0" encoding="UTF-8"?>
<alumnos>
    <alumno id="15">Pepe</alumno>
    <alumno id="20">María</alumno>
</alumnos>
```



- Toda etiqueta de apertura terá a correspondente etiqueta de peche (símbolo / antes do nome da etiqueta), agás se non ten contido que poderá indicarse como <nome_etiqueta/ >.

```
<becario></becario>
<beca>1200</beca>
<disfruta_beca/>
```

- Os elementos poden aniñarse, pero as etiquetas pecharanse de forma estruturada, é dicir, en orden inversa a súa apertura.

```
<observacions>O solar está <importante>pendente de revisión
</importante> no Concello</observacions>
```

- Un atributo, se existe, sempre ten que ter un valor entre comiñas simples ou dobres e non pode aparecer máis dunha vez nunha etiqueta, aínda que se poden especificar en calquera orde.

```
<modulo horas="125" idioma="galego">LMSXI</modulo>
<modulo idioma="catalán" horas="200" >CODE</modulo>
<modulo>BADA</modulo>
```

- Os documentos XML poden conter comentarios. Estes aparecerán sempre entre as marcas de inicio e peche de comentario, <!-- e --> respectivamente. Dentro dun comentario pódese engadir calquera carácter excepto dous guións consecutivos (--). Non pode haber comentarios nas etiquetas.

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
    Document:       proba.xml
    Created on:     7 de xaneiro de 2016, 11:00
    Author:         profesora
    Description:    Probas para introdución á XML.
-->
<root>
    <modulo horas="125" idioma="galego">LMSXI</modulo>
    <modulo idioma="catalán" horas="200" >CODE</modulo>
    <modulo>BADA</modulo>
</root>
```

- Nome de etiquetas e atributos:
  - Comezará sempre por unha letra (pode levar til e pódese utilizar ñ), un guión baixo (_) ou dous puntos (:) seguido de calquera combinación dos carácteres anteriores, números, guión (-) ou punto(.). Polo tanto, non se poden utilizar espazos en branco.
  - Non pode comezar por XML nin variantes de maiúsculas/minúsculas (xml, Xml, ...).
  - Distínguense entre maiúsculas e minúsculas (son *case-sensitive*).
  - O contido dos elementos e o valor dos atributos pode ter calquera carácter sempre que non interfira na sintaxe propia de XML. Por exemplo, os carácteres & e < non poden aparecer como parte do contido, salvo nos comentarios, instrucións de procesamento e seccións CDATA. O mesmo sucede coas comiñas (simples ou dobres): non poden formar parte do valor dun atributo delimitado mediante o mesmo tipo de comiñas. Para solucionar isto, utilizaranse referencias, entidades ou seccións CDATA.

1.5.5 Referencias, entidades e seccións CDATA

Os carácteres especiais como &, < , dobres comiñas (“) e apóstrofe ('), poden forma parte do contido dun elemento ou como valor dun atributo, utilizando unha das seguintes formas:

- Empregando unha referencia a un carácter Unicode. As referencias sempre empezan por &# (ou por &#x se empregamos hexadecimal) e rematan en punto e coma (;). Por exemplo, os carácteres & e < poderían substituírse respectivamente polas referencias & e < (en hexadecimal serían & e <).
- Empregando entidades. As entidades empezan por &, van seguidas do nome da entidade e terminan en punto e coma (;). Poden crearse novas entidades como se verá máis adiante e existen entidades predefinidas como por exemplo:

Carácter	Entidade
&	&amp;
<	&lt;

>	&gt;
"	&quot;
'	&apos;

O uso de calquera das dúas formas anteriores pode resultar pesado se hai que escribilo moitas veces, e ademais dificulta moito a lectura ás persoas, polo que a solución pode ser colocar o texto dentro dunha sección CDATA. Sintaxe:

```
<![CDATA[
texto
]]>
```

- *texto* non pode levar ningún símbolo que interfira na sintaxe.

Por exemplo, para poder ver nun navegador un elemento co contido seguinte:

```
<códigoFonte>
 O operador de concatenación é &. Os operadores de comparación son < e >
</códigoFonte>
```

Poderíase escribir:

```
<códigoFonte>
        <![CDATA[
        O operador de concatenación é &.
        Os operadores de comparación son < e >
        ]]>
</códigoFonte>
```

ou

```
<códigoFonte>
        O operador de concatenación é &amp;.
        Os operadores de comparación son &lt; e &gt;
</códigoFonte>
```

1.5.6 Atributos especiais

Non é común usalos

Aínda que cada documento XML ten os seus propios elementos, atributos e estrutura axeitada á súa finalidade, existen dous atributos especiais cun significado xa definido que ven recollido na especificación de XML.

1.5.6.1 xml:lang

Cando se quere indicar o idioma no que está descrito certo contido, emprégase o atributo xml:lang. O seu valor é o código de dúas letras que identifica o idioma segundo está definido na RFC 3066 (es, en, fr, etc.). A linguaxe especificada polo atributo aplícase ao elemento no que se insire, incluíndo o valor dos outros atributos se os houbese, e a todos os seus fillos sempre que non teñan especificada outra linguaxe mediante o seu propio atributo xml:lang.

```
<descrición xml:lang="es">Color rojo</descrición>
```

1.5.6.2 xml:space

XML define como "espazos en branco" o espazo en branco (32), tabulador (9), salto de liña (10) e retorno de carro (13), que permiten facer máis lexible o código permitindo ademais mostrar claramente a estrutura aniñada. Sen embargo, en ocasións é necesario indicar ás aplicacións que procesan os contidos, que certos espazos deben ser preservados. O atributo xml:space (default | preserve) pode empregarse co valor *preserve* naqueles elementos nos que é importante manter os espazos tal e como están definidos orixinalmente.

2. A02. Espazos de nomes

Non está mal botarlle un ollo, pero non os imos usar.

Os espazos de nomes ou *namespaces* son unha recomendación W3C (*World Wide Web Consortium*) para que os nomes comúns de etiquetas ou atributos non choquen, ben porque proveñen de distintos documentos XML e se vaian a utilizar no mesmo documento destino, ben porque nos interesa separar dun xeito estruturado as etiquetas ou atributos dun documento XML.

Por exemplo, se nun documento hai enderezos de clientes e provedores marcados coas etiquetas nome, enderezo e teléfono e se desexa poder diferencialas, pódense crear dous espazos de nomes: un para os clientes e outro para os provedores. A combinación entre o espazo de nomes e o nome da etiqueta identificará se o nome, enderezo ou teléfono é dos clientes ou dos provedores. Esta solución é máis necesaria se os clientes e os provedores están en dous documentos diferentes con orixes distintas e preténdese elixir información deles para crear un novo documento.

A declaración dun espazo de nomes realízase a través dun atributo *xmlns* ou *xmlns:prefixo*. A súa sintaxe é:

```
<nome_etiqueta xmlns:prefixo="espazo_nomes">
```

- *nome_etiqueta* é o nome da etiqueta na que se declara o espazo de nomes. O espazo terá efecto nese elemento e no seu contido.
- *espazo_nomes* é un URI identificador global único (*Uniform Resource Identifier*), normalmente un tipo especial de URI, o URL (*Uniform Resource Locator)*, pero tratado como cadea, isto é, non se comproba se é un localizador real. Non se recomenda o emprego de identificadores (URIs) relativos. Dous identificadores fan referencia ao mesmo espazo de nomes se son idénticos (tratados como cadea de texto), diferenciando incluso maiúsculas de minúsculas. Este valor é o que identifica realmente ao espazo de nomes.

  Se o espazo de nomes se deixa en branco, indica que o elemento non pertence a ningún espazo de nomes.
- *prefixo* é unha forma curta de facer referencia ao URI e é o nome que se colocará antes de cada elemento seguido de dous puntos para cualificar a ese elemento como pertencente a ese espazo de nomes. Se non existe prefixo, enténdese que é o espazo de nomes por defecto, e aplicarase ás etiquetas e atributos que non estean especificamente asociados a outro espazo de nomes.

Exemplo de espazo de nomes por defecto ao que pertencen tódalas etiquetas e atributos:

```
<?xml version="1.0" encoding="UTF-8"?>
<cursos xmlns="http://www.xunta.es/iesmurallaromana/di">
    <curso nome="LMSXI">
        <alumno>Pepe</alumno>
        <alumno>María</alumno>
    </curso>
    <curso nome="CODE">
        <alumno>Antonia</alumno>
        <alumno>Jesús</alumno>
    </curso>
</cursos>
```

Exemplo de espazo de nomes por defecto ao que pertencen a etiqueta *curso* do elemento de nome LMSXI, os seus atributos e as etiquetas *alumno* de Pepe e María:

```
<?xml version="1.0" encoding="UTF-8"?>
<cursos>
    <curso nome="LMSXI" xmlns="http://www.xunta.es/iesmurallaromana/di">
```

```
        <alumno>Pepe</alumno>
        <alumno>María</alumno>
    </curso>
    <curso nome="CODE">
        <alumno>Antonia</alumno>
        <alumno>Jesús</alumno>
    </curso>
</cursos>
```

Exemplo de espazo de nomes con prefixo ao que pertencen tódalas etiquetas e atributos:

```
<di:cursos xmlns:di="http://www.xunta.es/iesmurallaromana/di">
    <di:curso nome="LMSXI">
        <di:alumno>Pepe</di:alumno>
        <di:alumno>María</di:alumno>
    </di:curso>
    <di:curso nome="CODE">
        <di:alumno>Antonia</di:alumno>
        <di:alumno>Jesús</di:alumno>
    </di:curso>
</di:cursos>
```

Exemplo de espazo de nomes con prefixo ao que pertencen só as etiquetas e atributos vinculados a ese espazo mediante o prefixo:

```
<?xml version="1.0" encoding="UTF-8"?>
<cursos
    xmlns:di="http://www.xunta.es/iesmurallaromana/di">
    <curso nome="LMSXI">
        <alumno>Pepe</alumno>
        <alumno>María</alumno>
    </curso>
    <di:curso nome="CODE">
        <alumno>Antonia</alumno>
        <alumno>Jesús</alumno>
    </di:curso>
</cursos>
```

Exemplo de documento con dous espazos de nomes:

```
<?xml version="1.0" encoding="UTF-8"?>
<di:cursos
    xmlns:di="http://www.xunta.es/iesmurallaromana/di/cursos"
    xmlns:al="http://www.xunta.es/iesmurallaromana/di/alumnos">
    <di:curso nome="LMSXI">
        <al:alumno>Pepe</al:alumno>
        <al:alumno>María</al:alumno>
    </di:curso>
    <di:curso nome="CODE">
        <al:alumno>Antonia</al:alumno>
        <al:alumno>Jesús</al:alumno>
    </di:curso>
</di:cursos>
```

Exemplos iguais dende o punto de vista do analizador XML.

Primeiro exemplo:

```
<?xml version="1.0" encoding="UTF-8"?>
<di:cursos
    xmlns:di="http://www.xunta.es/iesmurallaromana/di">
    <di:curso nome="LMSXI">
        <di:alumno>Pepe</di:alumno>
        <di:alumno>María</di:alumno>
    </di:curso>
    <di:curso nome="CODE">
        <di:alumno>Antonia</di:alumno>
        <di:alumno>Jesús</di:alumno>
    </di:curso>
```

```
</di:cursos>
```

Segundo exemplo:

```
<?xml version="1.0" encoding="UTF-8"?>
<cursos
    xmlns="http://www.xunta.es/iesmurallaromana/di">
    <curso nome="LMSXI">
        <alumno>Pepe</alumno>
        <alumno>María</alumno>
    </curso>
    <curso nome="CODE">
        <alumno>Antonia</alumno>
        <alumno>Jesús</alumno>
    </curso>
</cursos>
```

Existen espazos de nomes de uso xeneralizado como o que se utiliza na etiqueta html dos documentos xhtml:

```
<html xmlns="http://www.w3.org/1999/xhtml">
```

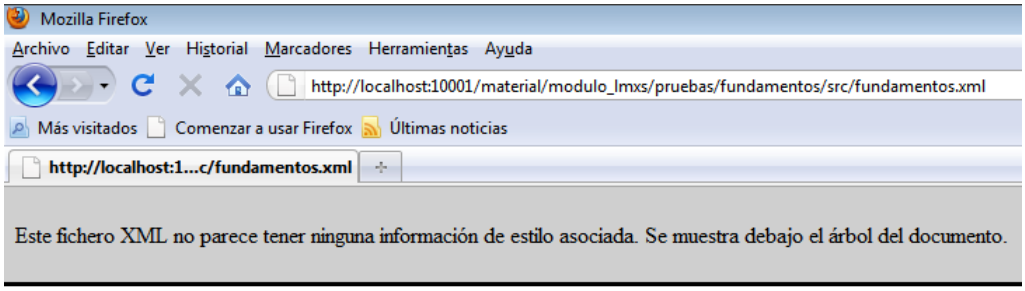
3. Ferramentas

Para editar documentos XML, pódese utilizar calquera procesador ou editor de textos que permita crear e modificar arquivos de texto plano, pero é máis práctico utilizar contornos de desenvolvemento que ademais de permitir editar documentos XML, axuden ó programador na escritura de XML. Recoméndase utilizar o IDE NetBeans xa que este contorno suxire e orienta sobre a colocación das etiquetas de peche, e ademais permite realizar outras operación relacionadas con documentos XML como por exemplo:

Podemos seguir usando o VS Code. Recoméndase instalar a extensión XML (autor Red Hat) e/ou XML Tools (autor Josh Johnson)

- Comprobar que un documento está ben formado.
- Escribir documentos DTD ou esquemas.
- Validar código XML cun DTD ou un Schema.
- Escribir transformación con XSLT e realizar a transformación.
- Localizar elementos mediante XPath.

Os clientes web ou navegadores, coma por exemplo Mozilla ou IE, mostran a estrutura de árbore dos elementos dos documentos XML se están ben formados ou mostran mensaxes de erro en caso contrario, pero normalmente non dan información sobre a validez dos documentos. Cando o navegador recibe a orde de cargar un documento XML, invoca ó analizador XML para que o revise e mostra o resultado que pode ser o propio documento ou un erro se é que está mal formado. Por exemplo en Mozilla, se o documento está ben formado, aparece a árbore do documento, resaltando as etiquetas do contido e permitindo expandir ou contraer elementos ao facer clic riba do signo máis (+) ou menos (-) respectivamente.




```
−<alumnos>
  −<alumno>
     <nome>Pepe</nome>
     <apelidos>Ruiz Arias</apelidos>
   </alumno>
  +<alumno></alumno>
  +<alumno></alumno>
  +<alumno></alumno>
 </alumnos>
```

Se o documento está mal formado, como o seguinte:

```
<?xml version="1.0" encoding="UTF-8"?>
<root>
    <modulo horas="125" idioma="galego">LMSXI</modulo>
    <modulos idioma="catalán" horas="200" >CODE</modulo>
    <modulo>BADA</modulo>
</root>
```

Mozilla mostrará a mensaxe de erro parecida á seguinte, advertindo da liña e columna onde o analizador encontrou o fallo sintáctico:

Error de lectura XML: etiqueta sin pareja. Se esperaba: </modulos>.
Ubicación: http://localhost:10001/probas/newXMLDocument.xml
Número de línea 4, columna 49:

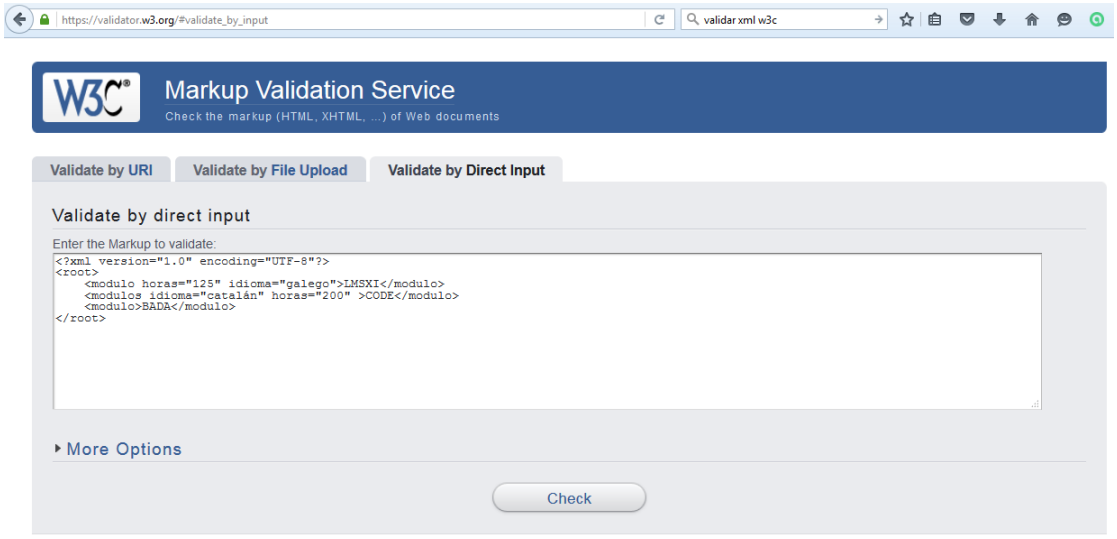
```
    <modulos idioma="catalán" horas="200">CODE</modulo>
----------------------------------------------^
```

Outra ferramenta útil para é o servizo de validación de W3C https://validator.w3.org/ que permite comprobar se un documento XML está ben formado elixindo unha das seguintes opcións:

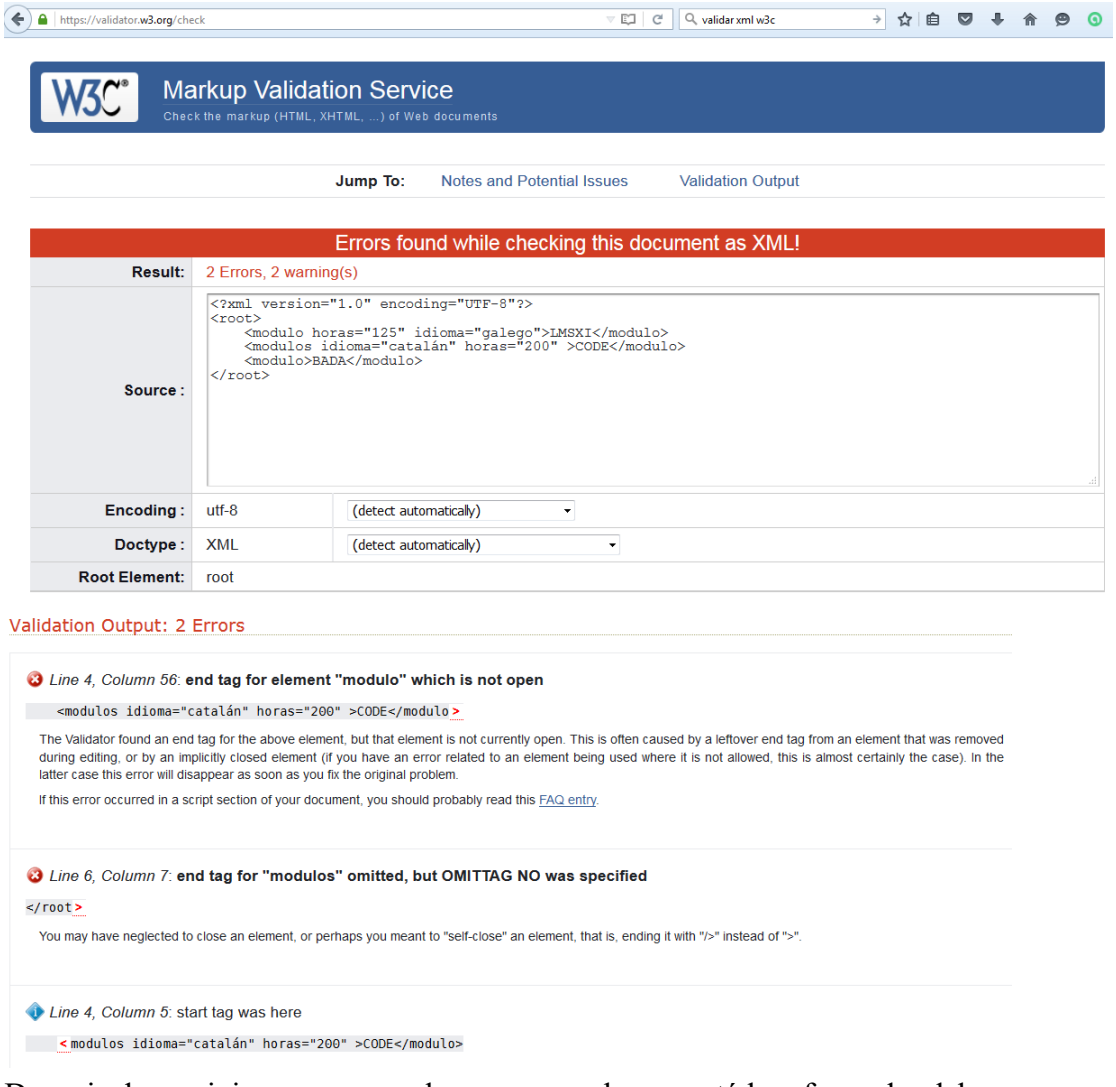
Este validador xa non funciona con XML, hai moitos se buscades en internet. Un recomendable é: https://onlinexmltools.com/validate-xml

- Tecleando o URI (*Validate by URI*).
- Subindo o arquivo ó servidor (*Validate by File Upload*).
- Tecleando directamente o código (*Validate by direct input*).

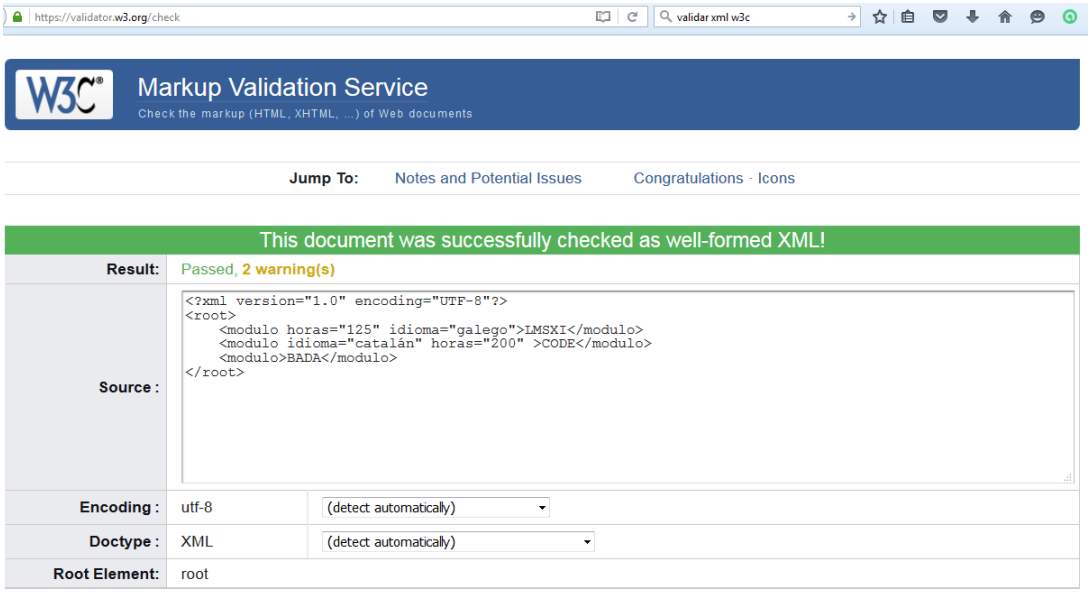
Pinchando en *More Options,* pódese seleccionar o tipo de documento que se quere validar ou deixar a opción por defecto que é que detecte automaticamente o tipo de documento.



Para validar, débese premer no botón *Check*. O servizo de validación encontrou 2 erros graves e 2 leves que están máis explicados abaixo na sección *Validation Output*:



Despois de corrixir os erros e volver a comprobar se está ben formado, debera aparecer que o documento está ben formado:



4. Documentos de apoio ou referencia

- PIN RODRIGUEZ, Margarita, LOURIDO ESTÉVEZ, Víctor M. *Unidade didáctica 3(XML), actividade 1(A linguaxe XML) do módulo Linguaxes de marcas e sistemas de xestión da información.* Xunta de Galicia, Consellería de Cultura, Educación e Ordenación Universitaria. 2013.
- S. ZURDO, Javier, TOHARIA RABASCO, Pablo, RAYA GONZÁLEZ, Laura. *Lenguajes de Marcas y Sistemas de Gestión de la Información.* Editoria Ra-Ma.
- *Wikipedia.* http://es.wikipedia.org/wiki/Wikipedia:Portada
- MOLINARI, Lia. *Arquitecturas orientadas a Web Services*. 2004. http://es.scribd.com/doc/45692649/58/Los-analizadores-XML