DESARROLLO DE INTERFACES

TÉCNICO EN DESARROLLO DE APLICACIONESMULTIPLATAFORMA

Interfaces a partir de XML

El lenguaje XML se suele usar para:

- **Intercambiar de información entre aplicaciones**.
- **Computación distribuida**: intercambiar información entre diferentes ordenadores a través de **redes**.
- **Información empresarial**: generar **interfaces empresariales estructurando** los **datos** de la forma **más apropiada** para cada empresa.

Conseguiremos **generar interfaces** a partir de documentos XML.

Lenguajes de descripción de interfaces basados en XML

XML (**eXtensible Markup Language**) maneja datos:

- **estructurándolos.**
- **almacenándolos.**
- **intercambiándolos.**

Es un metalenguaje, y **define** otros

lenguajes

XHTML

EXtensible HyperText Markup Language: similar a HTML, pero **más robusto y aconsejable** para la modelación de **páginas web**.

Elementos obligatorios:

- <! DOCTYPE>
- <html **xmlns**> atributo **xmlns.**
- <html>, <head>, <title> y <body>.

Características de diseño de sus elementos:

- Correctamente **anidados**.
- **Cerrados**.
- En **minúsculas** los elementos y los nombres de los atributos.
- Los **valores** de los **atributos siempre** se deben citar.
- **Prohibido minimizar atributos**.

En el siguiente ejemplo se muestra el prototipo de un documento redactado con formato XHTML en el que podemos comprobar que se cumplen todas las características descritas.

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
        <cabeza>
                <title>Un documento XHTML</title>
        </cabeza>
        <cuerpo>
                <h1>Cabecera principal del documento</h1>
                <p>Nuestro primer párrafo</p>
                <h2>Cabecera secundaria</h2>
                <p>Otro párrafo con contenido distinto</p>
        </cuerpo>
</html>
```

GML

GML (**Geography Markup Language**). Puede **recibir un formato** que define el **tipo de texto** que es (h1, p, lo, li...). Estos tipos de documentos están compuestos de **marcas precedidas de doble punto(:)**. Ejemplo:

:h1.Lenguaje XML

:p.Lenguajes basados en XML

:ol

:li.GML

:li.MathML

:li.RSS

:li.SVG

:li.XHTML

:eol.

MathML

El lenguaje MathML (**Mathematical Markup Language**) se usa junto con el lenguaje XHTML y se basa en el intercambio de información de tipo matemático entre programas.

```
<math>
        <mi>x</mi> <mo>=</mo>
        <mrow>
                <mfrac>
                        <mrow>
                                <mo>−</mo>
                                <mi>b</mi>
                                <mo>±</mo>
                                <msqrt>
                                        <msup><mi>b</mi><mn>2</mn></msup>
                                        <mo>−</mo>
                                        <mn>4</mn><mi>a</mi><mi>c</mi>
                                </msqrt>
                        </mrow>
                        <mrow> <mn>2</mn><mi>a</mi> </mrow>
                </mfrac>
        </mrow>
        <mtext>.</mtext>
</math>
```

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}.$$

RSS

Really Simple Syndication: Difusión de información entre los usuarios suscritos a una **fuente de contenidos** actualizada frecuentemente. Ejemplo:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<rss version="1.0">
        <channel>
                <title>Última hora</title>
                <description>
                        Noticia importante
                </description>
                <link>
                        https://elpais.com/ultimas-noticias/
                </link>
                <lastBuildDate>
                        Mon, 06 Jan 2020 00:10:00
                </lastBuildDate>
                <pubDate>
                        Mon, 06, Jan 2020 16:20:00 +0000
                </pubDate>
        </channel>
</rss>
```

XSLT

EXtensible Stylesheet Languaje for Transformation: Parecidas a **hojas de estilo** CSS, pero **dirigidas a XML**. Presenta un funcionamiento más completo que CSS, puesto que **permite agregar o eliminar elementos y atributos** desde un archivo.

Además, permite realizar pruebas e, incluso, tomar decisiones sobre los elementos que se han de mostrar u ocultar.

SVG

Scalable Vector Graphics: Representa elementos geométricos vectoriales, imágenes de mapa de bits y texto.

circle

```
<!DOCTYPE html>
<html>
<body>
        <svg height=”100” width=”100”>
        <circle cx=”50” cy=”50” r=”40” stroke=”black”
        stroke-width=”3” fill=”cyan” />
        </svg>
</body>
</html>
```

rect

```
<svg width=”60” height=”60”>
        <rect x=”0” y=”0” width=”60” height=”60” fill=”red”/>
</svg>
```

ellipse

```
<svg width=”60” height=”60”>
        <ellipse cx=”30” cy=”30” rx=”20” ry=”16” fill=”orange”/>
</svg>
```

polygon

```
<svg width="60" height="60">
        <polygon fill="green" stroke="black" stroke-width="2" points="05,30
        15,10 25,30"/>
</svg>
```

El documento XML.

Intro

La **primera línea del fichero debe ser la siguiente**:

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

• Version: Versión de XML.

• Encoding: **Juego de caracteres**.

El resto del documento XML se escribirá con **etiquetas**, y siempre hay que abrirlas y cerrarlas:

<etiqueta> (...) </etiqueta>

El conjunto formado por las etiquetas (apertura y finalización) y el contenido se conoce como **elemento** o **nodo** (en el caso de hacer una representación jerárquica de los datos).

Por ejemplo, el conjunto

<nombre> Lucas </nombre>

es un elemento o nodo, con la etiqueta 'nombre' y el contenido 'Lucas'.

Estructura de un **elemento o nodo** (pueden estar anidados con otros elementos en su interior):

<nombre-del-elemento *atributo* = "valor-del-atributo" >Contenido del elemento</nombre-del-elemento>

Estructura anidada de manera **jerárquica**. **Etiquetas** para que **queden todas cerradas**. Las etiquetas de los descendientes/hijos deben cerrarse antes que la del padre.

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 el **cierre**.

Etiquetas

Son marcas que sirven para diferenciar un contenido específico del resto del contenido del documento.

Empieza con '<", continúa un nombre identificativo, y termina con ">".

Siempre debe empezar por una letra, pero, a continuación, pueden utilizarse: **letras, dígitos, guiones, rayas, punto o dos puntos**. Existen **tres tipos de etiquetas**:

• **Start-Tag**: Etiquetas de apertura. (<etiqueta>)

• **End-Tag**: Etiquetas de cierre, similar a las de apertura, pero **comienzan por "/"**. (</etiqueta>)

• **Empty-Tag**: Etiquetas vacías, que terminan por "/". (<etiqueta_vacía />)

Atributos

Es un componente de las etiquetas. Consiste en un **par 'nombre=valor'**. Se puede encontrar en las etiquetas de apertura o en las etiquetas vacías, pero no en las de cierre. **No pueden existir dos atributos con el mismo nombre**, y todos los atributos de un elemento **siempre son únicos**. Por ejemplo:

<conductor nombre="Marcos" apellido-1="García" apellido-2="López" />

En este caso tenemos tres atributos únicos, nombre, apellido-1 y apellido-2.

En cambio, en el siguiente caso, **no sería correcto**, dado que tenemos el atributo apellido repetido:

<programador nombre="Marcos" apellido="López" apellido="García">

Valores

El **atributo** de un elemento XML **proporciona información** acerca del elemento, es decir, sirve para definir las propiedades de los elementos. La **estructura de un atributo** XML es siempre un **par de 'nombre=valor'**.

```
<biblioteca>
    <texto tipo_texto="libro" titulo="Soft Skills:The software developer's life manual" editorial="Manning Publications">
        <tipo>
            <libro isbn="978-1617292392" edicion="1" paginas="503"/>
        </tipo>
        <autor nombre="John Sonmez"/>
    </texto>
</biblioteca>
```

En el ejemplo observamos que los ***elementos*** aparecen coloreados en rojo (biblioteca, texto, tipo), los **nombres** de los atributos en blanco (tipo_texto, título, editorial, isbn, edición, páginas) y sus ***valores*** en azul.

Eventos

Ugracias a los eventos. el usuario podrá llevar a cabo una determinada funcionalidad. Eventos dan lugar a **interfaces dinámicas**.

Trata las diferentes formas de interacción entre el usuario y la aplicación. De todos los posibles eventos en una aplicación, elegiremos cuales queremos usar para una determinada funcionalidad.

Ejemplos de eventos son:

- **MouseMove**: se produce al mover el ratón por encima de un control.
- **MouseDown**: se produce al pulsar **cualquier botón** del ratón.
- **Change**: se produce al **cambiar el contenido del control**.
- **Click**: se produce al hacer clic sobre el control con el **botón izquierdo** del ratón.
- **GetFocus**: se produce cuando el elemento **recibe el foco** de atención, normalmente se utiliza para introducir datos o realizar alguna operación en tiempo de ejecución.
- **LostFocus**: se produce cuando el elemento de control **pierde** el foco.

Un ejemplo de uso de eventos sería si queremos que un texto se ponga de color rojo al situarnos encima, y de color gris al salir, podríamos usar **MouseEntered** y **MouseExited**; el primer evento se encargaría de poner el texto de color rojo y, el segundo, de ponerlo de color gris.

Herramientas para crear interfaces de usuario multiplataforma

Notepad ++

Reconoce la sintaxis de **múltiples lenguajes** de programación. Es gratuito y disponible para **Linux y Windows**.

Ha triunfado bastante entre la comunidad de desarrolladores web por las características que ofrece y por lo ligero que es (ocupa poco espacio y es muy rápido). Se puede extender a través de plugins. Posee uno llamado XML Tools que añade un nuevo menú con numerosas opciones como, por ejemplo, validar un documento XML con su DTD. Su interfaz es minimalista, pero los desarrolladores pueden personalizarla.

Atom

Herramienta multinivel, tanto para novatos como para **profesionales**. Actualmente es uno de los editores **preferidos** para programadores. Se pueden añadir lenguajes que no se incluyen de serie o añadir distintos tipos de interfaces gráficas.

Cada ventana de Atom es, en esencia, una página web renderizada localmente, y el espacio de trabajo se compone de paneles que pueden recolocarse de manera flexible para que la programación resulte más cómoda.

Teletype (Atom)

Para colaborar en tiempo real , permite que muchos desarrolladores puedan editar un archivo a la vez, en tiempo real, de forma colaborativa.

Su funcionamiento se basa en que el **usuario con rol de anfitrión** comparte su código con el resto de usuarios. Para lograrlo, se utiliza un código generado por el anfitrión y que este comparte con los invitados. Cada uno de ellos podrá editar el código compartido en tiempo real y visualizar las modificaciones del resto del equipo.

Git y GitHub (plugin en Atom)

Permiten controlar distintas versiones de un proyecto mientras se está desarrollando. El proyecto en el que estamos trabajando podrá **sincronizarse automáticamente** con el repositorio Git y podremos visualizar en todo momento **si estamos trabajando en la misma versión** que se encuentra en el **repositorio** o qué **diferencias** existen.

Adobe Dreamweaver CC

Admite tanto el **método textual** como el **WYSIWYG (What You See Is What You Get)**. Esta es una frase aplicada a los editores de código que permiten escribir un documento **mostrando directamente el resultado final**. ¿Presentación visual en vivo? ¿o seguir el camino clásico?

Admite todos los lenguajes de programación importantes. Está disponible para **Windows y OS X**. Permite **confirmar el código y accesibilidad** de la página, característica que puede facilitarles seguir las **pautas** de **accesibilidad** de contenido web.

Visual Studio Code

De los **más utilizados**. Trabaja con **varios lenguajes** de programación y está disponible para **Windows, Linux y macOS**.

Una de sus principales características es el resaltado de sintaxis, además de la finalización inteligente de código, la interfaz personalizable y que es gratuito (aunque existe la versión de pago). Pertenece al software de Visual Studio y una de las novedades más potentes que ofrece es el servicio **Live Share**, extensión que permite compartir código base con un compañero de equipo, de forma que se puede **colaborar en tiempo real** (parecido al *teletype* de atom).

Generación de código para diferentes plataformas

El intercambio de datos con XML ayuda a que los datos pueden ser leídos por **diferentes plataformas**.

Al ser XML un lenguaje independiente de la plataforma, significa que **cualquier programa diseñado para lenguaje XML** puede leer y **procesar los datos XML** independientemente del hardware o del sistema operativo que se esté utilizando.

Es por eso una de las tecnologías más utilizadas como base para el **almacenaje** de contenidos, como modelo de representación de **metadatos**, y como medio de **intercambio** de contenidos:

• **XML para el almacenamiento de contenidos**: en los últimos años está creciendo la demanda de **bases de datos XML nativas**, es decir, bases de datos que almacenan y gestionan documentos XML directamente, sin ningún tipo de transformación previa.

• **XML como medio de intercambio de contenidos**: la integración que permite el lenguaje XML en diferentes plataformas se basa en la facilidad de intercambio de contenidos dado que, al utilizar documentos basados en este lenguaje, se puede procesar para múltiples fines: como integración en una base de datos, visualización como parte de un sitio web o mensajes entre aplicaciones.

• **XML para la representación de metadatos**: lo más importante para representar metadatos es el sistema de **indexación y recuperación**, para poder discriminar dentro de un contenido los **elementos o atributos** que se desea recuperar.

La generación de código para diferentes plataformas es posible *debido a que:*

- *Es un estándar* ***abierto****,* ***flexible*** *y utilizado a* ***nivel global*** *para almacenar, publicar e intercambiar todo tipo de información.*
- *Proporciona* ***portabilidad*** *y* ***facilita la gestión de la información*** *a través de distintas plataformas.*
- *Permite que* ***diversas aplicaciones de datos*** *puedan funcionar de forma* ***independiente****.*
- *Es soportado por multitud de aplicaciones en diferentes plataformas. Además, existen varias* ***bibliotecas para diferentes lenguajes*** *de programación que permiten desarrollar nuevas aplicaciones. Es un* ***metalenguaje*** *de "carácter* ***universal****".*
- *El* ***formato*** *es* ***legible*** *tanto por humanos como por sistemas informáticos.*

*XML permite el **intercambio** entre plataformas del conjunto de **datos estructurados** que componen una interfaz de **forma simple, rápida y segura**.*