

DESARROLLO DE INTERFACES

UD2. GENERACIÓN DE INTERFACES A PARTIR DE DOCUMENTOS XML

2CFGS

DAM

GENERACIÓN DE INTERFACES A PARTIR DE DOCUMENTOS XML



Pau Gallego Garcia

CFGS DISEÑO APLICACIONES MULTIPLATAFORMA

Módulo: 0488 – Desarrollo de Interfaces

UD2. Generación de interfaces a partir de documentos XML.

INTRODUCCIÓN

Para diseñar y desarrollar una interfaz gráfica de usuario se pueden elegir entre muchas opciones de entornos de desarrollo y entre muchos lenguajes de programación. Si se elige un lenguaje de cuarta generación y un entorno integrado de desarrollo, se podrá disponer de muchas más herramientas y elementos que facilitarán este trabajo.

La primera decisión que debe establecerse en cuanto al diseño de una aplicación que tendrá varias interfaces de usuario es el entorno de ejecución, si será local o en un entorno cliente-servidor, si será para WinForms o para un entorno web, si RichClient o ThinClient... Esta decisión influirá en el lenguaje de programación de la aplicación y de las interfaces, para decidir, a posteriori, qué entorno de desarrollo habrá que escoger.

1. INTRODUCCIÓN A XML

¿Qué es XML? Se verá con más detalle en los siguientes apartados, pero básicamente es un lenguaje con una **función principal: describir datos por medio de etiquetas**, sin preocuparnos cómo se tratarán estos datos ni cómo se mostrarán. ¿Qué tiene de positivo? Este lenguaje puede ser desarrollado por los programadores o diseñadores de software desde otros muchos lenguajes como Java, C#, .NET, etc.

W3C

W3C és el World Wide Web Consortium. Es tracta d'una comunitat internacional que desenvolupa estàndards, criteris i recomanacions referents a les tecnologies web que assegurin, a llarg termini, el creixement d'Internet.

¿Por qué hablamos de XML en un tema que trata sobre la generación de interfaces gráficas de usuario? Un archivo en XML no permite diseñar ni configurar una interfaz gráfica, sólo ofrecerá los datos necesarios para ser mostrados o manipulados.

A partir de otros lenguajes de programación que utilizan las características de XML se pueden desarrollar interfaces de forma muy sencilla, para aplicaciones especialmente pensadas para un entorno Web y cumpliendo los estándares, criterios y recomendaciones definidas por el W3C.

1.1 ¿Qué es el XML?

XML (**eXtensible Markup Language**) tiene como **función principal la de describir datos por medio de etiquetas y no se preocupa de la forma de tratarlos o mostrarlos.**

XML es un **metalenguaje extensible de etiquetas**, es decir, que **a partir de la definición de datos por medio de etiquetas y ciertas reglas sirve como base para definir otros lenguajes de marcas.** Algunos ejemplos de lenguajes definidos sobre XML son:

- **XHTML**

- **RSS** (Really Simple Syndication)
- **SVG** (Scalable Vector Graphics)
- **MathML**
- **GraphML**
- **MusicXML**

Las etiquetas son un conjunto de caracteres alfanuméricos que quedarán escritos entre los símbolos > y <. Este conjunto de caracteres, el sustantivo que representan, será el identificador de la etiqueta. Por ejemplo, <nombre> es una etiqueta con el identificador "nombre"

Cada vez más aplicaciones son capaces de definir su propio lenguaje específico basado en el uso de etiquetas XML. Este hecho hace que el lenguaje XML sea cada vez más utilizado y más popular entre los desarrolladores de aplicaciones y de interfaces. Todo esto es gracias a su enfoque genérico, extensible y muy sencillo, basado en muy pocos elementos básicos y unas reglas de formato muy estrictas.

A partir de archivos XML, y algunas herramientas o lenguajes más, se podrán desarrollar todo tipo de archivos y formatos, como los representantes en la siguiente figura:



Para entender mejor el funcionamiento del XML se muestra a continuación un ejemplo de sus características y de su código:

La X de XML significa *eXtensible*, es decir, ampliable. Ser ampliable significa que puede definir cualquier conjunto de etiquetas adicionales sin que bloquee la aplicación. Por ejemplo, a continuación, tenemos una parte de código implementado en XML. Para una asignatura, indicaremos el nombre y las horas que durará la impartición.

```
<assignatura>
  <nom> Programació </nom>
  <hores> 150 </hores>
</assignatura>
```

En este código, las etiquetas son: asignatura, nombre y horas.

Ahora se querrá hacer más extenso este ejemplo con más informaciones, esto es, ampliarlo. Además de los datos descritos, se quiere añadir el nombre del profesor, pero también añadir

un nivel más a estos datos (un hijo más), que son los nombres de los alumnos. Para conseguir esto deberemos modificar el archivo XML para que tenga este nuevo aspecto:

```
<assignatura>
  <nom> Programació </nom>
  <professor> Joan Castells </professor>
  <alumnes>
    <alumne> Joan Garcia </alumne>
    <alumne> Carol Perez </alumne>
    <alumne> Pep Perez </alumne>
  </alumnes>
  <hores> 150 </hores>
</assignatura>
```

La aplicación que lea este documento XML podrá saber además qué alumnos cursan la asignatura.

Un archivo XML es un archivo que contiene a su vez los **datos** y su **estructura** en el **mismo documento**. Ésta es una característica muy importante de los archivos XML. El hecho de que sean **autodescriptivos** simplifica mucho el trabajo con este tipo de archivos, ya que son muy **sencillos de entender y manipular**. Este hecho es muy importante tanto para el desarrollo de programas que los utilicen, como para utilizarlos con configuraciones o envío de datos entre diferentes entornos o plataformas.

1.2 ÁMBITO DE APLICACIÓN

Los documentos XML son archivos de texto normal que se pueden abrir desde cualquier editor de texto, por muy sencillo que sea. No son como otros tipos de archivos que necesitan un software de propiedad específico para interpretarlos, como ocurre con la mayoría de los archivos binarios. Esto significa que se puede utilizar un editor tan sencillo como el Vio en Linux o el Bloc de notas de Windows para abrir y editar un archivo XML.

Una de las consecuencias más importantes de este hecho es la portabilidad que ofrece XML. El formato de archivos tipo texto y el hecho de ser autodescriptivos hacen que se puedan manipular y entender desde cualquier plataforma.

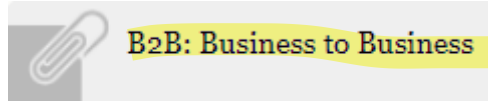
Hay que comentar también otros ámbitos de aplicación del XML:

- Cuando se necesita intercambiar información a través de varias plataformas (compatibles o incompatibles) de hardware o software, o de diversas aplicaciones, XML puede ser la elección más acertada. Éste es el caso de aplicaciones de XML en el ámbito de tecnologías de la comunicación como, por ejemplo, el popular WML (lenguaje de formato inalámbrico) y WAP (protocolo aplicaciones inalámbricas).

WML

WML és un llenguatge basat en XML que s'utilitza per donar format a aplicacions d'Internet per a dispositius de mà, com telèfons o PDA.

- En aplicaciones de comercio electrónico entre empresas (B2B), en las que las empresas necesitan intercambiar una gran cantidad de información financiera de forma independiente de la plataforma, el XML también será una buena elección. Varias aplicaciones utilizan SOAP (Protocolo sencillo de acceso a objetos), un protocolo muy popular basado en XML, para intercambiar este tipo de información a través de Internet. Estas aplicaciones basadas en XML que se utilizan para compartir información se llaman servicios web.

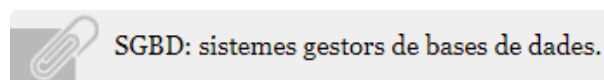


En el caso de desarrolladores que utilizan lenguajes como HTML y CSS para diseñar las interfaces y la presentación de los datos, utilizando XML para transferirlos se reducirá el riesgo de incluir contenido redundante. Si hay cambios con respecto a las interfaces o en la configuración de la presentación, esto no afectará a los datos, que se almacenan por separado en un archivo XML.

Como ejemplo para este último caso, se puede plantear un sistema de gestión de contenidos. Este sistema puede generar documentos para usuarios finales en varios formatos, como HTML, PDF, etc. Sin embargo, no tiene sentido almacenar múltiples versiones (una en cada formato) de un mismo documento con los mismos datos. El contenido se duplicaría y ocuparía un valioso espacio en el disco, lo que llenaría el CMS de información redundante y haría más lento su uso. Si se utiliza un motor basado en XML, el contenido se puede almacenar una sola vez para extraerlo y mostrarlo posteriormente en el formato solicitado. Si XML separa el contenido de la presentación, ¿qué deberá utilizarse para presentar mis datos? Existen otros lenguajes basados en los documentos XML, que se trabajarán en los siguientes apartados.

En resumen, algunos de los ámbitos de aplicación de XML podrían ser:

- En aplicaciones que manejan gran cantidad de datos y, al mismo tiempo, necesitan ser flexibles y ampliables. Por ejemplo, sitios web, listas de ofertas de empleo o aplicaciones financieras.
- En aplicaciones en las que la presencia de contenidos redundantes sea un peligro, como en sistemas de administración de contenido, bibliotecas de documentos o sistemas de seguimiento de sitios web.
- En aplicaciones en las que es necesario intercambiar gran cantidad de datos a través de diferentes plataformas, como las aplicaciones B2B, clientes de correo electrónico compatibles con servidores de noticias o dispositivos móviles.
- En aquellas situaciones en las que la información debe estar disponible para un gran número de clientes. Por ejemplo, titulares de noticias, comunicados de prensa de empresas, avisos y anuncios importantes, marcadores, listas de reproducción, calendarios de eventos o listas de correo.



Actualmente, XML se utiliza habitualmente para transferir datos entre distintas aplicaciones de bases de datos. La mayoría de los SGBD, incluyendo Microsoft Access y Oracle, permiten exportar tablas de bases de datos como archivos XML.

1.3 ESTRUCTURA DE UN DOCUMENTO XML: etiquetas, atributos y valores

Un archivo XML es un archivo en formato texto, que **contiene etiquetas para identificar los elementos y datos que componen el documento**. Lo primero que hay que tener en cuenta a la hora de definir un archivo XML es que la primera línea del archivo debe ser la siguiente:

```
<? Xml version = "1.0" encoding = "ISO-8859-1"?>
```

El resto del documento XML se escribirá con etiquetas, y siempre es necesario abrirlas y cerrarlas:

```
<etiql> ..... </etiql>  
<etiql2> ..... </etiql2>  
<etiql3> ..... <etiql4> ..... </etiql4>..... </etiql3>
```

El conjunto formado por las etiquetas (apertura y finalización) y el contenido se conoce como **elemento o nodo** (en el caso de realizar una representación jerárquica de los datos). Por ejemplo, el conjunto **<nom>Pere</nom>** es un elemento o nodo, con la etiqueta “nombre” y el contenido “Pere”.

Puede darse el caso de que el contenido de un elemento contenga otros elementos en vez de los datos en formato de texto, pero no podrá contener ambas cosas: los otros elementos y datos en formato de texto.

Los comentarios en XML se escribirán de esta forma:

```
<!-- ----- Comentari ----- -->
```

Este tipo de estructura de documento con las etiquetas y los atributos pide cuidar el modo de almacenar la información, estructurando muy bien el documento y ordenando adecuadamente las informaciones.

Un documento XML tiene una estructura imbricada de forma jerárquica. **Es obligatorio cerrar todas las etiquetas, y si un elemento tiene vinculados otros elementos (hijos)**, es decir, otros elementos que descienden deben cerrarse las etiquetas de los hijos antes de poder cerrar la etiqueta del padre.

Un documento XML está bien formado si dispone de una estructura sintácticamente correcta.

Para conseguir esto será necesario que la estructura jerárquica se cumpla de manera estricta en lo que se refiere a las etiquetas que se utilizan en el documento. Es decir, todas las etiquetas abiertas deben haber sido cerradas en el orden adecuado. Los valores de los datos o contenidos de los nodos se encuentran entre el texto que indica la apertura de la etiqueta y lo que indica su cierre.

ETIQUETAS

Todas las **etiquetas** deben estar correctamente cerradas, es decir, con una etiqueta de cierre que se corresponda con la de apertura. Las etiquetas deben estar **siempre** cerradas.

Un **ejemplo incorrecto** sería:

```
<menu>
  <element posicio="15;3">
    <nom>Menu
  <element posicio="66;30">
    <nom>Finestra
```

En cambio, véase ahora una **versión correcta**:

```
<menu>
  <element posicio="15;3">
    <nom>Menu</nom>
  </element>
  <element posicio="66;30">
    <nom>Finestra</nom>
  </element>
</menu>
```

Las **etiquetas vacías** (es decir, sin contenido) tienen una sintaxis especial:

```
<etiqueta/>
```

ATRIBUTOS

Los elementos pueden tener **atributos**, que son una forma de **incorporar características o propiedades a los elementos de un documento**.

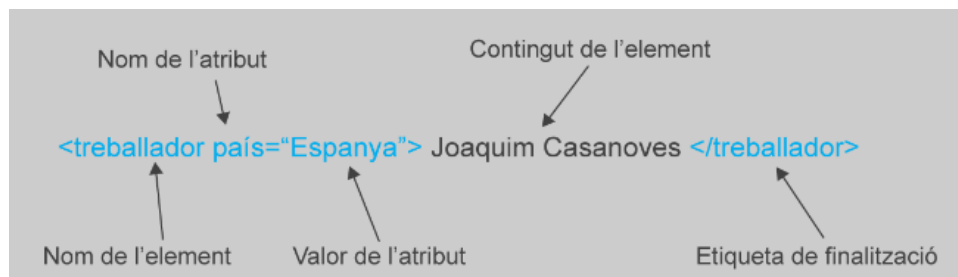
La forma de representar esta información será a partir de añadir a la etiqueta de apertura del nodo los atributos (tanto el nombre como los valores) que deseamos.

```
<gos color= "blanc"> Pastor alemany </gos>
```

Los valores de los atributos de los elementos siempre deben estar marcados con las **comillas dobles** (") o **simples** (').

```
<a href = "http://www.ioc.com"> És correcte </a>
<a href = 'http://www.ioc.com'> És correcte </a>
<a href = http://www.ioc.com> No és correcte </a>
```

Como puede verse en la siguiente imagen, en el caso del trabajador de nombre "Joaquim Casanoves" se ha añadido el atributo *país* para indicar su nacionalidad, y se indica además el valor de ese atributo.



El nombre de la etiqueta de un elemento, atributo, entidad... comienza por una letra, y continúa con letras, dígitos, guiones, rayas, punto o dos puntos.

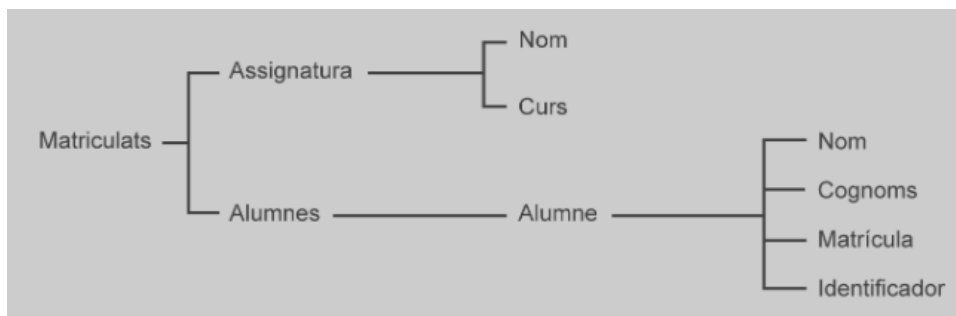
El texto *XML* (o *xml* o *xML*, etc.) no se puede utilizar como caracteres iniciales de un segundo nombre de elemento, atributo, entidad, etc.

XML es **case-sensitive** (sensible en la caja, el uso de mayúscula o minúscula), es decir, no es lo mismo `<autor>` que `<Autor>`.

Un ejemplo de XML completo y bien formado sería:

```
<?xml version = "1.0" encoding = "ISO-8859-1"?>
<Matriculats>
  <Assignatura>
    <Nom> Generacio interficies </Nom>
    <Curs> 2 </Curs>
  </Assignatura>
  <Alumnes>
    <Alumne>
      <Nom> Joan </Nom>
      <Cognoms> Garcia Castellejos</Cognoms>
      <Matricula> 060000 </Matricula>
      <Identificador Tipus="DNI"> 4060000 </Identificador>
    </Alumne>
    <Alumne>
      <Nom> Albert </Nom>
      <Cognoms> Lucas Martinez </Cognoms>
      <Matricula> 050000 </Matricula>
      <Identificador Tipus="NIF"> 3060000H </Identificador>
    </Alumne>
  </Alumnes>
</Matriculats>
```

De esta forma la estructura jerárquica del XML será la que se muestra en la figura con el nodo de *Matriculados* en el primer nivel, los alumnos y las asignaturas en segundo nivel, el nombre y curso de las asignaturas en el tercer nivel y los datos del alumno como tercer nivel de los alumnos. El nivel cuarto, sólo alcanzado por la rama del *Alumno*, nos indica los datos de éste (nombre, apellidos, matrícula e identificador).

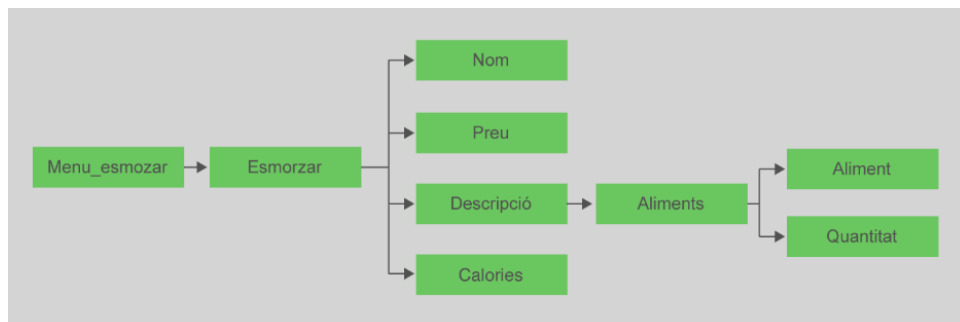


Otro ejemplo podría ser el siguiente:


```

<?xml version="1.0" encoding="utf-8"?>
<menu_esmorzar>
  - <esmorzar>
    <nom>Aida</nom>
    <preu>6,00 €</preu>
    <descripció> esmorzar americana
    <aliments>
      <aliment> ous </aliment>
      <quantitat> 2 </quantitat>
    </aliments>
    <aliments>
      <aliment> bacon </aliment>
      <quantitat> 3 </quantitat>
    </aliments>
    <aliments>
      <aliment> patates </aliment>
      <quantitat> 20 </quantitat>
    </aliments>
    </descripció>
    <calories>1200</calories>
  </esmorzar>
  - <esmorzar>
    <nom>Abril</nom>
    <preu>$4.95</preu>
    <descripció>
      Macedònia de fruita
    <aliments>
      <aliment> poma </aliment>
      <quantitat> 1 </quantitat>
    </aliments>
    <aliments>
      <aliment> taronja </aliment>
      <quantitat> 1 </quantitat>
    </aliments>
    <aliments>
      <aliment> plàtan </aliment>
      <quantitat> 1 </quantitat>
    </aliments>
    <aliments>
      <aliment> pera </aliment>
      <quantitat> 1 </quantitat>
    </aliments>
    </descripció>
    <calories>600</calories>
  </esmorzar>
</menu_esmorzar>

```



1.4 EDICI3N DE UN DOCUMENTO XML

HTML

HTML s3n les sigles d'Hyper Text Markup Language, 3s a dir, 'llenguatge de marques d'hipertext').

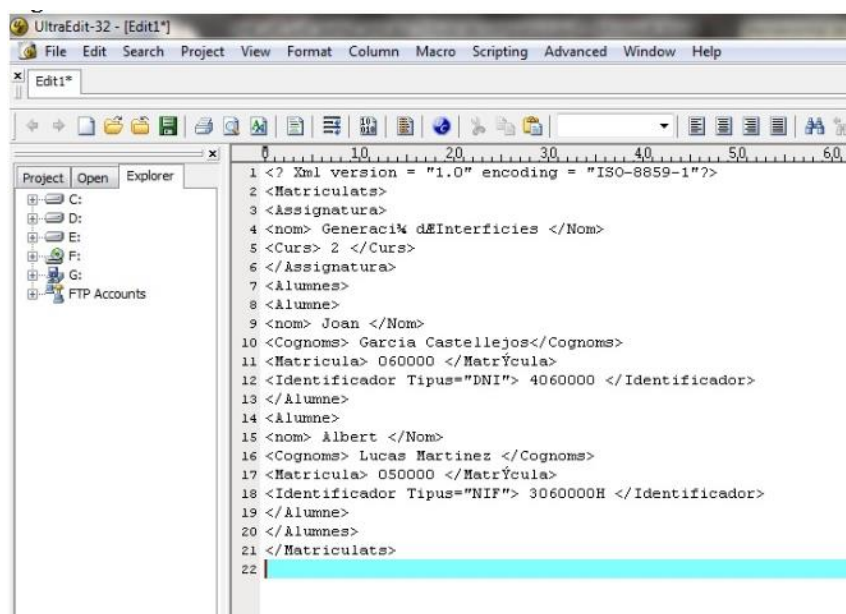
Un documento XML es un documento de **texto ASCII**, como muchos otros tipos de documentos que se desarrollan a partir de diferentes lenguajes de programaci3n (por ejemplo, el HTML). Este **lenguaje es interpretado por otros lenguajes o programas espec3ficos**, pero para editarlo no es necesaria ninguna herramienta especial.

Una aplicaci3n tan sencilla como un Edit en MSDOS o un Bloque de notas en un entorno Windows permiten la visualizaci3n y edici3n del documento XML.

En la figura se puede ver c3mo podemos editar el c3digo del ejemplo tratado anteriormente con una herramienta de edici3n como Ultraedit.

Ultraedit

L'Ultraedit 3s una eina de propietat d'edici3 de codi no integrada amb cap entorn de desenvolupament per3 que permet el reconeixement de molts llenguatges de programaci3, com l'XML. Per a Windows i per a Linux.



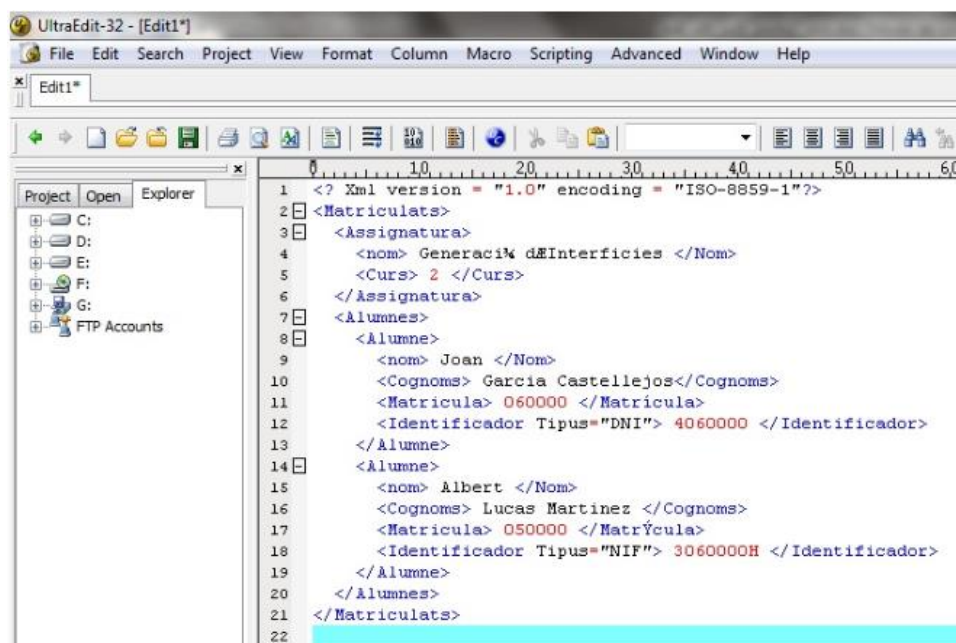
Otras herramientas permiten el reconocimiento de las etiquetas que se utilizan, aunque sea el desarrollador el que decida su nombre, lo que permite un desarrollo m3s sencillo y agradable. Algunos entornos integrados de desarrollo permitir3n este reconocimiento adem3s de permitir indicar si la sintaxis es la correcta o mostrar c3mo queda el resultado del c3digo una vez interpretado o ejecutado, en funci3n del lenguaje de programaci3n utilizado.

Pero, en otras ocasiones, interesa utilizar un editor de c3digo gen3rico que ofrezca algunas funcionalidades m3s que las que ofrecer3 un bloc de notas, sin llegar a necesitar un IDE o, sencillamente, porque no existe para un lenguaje de programaci3n determinado. Estas

funcionalidades pueden ser reconocimiento de lenguajes de programación o la posibilidad de escribir en formato de columnas.

Otra característica que permite un mejor tratamiento del código, tanto en la edición como en la comprensión, es el sangrar hacia la derecha aquellas etiquetas que forman parte de un mismo nivel jerárquico. El hecho de poder agruparlas ayudará mucho a la comprensión del código.

En la figura puede verse la edición del ejemplo con el editor UltraEdit, con el reconocimiento de las etiquetas y la sangría del código.



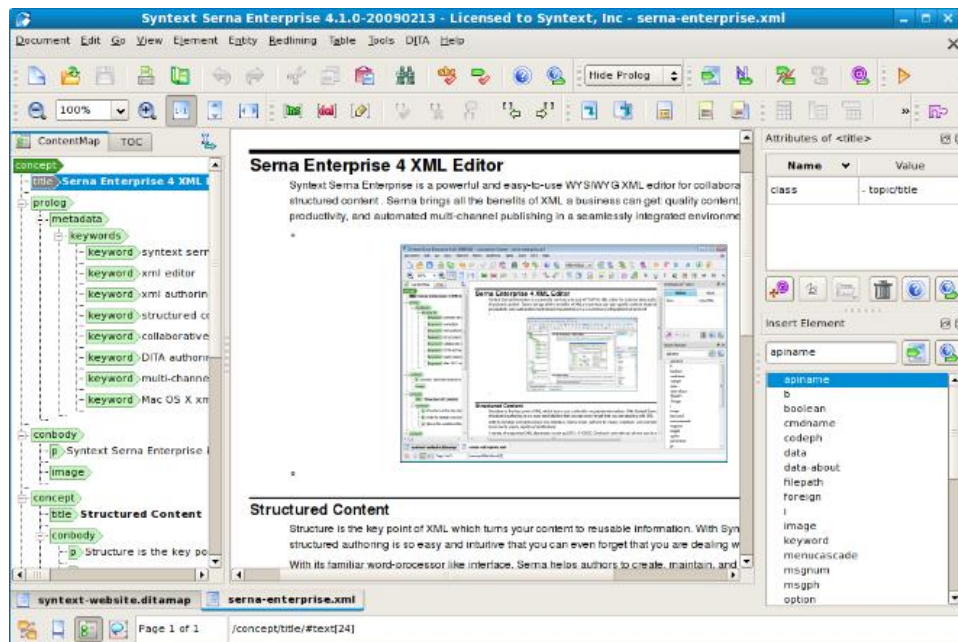
The screenshot shows the UltraEdit-32 interface with a menu bar (File, Edit, Search, Project, View, Format, Column, Macro, Scripting, Advanced, Window, Help) and a toolbar. The left pane shows a file explorer with drives C:, D:, E:, F:, G: and FTP Accounts. The main editor window displays XML code with syntax highlighting and indentation. The code is as follows:

```
1 <? Xml version = "1.0" encoding = "ISO-8859-1"?>
2 <Matriculats>
3   <Assignatura>
4     <nom> Generació d'Interfícies </Nom>
5     <Curs> 2 </Curs>
6   </Assignatura>
7   <Alumnes>
8     <Alumne>
9       <nom> Joan </Nom>
10      <Cognoms> Garcia Castellejos </Cognoms>
11      <Matricula> 060000 </Matricula>
12      <Identificador Tipus="DNI"> 4060000 </Identificador>
13    </Alumne>
14    <Alumne>
15      <nom> Albert </Nom>
16      <Cognoms> Lucas Martinez </Cognoms>
17      <Matricula> 050000 </Matricula>
18      <Identificador Tipus="NIF"> 3060000H </Identificador>
19    </Alumne>
20  </Alumnes>
21 </Matriculats>
22
```

Existen otras muchas herramientas específicas para la edición de XML. Algunas de ellas son:

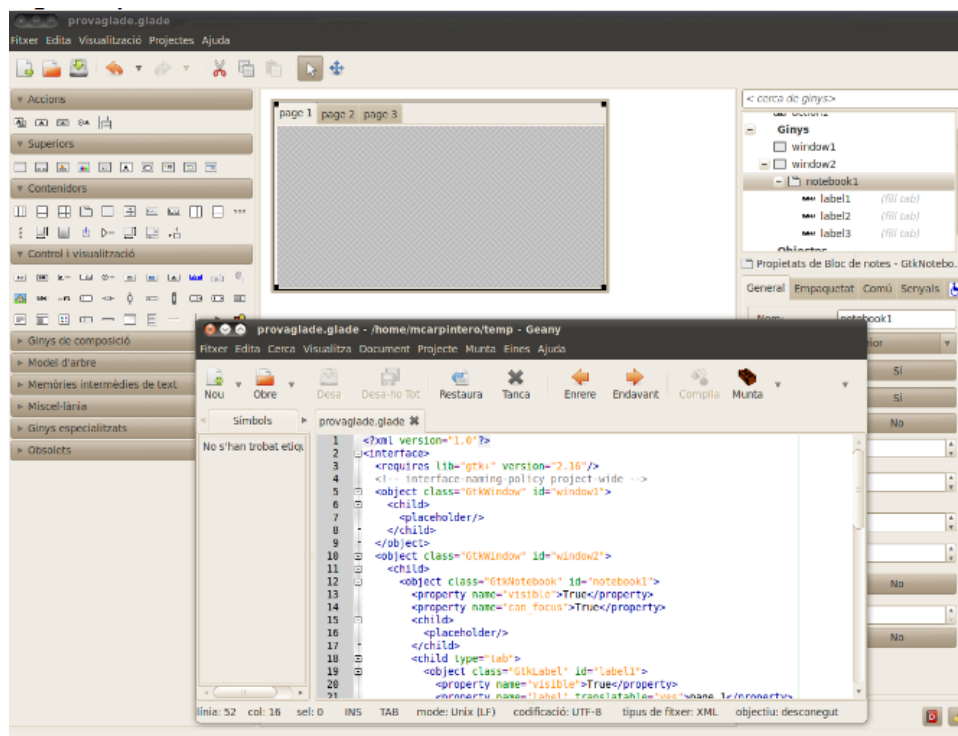
- **Syntax Serna.** Editor para Windows, Linux y Mac, herramienta con una versión libre que permite la edición de texto, la edición gráfica y la edición de tipos WYSIWYG.
- **XML Studio.** Editor para Windows, herramienta de propiedad que permite la edición de texto, la edición gráfica y la edición de tipos WYSIWYG.
- **Oxygen XML Editor.** Editor para Windows, MAC y Linux, herramienta de propiedad que permite la edición de texto, la edición gráfica y la edición de tipos WYSIWYG.
- **XML Notepad 2007.** Editor para Windows, con licencia pública de Microsoft, que permite la edición de texto, edición gráfica y edición de tipo WYSIWYG.

En la figura puede verse un ejemplo de la herramienta Syntax Serna.



Otra alternativa para editar documentos XML es hacerlo a partir de código generado por herramientas específicas, como MonoDevelop o Glade.

Glade es una herramienta libre que ayuda a la creación de interfaces gráficas de usuario. Una vez diseñada la interfaz, se puede generar el código correspondiente en XML, pudiendo éste editarse para conseguir modificaciones.



2. LENGUAJES DE DESCRIPCIÓN DE INTERFACES BASADOS EN XML

XML es un lenguaje que, por sí mismo, no ofrecerá el diseño de una interfaz gráfica de usuario. Necesitará otros lenguajes que aprovechen sus características para ofrecer el desarrollo de una GUI.

Es decir, existen lenguajes basados en el estándar XML que permiten describir cómo debe ser tratada la información que contiene un documento XML para presentarla en un medio como puede ser una página web (HTML) o cualquier otro documento estructurado.

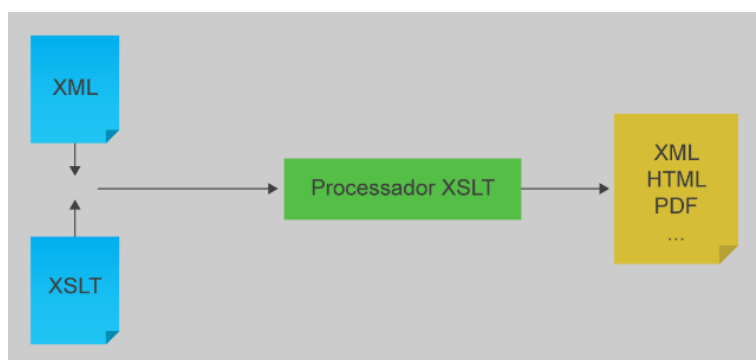
En este apartado se llevará a cabo una descripción de los lenguajes de programación basados en XML:

- XSLT (eXtensible Style Language Transformation)
- XUL (eXtensible User interface Language)
- XIML (eXtensible Interface Markup Language)

2.1 XSLT (eXtensible Style Language Transformation)

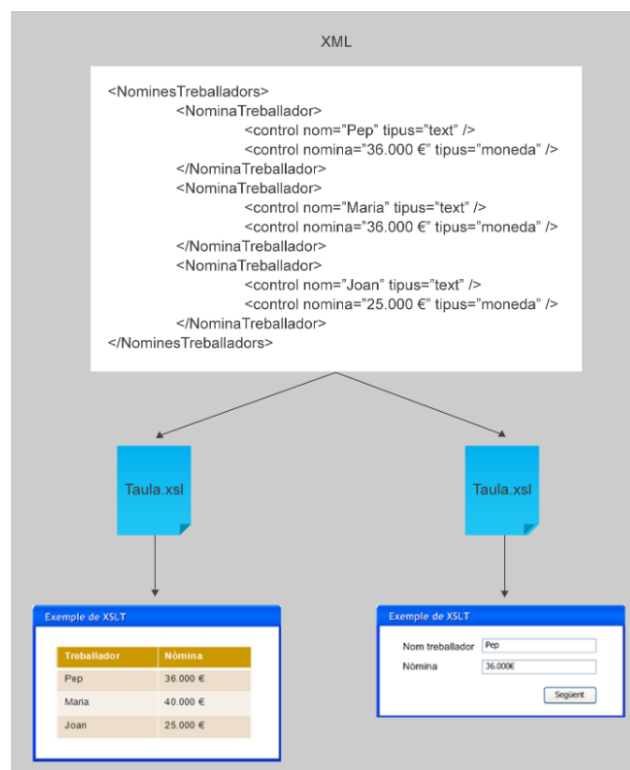
XSLT (o XSL Transformations) es un estándar de la organización W3C que presenta una forma de transformar documentos XML en otros tipos de documentos, incluso en formatos que no son XML.

En la figura se puede ver la forma de funcionar del estándar XSLT. Las hojas de estilo XSLT realizan la transformación del documento utilizando una o varias reglas de plantilla unidas al documento fuente a transformar.



XSLT está obteniendo, actualmente, mucho reconocimiento en la edición web, y genera páginas HTML o XHTML. La unión de XML y XSLT ayudará al aumento de la productividad por permitir la separación de contenido y presentación gráfica.

De esta forma se puede observar en las siguientes figuras que, a partir de un mismo XML, aplicando diferentes hojas XSLT, se puede representar la información con distintos formatos. También se observan dos interfaces distintas, con los datos de varios trabajadores, que tendrán el mismo origen, un único documento XML.



Para entender mejor el funcionamiento del XSLT se plantea el siguiente ejemplo:
Disponemos de un XML que trata de una agrupación de personas:

```

<?xml version="1.0" encoding="iso-8859-1"?>
<alumnes>
  <alumne>
    <nom>Esther Garcia</nom>
    <dni>38293450S</dni>
    <notaMitjana>7,5</notaMitjana>
  </alumne>
  <alumne>
    <nom>Carles Aries</nom>
    <dni>34839593G</dni>
    <notaMitjana>8,5</notaMitjana>
  </alumne>
  <alumne>
    <nom>Marc Fernandez</nom>
    <dni>30492839P</dni>
    <notaMitjana>5,2</notaMitjana>
  </alumne>
</alumnes>

```

Este documento XML contiene los datos (nombre, DNI y nota media) de tres alumnos, siguiendo una estructura jerárquica con Alumnos/Alumno/Nombre-DNI-notaMedia.

El documento XSL de transformación que se utilizará es:

```

<?xml version="1.0" encoding="iso-8859-1"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<html>
<body>
  <h2>Notes dels alumnes</h2>
  <table border="1">
    <tr bgcolor="#9acd32">
      <th>Nom</th>
      <th>Nota Mitja</th>
    </tr>
    <xsl:for-each select="alumnes/alumne">
      <tr>
        <td>
          <xsl:value-of select="nom"/>
        </td>
        <td>
          <xsl:value-of select="notaMitjana"/>
        </td>
      </tr>
    </xsl:for-each>
  </table>
</body>
</html>
</xsl:template>
</xsl:stylesheet>

```

Un documento XML se compone de información estructurada en nodos en forma de árbol. Si se tiene en cuenta que un XSLT debe recorrer un documento XML, entenderemos que la sintaxis básica para el desarrollo de XSLT son recorridos, bucles y condiciones que navegan por la estructura del XML. En este ejemplo, se hará un recorrido por el documento XML recogiendo los datos de los alumnos referente en los nombres y en la nota media, dejando de lado la información referente al DNI.

La cabecera para todos los documentos será:

```

<xsl:stylesheet version="1.0"
  xmlns="http://www.w3.org/1999/xhtml"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

```

En la que se indica una hoja de estilo XSL mediante la etiqueta stylesheet para todos los documentos que hacemos. Indicamos la versión 1 de la hoja de estilo con el atributo version. En caso de cumplir las recomendaciones del W3C se especificará con **xmlns=http://www.w3.org/1999/xhtml** y finalmente su namespace (criterio y restricciones de las recomendaciones sobre la estructura de tipos de elemento y nombres del atributo), con xmlns y xmlns:xsl.

En el ejemplo se está transformando el documento XML en un HTML compuesto por una tabla con dos columnas: el nombre del alumno y su nota media.

El resultado de la transformación será un documento HTML como el siguiente:


```

<html>
  <body>
    <h2>Notes dels alumnes</h2>
    <table border="1">
      <tr bgcolor="#9acd32">
        <th>Nom</th>
        <th>Nota Mitja</th>
      </tr>
      <tr>
        <td>Esther Garcia</td>
        <td>7,5</td>
      </tr>
      <tr>
        <td>Carles Aries</td>
        <td>8,5</td>
      </tr>
      <tr>
        <td>Marc Fernandez</td>
        <td>5,2</td>
      </tr>
    </table>
  </body>
</html>

```

En la figura 2.12 se puede ver cuál es el resultado de este documento HTML si lo abrimos con un navegador. Obtendremos una tabla con 4 filas y dos columnas; la primera de las filas contiene la cabecera y el resto de las filas el nombre de los alumnos y su nota media.

Notes dels alumnes

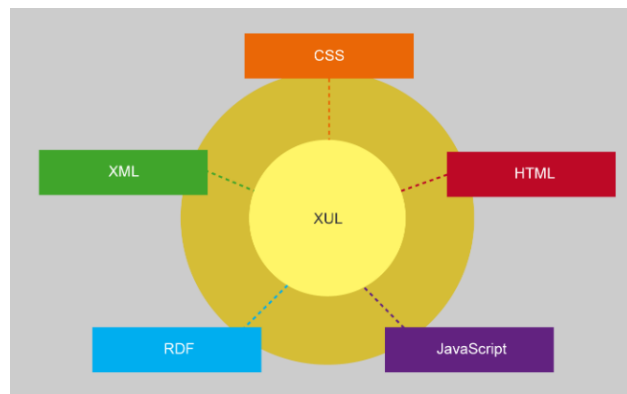
Nom	Nota Mitja
Esther Garcia	7,5
Carles Aries	8,5
Marc Fernandez	5,2

2.2 XUL (eXtensible User Interface Language)

XUL es el acrónimo de eXtensible User interface Language; traducido del inglés significa 'lenguaje de interfaz de usuario extensible'. Este lenguaje ha sido desarrollado para crear interfaces de usuario a partir de documentos XML para los entornos Netscape y Mozilla.

XUL utiliza los elementos de un lenguaje de marcas para crear interfaces gráficas de usuario, como lo hace otro lenguaje como HTML. Se podrá definir la apariencia de esta interfaz con hojas de estilo CSS y usar JavaScript para manipular su comportamiento. Además, XUL tiene un conjunto extenso de componentes gráficos usados para crear menús, barras de herramientas, cajas de texto, entre otros muchos componentes. Este último punto lo diferencia del lenguaje HTML.

XUL, además de ser un lenguaje con el que es posible crear interfaces de forma fácil, sencilla y rápida, está disponible para todas las versiones de Windows, Mac, Linux y UNIX. Cabe decir que, actualmente, todavía no es compatible al 100% con el software de Microsoft Internet Explorer, lo que dificulta su extensión. En la figura se hace una muestra de los diferentes elementos con los que puede interactuar un documento desarrollado en XUL, como son el XML, el CSS, el JavaScript, el HTML y el RDF.



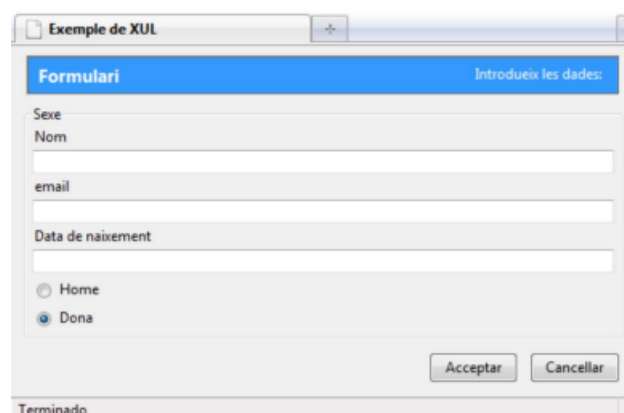
A continuación, se desarrolla un ejemplo en XUL que implementará una interfaz muy sencilla. Se trata de un pequeño formulario preparado para mostrar los datos contenidos en un documento XML.

Para ello, es necesario crear un archivo llamado ejemploXUL.xul (con un editor de texto como podría ser el Notepad, Ultraedit...) al que se añade el siguiente código:

```
<?xml version="1.0"?>
<?xml-stylesheet href="chrome://global/skin/global.css" type="text/css"?>
<dialog id=" IdentificadorDelFormulari" title=" Exemple de dialeg"
  xmlns="http://www.mozilla.org/keymaster/gatekeeper/there.is.only.xul"
  buttons="accept,cancel"
  buttonlabelcancel="Cancelar"
  buttonlabelaccept="Aceptar"
  ondialogaccept="return doOK();"
  ondialogcancel="return doCancel();">

  <dialogheader title="Options" description="My preferences"/>
  <label value="Nom"/>
  <textbox />
  <label value="email"/>
  <textbox />
  <label value="Data de naixement"/>
  <textbox />
  <groupbox>
    <caption label="Sexe"/>
    <radiogroup>
      <radio label="Home"/>
      <radio label="Dona" selected="true"/>
    </radiogroup>
  </groupbox>
</dialog>
```

Posteriormente se abrirá el navegador Mozilla y arrastraremos el archivo ejemplo-XUL.xul hacia el navegador, y veremos el formulario que se muestra en la figura.



Llegados a este punto, vamos a analizar el código. Cualquier archivo XML, independientemente de su función, debe tener algunas líneas al principio que indican la

versión de XML, y una hoja de estilo si es oportuno. En nuestro ejemplo tenemos las dos líneas siguientes:

```
<?xml version="1.0"?>
<?xml-stylesheet href="chrome://global/skin/global.css" type="text/css"?>
```

Como puede observarse, tenemos una línea de versión que indica que la versión XML es la 1.0. La segunda línea indica que nuestra hoja de estilo está en la ruta "chrome". La ruta "chrome" contiene algunas utilidades internas de Mozilla que gestionan habitualmente las interfaces de usuario de Mozilla. Las siguientes líneas contienen nuestras primeras etiquetas:

```
<dialog id="IdentificadorDelFormulari" title="Exemple de diàleg"
xmlns=http://www.mozilla.org/keymaster/gatekeeper/there.is.only.xul>
</dialog>
```

Cada página XUL que se crea necesita una etiqueta tipo **<window>** o **<dialog>**, entre otras, que puedan ser usadas para contener los componentes que forman la interfaz que queremos desarrollar. Dentro de esta etiqueta se han utilizado tres atributos.

- El primer **id** (identificador) es una referencia única que apunta a esta etiqueta en XML. El atributo **id** es esencial para poder comunicarnos con etiquetas y actualizar con cambios e información.
- La segunda etiqueta, **title** (título), contiene una serie de caracteres legibles por humanos que se muestra en la barra de título cuando lanzamos el archivo XUL en la ventana oportuna.
- El último atributo es la parte **xmlns**. Este atributo especifica el namespace (espacio de nombres) de la etiqueta **<window>**, **<dialog>** ... en la que se basarán todas las etiquetas que contenga.

Namespace

Un namespace és com un grup especial que podem especificar per determinar d'on ve una etiqueta. Això ajuda en situacions en què es tingui una etiqueta **<window>** d'un altre llenguatge XML i una etiqueta **<window>** del llenguatge XUL: el namespace les diferencia.

Ahora se podrá rellenar con datos el formulario desarrollado antes. En nuestro ejemplo hemos creado un pequeño formulario: Creamos un **<label>** (etiqueta) con el valor "Nombre", en el que el usuario podrá escribirlo dentro del campo **<textbox>** (cuadro de texto)

```
<label value="Nom"/>
<textbox />
```

Del mismo modo el usuario podrá indicar la dirección electrónica y la fecha de nacimiento.

Por otra parte, se crea un **<groupbox>** (agrupación de casillas) para indicar el sexo.

```
<groupbox>
  <caption label="Sexe"/>
  <radiogroup>
    <radio label="Home"/>
    <radio label="Dona" selected="true"/>
  </radiogroup>
</groupbox>
```

2.3 Los Eventos

En muchas ocasiones, es necesario asociar eventos a elementos del formulario que se está desarrollando. En este ejemplo se utilizará JavaScript para añadir funcionalidad a los componentes. Esto se hace de forma muy similar a la del HTML. En el HTML, un conductor de eventos se asocia con un elemento y se realiza alguna acción cuando se activa el conductor. La mayoría de los conductores de HTML también se encuentran en XUL, y hay algunos más que sólo se encuentran en XUL.

Se puede añadir código dentro del archivo de XUL, pero una forma más eficiente es hacerlo en un archivo separado. Esto permite que la carga de la interfaz se realice más rápidamente.

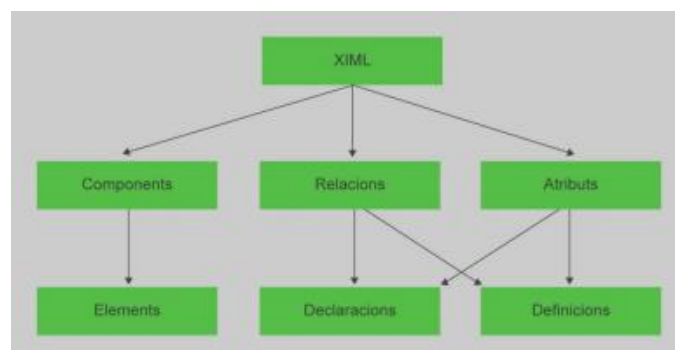
Un ejemplo sencillo de JavaScript asociado a nuestro ejemplo sería:

```
<script language= "JavaScript" >
    function doOK(){ alert("Desa el dialeg"); }
    function doCancel(){ alert("Tanca el dialeg"); }
</script>
```

2.4 XIIML (eXtensible Interface Markup Language)

Otro ejemplo de lenguaje para la creación de interfaces a partir de documentos XML es el XIIML. Es un acrónimo de eXtensible Interface Markup Language ('lenguaje de marcas para la creación de interfaces extensible'). Este lenguaje es independiente de la plataforma en la que se utilizará, poniendo especial énfasis en desacoplar la representación gráfica de los datos de la interacción con el usuario.

En la figura puede verse cuál será la estructura del lenguaje XIIML.



Esta estructura está compuesta por componentes, relaciones y atributos. Los componentes agruparán los elementos de las interfaces. Las relaciones definirán las interacciones entre distintos elementos y los atributos dispondrán de valores que definirán los elementos. Cabe remarcar la importancia de la estructura jerarquizada del XIIML.

COMPONENTES DEL XIIML

El lenguaje XIIML está compuesto por un conjunto de elementos de interfaz de usuario organizados que se encuentran clasificados en uno o varios componentes de interfaces principales. El lenguaje XIIML no limita el número y el tipo de componentes que podrán definirse en un documento.

Se definen cinco tipos de componentes de interfaz en la primera versión del lenguaje XIML: presentación, dominio, tarea, diálogo y usuario.

- **Presentación.** El componente de interfaz de presentación sirve para definir una jerarquía de elementos. Éstos serán los componentes que se mostrarán al usuario. En este componente no se especificará el aspecto y las propiedades que tendrán que tener estos elementos.
- **Dominio.** El componente dominio representa un conjunto de instancias y clases. Este conjunto estará organizado en una jerarquía. Esta jerarquía es conceptualmente como una ontología, pero mucho más sencilla: habrá una definición de los objetos a partir de atributos y sus valores. Los objetos, que podrán ser simples o complejos, sólo se considerarán objetos si el usuario los ha visualizado o modificado.
- **Tarea.** Este componente representa los procesos de negocio en los que interviene un usuario junto con las acciones que podrá desarrollar ese usuario en su interacción. Un componente tarea define una descomposición jerárquica de tareas y subtareas que definen el flujo de trabajo de estas tareas y los datos que utilizan.
- **Diálogo.** Este componente define una colección estructurada de elementos que determinan las acciones de interacción que puede realizar el usuario. Este componente representará la implementación de las tareas y sus acciones, así como la navegación del usuario por la aplicación.
- **Usuario.** El componente usuario representa la jerarquía de usuarios que deberá tener en cuenta el XIML. Los elementos de este componente serán nodos que podrán identificar a un único usuario que navegará por las interfaces de la aplicación, o varios usuarios (un grupo de usuarios relacionado, con características comunes).

El lenguaje XIML ofrece la posibilidad de añadir nuevos elementos y componentes a su estructura. Al estar compuesto por una estructura jerárquica, está listo para soportar añadir estos nuevos elementos.

RELACIONES

Las relaciones en el lenguaje de programación XIML son una definición que vincularán dos o más elementos XIML que se encuentren dentro de un mismo componente o distribuidos entre distintos componentes.

Los elementos y componentes son muy importantes en XIML, ya que ofrecen una información o conocimiento básico en lo referente a las interfaces de usuario. Pero las relaciones entre los elementos son también, en sí mismas, una información o conocimiento también igual de básico que lo que ofrecen los elementos.

Los tipos de relaciones que contiene XIML son:

- Tipo **definición**, es decir, relaciones que especifican la forma canónica de la relación.
- Tipo **declaración**, es decir, relaciones que especifican la instancia actual de la relación.

ATRIBUTOS

Dentro del lenguaje de programación XIML, los atributos son unos valores que podrán asignarse a algunos elementos. El significado de los atributos vinculados a los elementos indicará sus propiedades o características.

En XIML, los valores que se pueden asignar a los atributos podrán ser de dos tipos:

- **Valores de tipos básicos** de datos (enteros, caracteres, cadenas de caracteres...)
- **Instancias a otros elementos** existentes en el mismo componente o en otros componentes

OTRAS CARACTERÍSTICAS DE XIML

A continuación, se presentan otras características importantes del lenguaje de programación XIML para tener en cuenta:

- El lenguaje XIML permite mostrar un mismo elemento o un componente de muchas formas diferentes, de modo que el usuario podrá escoger las propiedades y el aspecto más conveniente.
- Otra característica interesante es la posibilidad de modificar el aspecto de las interfaces de usuario de forma dinámica, y entonces da la posibilidad de adaptarse a los cambios que puedan producirse en los dispositivos de visualización. Ésta es una característica que también se puede encontrar en los XForms; eso sí, limitados a tres estados. Con el lenguaje XIML no hay límite de estados por definir. Un ejemplo para esta característica es la posibilidad de adaptar las interfaces a un cambio de tamaño de visualización.
- El lenguaje XIML permite la creación de presentaciones, una o varias. Cada presentación se puede vincular a un tipo de dispositivo con características diferentes (PC de sobremesa, portátiles de pocas pulgadas, dispositivos móviles, teléfonos...). Esto se logra gracias a la posibilidad de desvincular completamente la definición de las interfaces de la visualización. Cada presentación se utilizará (se activará) en el momento que el sistema detecte el dispositivo en el que se utilice la aplicación.
- Otra característica es la posibilidad de utilizar las múltiples visualizaciones que ofrecen las empresas externas, y que podrán utilizar los usuarios de las interfaces sin necesidad de instalaciones ni descargas.