MEDAC
Instituto Oficial de Formación Profesional

DESARROLLO DE INTERFACES

TÉCNICO EN DESARROLLO DE APLICACIONES MULTIPLATAFORMA

Generación de interfaces a partir de documentos XML

03

ÍNDICE

ÍNDICE

OBJETIVOS

Reconocer las ventajas de generar interfaces de usuario a partir de su descripción XML.

Generar la descripción del interfaz en XML usando un editor gráfico.

Analizar el documento XML generado.

Modificar el documento XML.

Generar el código correspondiente al interfaz a partir del documento XML.

Programar una aplicación que incluya la interfaz generada.

/ 1. Introducción y contextualización práctica

En la actualidad, la generación de sistemas que permitan compartir todo tipo de información de una forma fiable y sin un coste elevado resulta clave. Esto es posible a través de la tecnología XML, la cual permite esta compatibilidad entre sistemas, puesto que puede utilizarse en bases de datos, editores de textos, hojas de cálculo y diferentes plataformas.

Este lenguaje hoy en día es muy importante por diversos usos, entre los que destacan:

- **Intercambio de información entre aplicaciones:** al almacenar información mediante documentos de texto plano, no se requiere software especial.
- **Computación distribuida:** se trata de la posibilidad de utilizar XML para intercambiar información entre diferentes ordenadores a través de redes.
- **Información empresarial:** este lenguaje tiene cada vez más importancia para generar interfaces empresariales gracias a la facilidad de estructurar los datos de la forma más apropiada para cada empresa.

El propósito de este tema es servir de guía para generar interfaces a partir de documentos XML, para lo cual se estudiarán los lenguajes basados en XML y los principales editores de documentos XML.

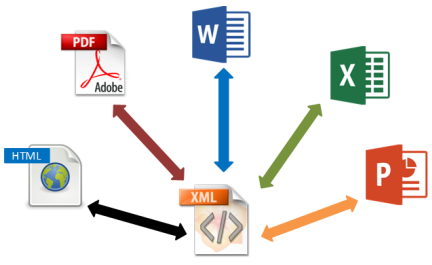

Fig. 1. Usos del lenguaje XML.

Escucha el siguiente audio donde contextualizaremos el caso práctico del tema y que encontrarás resuelto en el último apartado de la unidad:

Audio intro. "Creación de interfaces a partir de documentos XML"
https://bit.ly/2DQNcd7

/ 2. Lenguajes de descripción de interfaces basados en XML

XML (eXtensible Markup Language) es un lenguaje utilizado para estructurar, almacenar e intercambiar datos entre distintas plataformas. Al ser un metalenguaje, puede emplearse para definir otros lenguajes. Entre los más importantes, actualmente se encuentran XHTML, GML, MathML, RSS y SVG.

2.1. XHTML

XHTML (eXtensible HyperText Markup Language) es un **lenguaje derivado de XML similar a HTML**, pero con algunas diferencias que lo hacen más robusto y aconsejable para la modelación de páginas web.

Los documentos XHTML tienen que cumplir como **elementos obligatorios:**

- <! DOCTYPE>
- El atributo xmlns en <html>
- <html>, <head>, <title> y <body>

Además, la implementación de **los elementos** ha de cumplir un conjunto de características de diseño:

- Tienen que aparecer correctamente anidados.
- Deben estar cerrados.
- Siempre deben estar en minúsculas los elementos y los nombres de los atributos.
- Los valores de los atributos siempre se deben citar.
- La minimización de atributos está prohibida.

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.

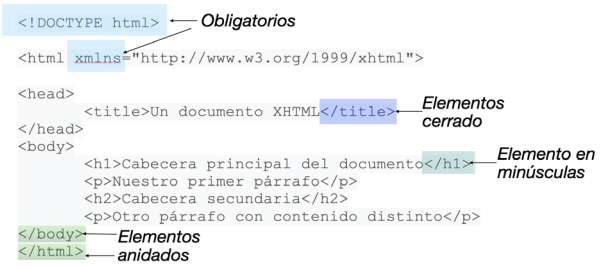

Fig. 2. Código ejemplo XHTML.

2.2. GML

GML (Geography Markup Language). Cualquier documento GML puede **recibir un formato que define el tipo de texto** que es (títulos, párrafos, listas...). Estos tipos de documentos están compuestos de marcas precedidas de doble punto(:). En el siguiente ejemplo se muestra un listado de lenguajes definidos a partir del lenguaje XML.

```
:h1.Capítulo 1.- Lenguaje XML
:p.Lenguajes de descripción de interfaces basados en XML
    :ol
    :li.GML
    :li.MathML
    :li.RSS
    :li.SVG
    :li.XHTML
    :eol.
```

Código 1. Implementación ejemplo GML.

2.3. 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. En el siguiente ejemplo se muestra la fórmula matemática a2+b2=c2 escrita en lenguaje MathML.

```
<math>
    <mrow>
        <mi>a</mi>
        <mn>2</mn>
        <mo>+</mo>
        <mi>b</mi>
        <mn>2</mn>
        <mo>=</mo>
        <mi>c</mi>
        <mn>2</mn>
    </mrow>
</math>
```

Código 2. Implementación ejemplo MathML.

2.4. RSS

El lenguaje RSS (Really Simple Syndication) tiene como objetivo principal la **difusión de información** entre los usuarios suscritos a una fuente de contenidos actualizada frecuentemente. Los programas de este tipo suelen estar compuestos por novedades del sitio web, el título, fecha de publicación o descripción. En el siguiente ejemplo se muestra una noticia publicada en lenguaje RSS.

```
<?xml version="1.0" encoding="UTF-8" ?>
<rss version="2.0">
        <channel>
                <title>Última hora</title>
                <description>Noticia importante
                </desciption>
                <link>https://elpais.com/ultimas-noticias/
                </link>
                <lastBuildDate>Mon, 06 Jan 2020 00:10:00
                </lastBuildData>
                <pubDate>Mon, 06, Jan 2020 16:20:00
+0000
                </pubDate>
        </channel>
</rss>
```

Código 3. Implementación ejemplo RSS.

2.5. XSLT

XSLT (eXtensible Stylesheet Languaje for Transformation) podríamos decir que se trata de las **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.

2.6. SVG

SVG corresponde a Scalable Vector Graphics (o Gráficos Vectoriales Escalables). Este lenguaje **permite representar elementos geométricos vectoriales, imágenes de mapa de bits y texto.** En los siguientes ejemplos se muestra la creación de diferentes elementos gráficos utilizando el lenguaje vectorial.

Ejemplo 1:

```
<!DOCTYPE html>
<html>
<body>

<svg height="100" width="100">
        <circle cx="50" cy="50" r="40" stroke="black"
        stroke-width="3" fill="cyan" />

        Lo siento, tu navegador no es compatible con SVG.
</svg>
</body>
</html>
```

Código 4. Implementación

Otras de las formas geométricas más utilizadas en SVG son el cuadrado, la elipse, la línea, la polilínea y el polígono.

Ejemplo 2:

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

Código 5. Implementación ejemplo SVG cuadrado

Ejemplo 3:

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

Código 6. Implementación ejemplo SVG elipse.

Ejemplo 4:

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

Código 7. Implementación ejemplo SVG polígono.

Vídeo 1. "Gráficos vectoriales escalables"
https://bit.ly/30gHsAH

/ 3. El documento XML. Análisis y edición

Seguro que ya has aprendido en otras asignaturas que un fichero XML es un archivo en formato texto que contiene etiquetas para identificar los elementos, así como datos que componen el documento donde se representa la información. Lo primero que hay que tener en cuenta a la hora de definir un fichero XML es que la primera línea del fichero debe ser la siguiente:

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

- **Version:** indica la versión de XML que se está empleando.
- **Encoding:** especifica el juego de caracteres con el que se ha codificado el documento.

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

```
< etiq1> ..... </ etiq1>
< etiq2> ..... </ etiq2>
```

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

Este tipo de estructura de documento con las etiquetas y los atributos implica que debemos tener mucho cuidado en la forma de almacenar la información, ya que debemos estructurar muy bien el documento y ordenar adecuadamente las informaciones.

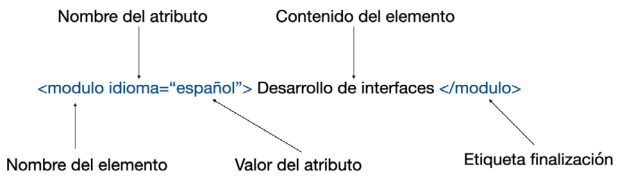

Fig. 3. Estructura de un XML

Un documento XML tiene una estructura anidada de manera jerárquica. Hay que prestar especial atención a la apertura y cierre de todas las etiquetas para que queden todas cerradas. Es habitual que los elementos tengan otros vinculados, es decir, elementos que descienden de él. En estos casos, las etiquetas de los descendientes/hijos deben cerrarse antes que la del padre.

Para lograr esto, será necesario que la estructura jerárquica se cumpla de manera estricta con respecto 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 el cierre.

3.1. Etiquetas

Las etiquetas XML son marcas que sirven para diferenciar un contenido específico del resto del contenido del documento. Una etiqueta empieza con el carácter '<", continúa un nombre identificativo, y termina con el carácter ">". El nombre de una etiqueta 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 />)

3.2. Atributos

Un atributo es un componente de las etiquetas que 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. Hay que tener en cuenta que en una misma etiqueta no pueden existir dos atributos con el mismo nombre, y todos los atributos de un elemento siempre son únicos. Por ejemplo:

```
<programadora nombre="Augusta" apellido1="Ada" apellido2="King" />
```

En este caso tenemos tres atributos únicos, nombre, apellido1 y apellido2. En cambio, en el siguiente caso, no sería correcto, dado que tenemos el atributo apellido repetido:

```
<programadora nombre="Augusta" apellido="Ada" apellido="King">
```

3.3. Valores

Tal y como se ha visto en el apartado anterior, 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="Diseño de interfaces web"
editorial="Síntesis">
        <tipo>
            <libro isbn="9788491713241" edicion="1"
paginas="210"/>
        </tipo>
        <autor nombre="Diana García-Miguel López"/>
    </texto>
</biblioteca>
```

Código 9. Ejemplo de una estructura de biblioteca XML.

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

/ 4. Eventos

Una interfaz, por sí sola, no puede saber en qué momento querrá el usuario llevar a cabo una determinada funcionalidad. Para ello, existen lo que denominamos eventos, que dan lugar a interfaces dinámicas.

Los eventos proporcionan un mecanismo adecuado para tratar las diferentes formas interacción entre el usuario y la aplicación, por ejemplo, cuando el usuario presiona una tecla o pulsa con el puntero del ratón. Ante la llegada de un evento, en algunas ocasiones nos interesará tratarlo. Por ejemplo, la pulsación de un número en una aplicación que presenta la funcionalidad de una calculadora. Otras veces no será necesario el tratamiento del evento con ninguna acción, por ejemplo, cuando el usuario presiona con el ratón sobre un texto al que no hemos asignado ninguna información complementaria.

Algunos de los eventos más comunes que se pueden producir en una aplicación, como interacción con las interfaces gráficas son:

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

De igual modo, si queremos que un texto se ponga de color rojo al situarnos encima, y de color gris al salir, existen otros eventos que podemos utilizar como **MouseEntered y MouseExited;** el primer evento se encargará de poner el texto de color rojo y, el segundo, de ponerlo de color gris.

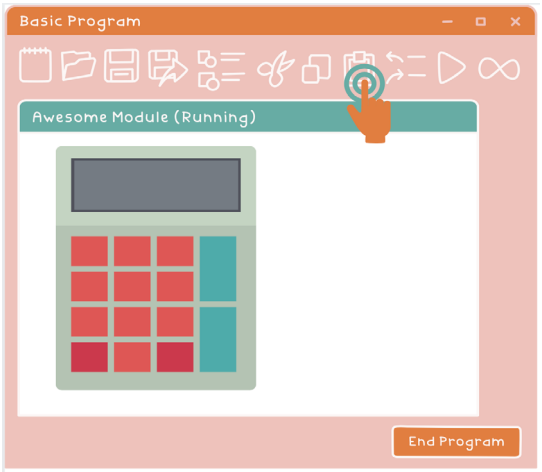

Fig. 4. Interfaz que permite interacciones utilizando eventos.

/ 5. Herramientas para la creación de interfaces de usuario multiplataforma

Los editores de XML que existen actualmente nos permiten escribir código de una forma rápida y eficiente. Podemos utilizar desde un simple editor de texto hasta las últimas versiones de editores que contienen funciones que analizaremos en este apartado.

Una de las características fundamentales que presentan los editores de XML son las facilidades de implementación que ofrecen para escribir código resaltando la sintaxis, insertando elementos y estructuras de XML de uso común a través de la función de autocompletado.

	Atom	Notepad++	Dreamweaver	Visual Studio
Multinivel	✓	✓	✓	✓
Paneles	✓	✓	✓	✓
Control versiones	✓			✓
Libre	✓			✓
WYSIWYG	✓	✓		✓

Tabla 1. Tabla comparativa de los editores.

5.1. Notepad ++

Editor de textos de propósito general que reconoce la sintaxis de múltiples lenguajes de programación. Es gratuito, está disponible para Linux y Windows y su código fuente se puede descargar.

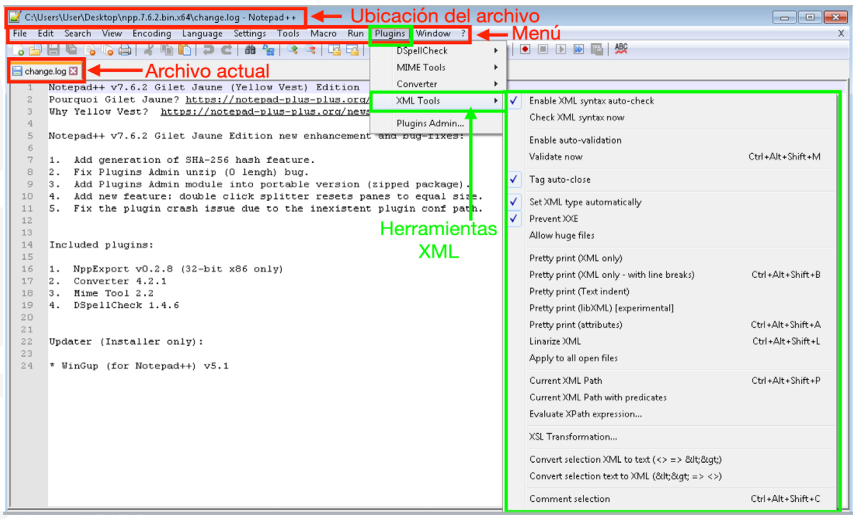

Fig. 5. Interfaz Notepad++.

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.

5.2. Atom

El editor Atom es una **herramienta multinivel** que sirve tanto para los usuarios que comienzan a programar, como para uso profesional. 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.

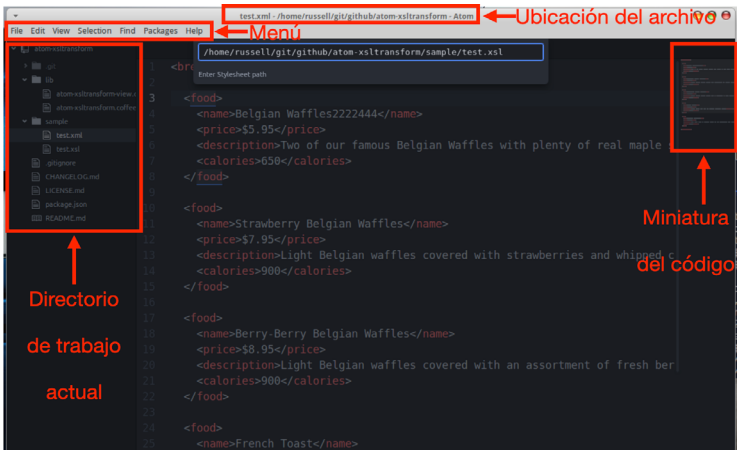

Fig. 6. Interfaz de Atom.

5.2.1. Teletype

Atom tiene una manera de colaborar en tiempo real mediante la herramienta 'Teletype', la cual **permite que muchos desarrolladores puedan editar un archivo a la vez**, en tiempo real. Son herramientas muy útiles cuando un desarrollador tiene que trabajar de forma colaborativa con otras personas.

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.

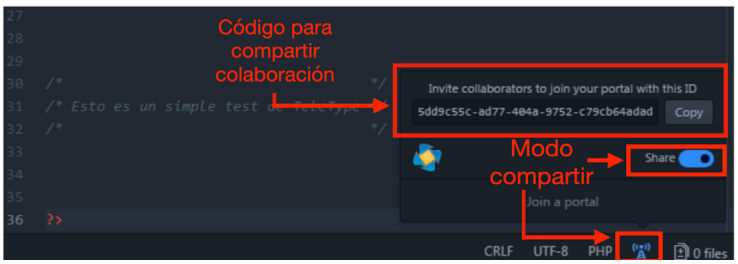

Fig. 7. Interfaz herramienta Teletype de Atom.

5.2.2. Git y GitHub

Todavía en Atom, queremos destacar una de las mayores ventajas de este editor de código, como son las herramientas **Git y GitHub**, que **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.

5.3. Adobe Dreamweaver CC

Es uno de los editores que admite tanto el método textual como el WYSIWYG, del inglés What You See Is What You Get (lo que ves es lo que obtienes). Esta es una frase aplicada a los editores de código que permiten escribir un documento mostrando directamente el resultado final. Por lo tanto, tú eliges si quieres programar con una presentación visual en vivo o seguir el camino clásico.

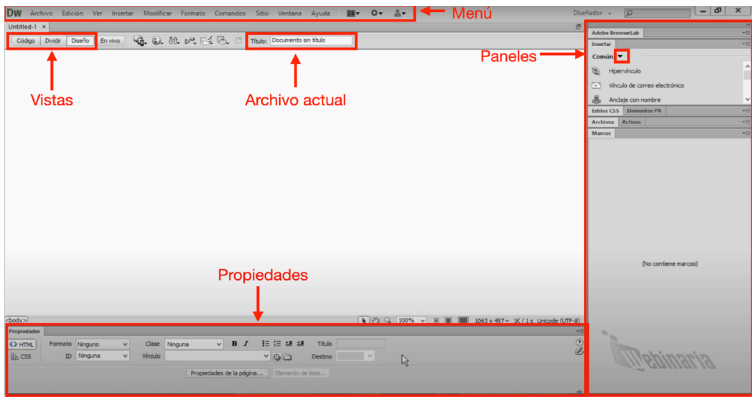

Fig. 8. Interfaz Dreamweaver.

Dreamweaver permite escribir código en todos los lenguajes de programación importantes. Totalmente integrado en el ecosistema de software de Adobe, está disponible para Windows y OS X. Dispone de vista previa para que los desarrolladores puedan programar mientras previsualizan el producto final. Además, permite confirmar el código y accesibilidad de la página, característica que puede facilitarles seguir las pautas de accesibilidad de contenido web.

5.4. Visual Studio Code

Es uno de los editores de código fuente más utilizados. Es compatible con varios lenguajes de programación y está disponible para Windows, Linux y macOS.

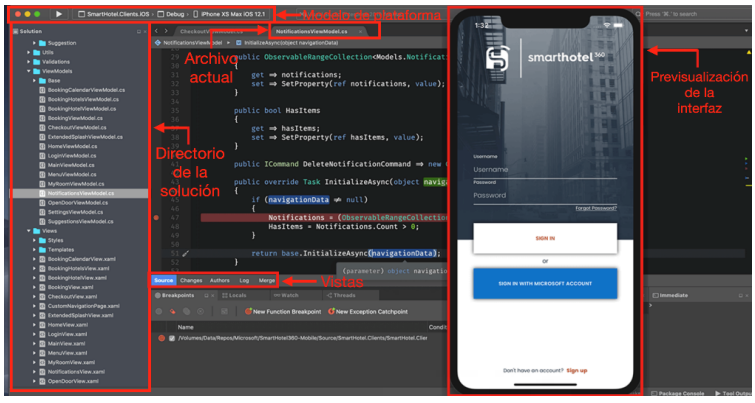

Fig. 9. Interfaz Visual Studio Code.

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 dispone de una versión de pago más completa). 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.

Vídeo 2. "Crear una interfaz gráfica desde un archivo XML en Eclipse"
https://bit.ly/3hh49vJ

/ 6. Generación de código para diferentes plataformas

En la actualidad, ahora que existen cada vez más tipos de sistemas, uno de los problemas más relevantes a los que se debe dar respuesta es al intercambio de datos entre sistemas incompatibles. El intercambio de datos con XML reduce en gran medida la dificultad de este problema, dado 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.

Gracias a su portabilidad, XML se ha convertido en 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.

Audio 1. "Generación de código para diferentes plataformas"

https://bit.ly/32nuALR

/ 7. Caso práctico 1: "Generación de un logo con gráficos vectoriales escalables"

Planteamiento: La empresa radiofónica FourWaves nos ha solicitado el diseño de un logo que represente su marca. Esta cadena se caracteriza por sus programas de naturaleza, mindfulness, corazón y economía. Para ello, nos indican que tiene que ser claramente representativo, sencillo y con alta calidad para poder ser reescalable a cualquier tamaño que necesiten en un futuro. El logo para su interfaz va tener cuatro componentes visuales claramente diferenciados:

- **Una onda verde** (que representará el contenido de naturaleza).
- **Una segunda onda azul** (que simbolizará la relajación del ámbito del mindfulness).
- **Una tercera onda rosa** (que servirá para representar los programas de corazón).
- **Una última onda amarilla** (correspondiente al color de las monedas de los programas de economía).

Nudo: En primer lugar, como se ha visto en los apartados anteriores, el lenguaje SVG permite representar elementos geométricos vectoriales e imágenes de mapa de bits. En este caso, se puede realizar el diseño de 4 elipses que representarán las cuatro ondas del nombre de la cadena y las cuatro secciones que trabajan. Se propone realizar un sencillo código SVG con cuatro componentes de tipo elipse y modificar sus radios y estilo de color para conseguir el logo que desea la empresa.

Desenlace: En base a las especificaciones de la empresa, una propuesta de diseño del logo sería el siguiente: la base sería una elipse verde, superpuesta a ella una elipse azul con menores radios en el eje x e y, y después, de manera sucesiva, se representan las elipses rosa y amarilla. Por último, se les envía un diseño con los componentes anteriores y queda diseño de interfaz como el que se muestra.

```
<!DOCTYPE html>
<html>
 <body>
  <svg height="150" width="500">
   <ellipse cx="240" cy="100" rx="220" ry="30" style="fill:lime" />
   <ellipse cx="220" cy="70" rx="190" ry="20" style="fill:cyan" />
   <ellipse cx="210" cy="45" rx="170" ry="15" style="fill:pink" />
   <ellipse cx="205" cy="25" rx="140" ry="12" style="fill:yellow" />
  </svg>
 </body>
</html>
```

Código 8. Código solución «Caso práctico 1».

Fig. 10. Logo resultado código implementación 8.

/ 8. Caso práctico 2: "Estructura documento XML"

Planteamiento: Para nuestra nueva interfaz de reserva de viajes TravelWithMe necesitamos tener almacenada información sobre las siete maravillas del mundo. Desarrolla un documento XML que almacene la información (nombre, país) sobre cada una de las 7 maravillas del mundo.

Nudo: El código implementado para esta propuesta tendría la siguiente estructura:

MARAVILLAS DEL MUNDO MODERNO	
Nombre	**País**
La Gran Muralla China	China
El Coliseo Romano	Italia
Chichén Itzá	México
Machu Picchu	Perú
Palacio de Taj Mahal	India
Cristo Redentor	Brasil
Petra	Jordania

Tabla 2. Tabla de datos de las maravillas del mundo moderno.

```
<?xml version="1.0" encoding="UTF-8"?>
<maravillas>
  <maravilla>
    <nombre>Muralla China</nombre>
    <pais>China</pais>
  </maravilla>
<maravilla>
    <nombre>El Coliseo Romano</nombre>
    <pais>Italia</pais>
  </maravilla>
<maravilla>
    <nombre>Chinchén Itzá</nombre>
    <pais>México</pais>
  </maravilla>
<maravilla>
    <nombre>Machu Picchu</nombre>
    <pais>Perú</pais>
  </maravilla>
<maravilla>
    <nombre>Palacio Taj Mahal</nombre>
    <pais>India</pais>
  </maravilla>
<maravilla>
    <nombre>Cristo Redentor</nombre>
    <pais>Brasil</pais>
  </maravilla>
<maravilla>
    <nombre>Petra</nombre>
    <pais>Jordania</pais>
</maravilla>
</maravillas>
```

Código 10. Datos implementados en XML.

Desenlace: La información quedará estructurada en elementos de tipo 'maravilla' con subelementos que contendrán el nombre y el país.

/ 9. Resumen y resolución del caso práctico de la unidad

A lo largo de este tema hemos visto que **XML** es un lenguaje utilizado para estructurar, almacenar e intercambiar datos entre distintas plataformas.

Es importante recordar que, para estructurar correctamente un documento XML, existen **reglas** en cuanto a las etiquetas, atributos y valores que debemos seguir. También hemos resaltado el uso de diferentes tipos de **eventos** como mecanismos necesarios para interactuar con la interfaz.

Una vez hemos tenido claros esos conceptos, hemos profundizado mediante ejemplos prácticos en los **principales lenguajes basados en XML: XHTML, GML, MathML, RSS, XSLT y SVG.** Por último, se ha realizado una **comparativa de los mejores editores** que existen en la actualidad para crear interfaces basadas en XML: Atom, Notepad++, Dreamweaver y Visual Studio Code.

Resolución del caso práctico de la unidad.

Basándonos en lo visto en este tema, el diseño que necesita la empresa radiofónica puede basarse en las áreas de sus contenidos (economía, corazón, mindfulness y naturaleza) y utilizar el código de amarillo, rosa, azul y verde para identificar cada una de ellas, respectivamente.

Como lenguaje de desarrollo basado en XML, para este proyecto se puede utilizar XHTML para la parte de la modelación de la interfaz que se basa en contenido generado en su página web. Para la parte de suscripciones al podcast, podemos utilizar el lenguaje RSS, dado que nos permite difundir información a usuarios que se han suscrito a una fuente de contenidos actualizada frecuentemente. Por último, se propone emplear SVG para realizar el diseño vectorial de los elementos de la interfaz, tal y como se muestra en el primer caso práctico, para que puedan ser reescalables y adaptables a cualquier plataforma.

Utilizando cualquiera de los editores descritos en este tema se puede realizar el desarrollo de una interfaz a partir de un documento XML. En concreto, si queremos realizar el desarrollo de manera que podamos ver el diseño gráfico mientras programamos, es decir, de manera WYSIWYG, sería recomendable utilizar los editores VisualStudio o Dreamweaver.

/ 10. Bibliografía

Fernández, A., García-Miguel, B. y García-Miguel, D. (2020). Desarrollo de Interfaces. (1.a ed.). Madrid: Síntesis.