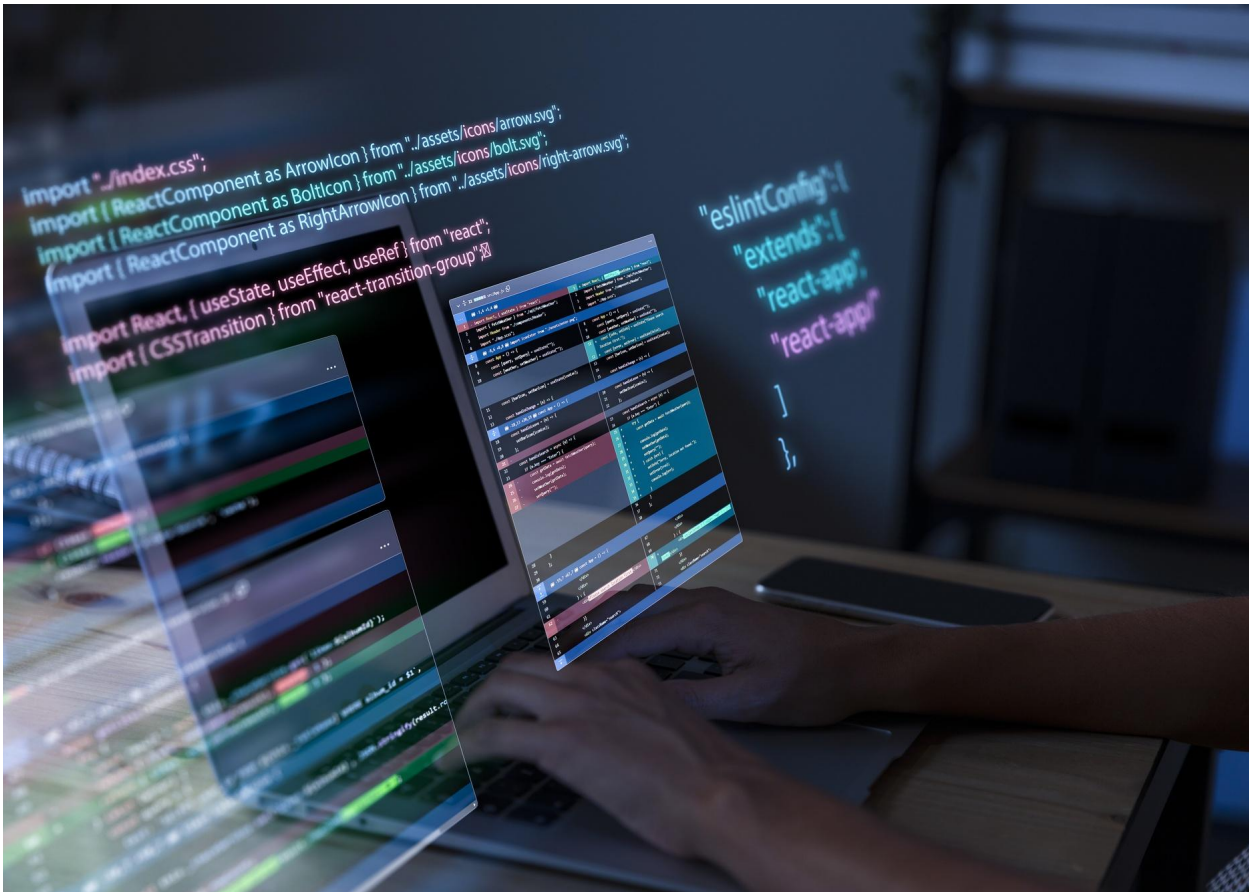
DAM 1 DAW 1 / Lenguajes de Marcas y sistemas de gestión de información

XML, DTD y Schemas

3.1 XML



Autor: Lisa ERIKSEN

Fecha: 2025 / 2026

1. ¿Qué es XML y por qué es útil?

XML (Extensible Markup Language) es un lenguaje de marcado diseñado para almacenar, transportar y estructurar información de una forma legible tanto para humanos como para máquinas. A diferencia de HTML, XML no está diseñado para mostrar datos, sino para estructurarlos.

- HTML = un lenguaje visual → visualización, diseño, colores, etc.
- XML = un lenguaje de datos → estructura, organización, sin visualización estilizada.

1.1. Descubrimos XML

Actividad: Una **plataforma escolar** exporta los datos de alumnos en XML. Tu objetivo es reproducir un archivo simple.

1. Crea una nueva carpeta XML
2. Ábrela con VS Code y crea un nuevo archivo **alumno.xml**
3. Copia este código y ábrelo en tu navegador.

```
<?xml version="1.0" encoding="UTF-8"?>
<alumno>
  <nombre>Laura</nombre>
  <curso>Lenguajes de Marcas</curso>
</alumno>
```

Prueba:

- ¿El navegador lo muestra?
- ¿Qué ves exactamente? ¿Colores? ¿Tablas? ¿o texto plano estructurado?
- ¿Por qué crees que XML sirve para guardar datos?

2. Sintaxis de XML

2.1. Prolog

Cuando se envían archivos XML a una institución o administración publica, es necesario que incluyan un prólogo correcto para que puedan ser validados.

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

Prueba:

1. Crea un archivo llamado **prolog.xml**
2. Escribe el código de arriba y ábrelo en el navegador.
3. Suprime el prolog *<?xml version="1.0" encoding="UTF-8"?>* y reintenta.
4. ¿Es obligatorio el prolog? ¿Qué pasa si lo quitas?

2.2. Elementos

Actividad: Una biblioteca sincroniza usuarios en XML.

1. Crea un archivo **usuario.xml**
2. Completa este código para que funcione sin error:

```
<usuario>
 <nombre>Laura</nombre>
 <edad>22</edad>
```

Prueba:

- ¿Se ve correctamente en el navegador?
- Crea otro usuario con los mismos elementos (nombre y edad).
- ¿Qué ocurre si repites dos veces el mismo elemento?

2.3. Comentarios

Los archivos de configuración usan comentarios para avisar a otros técnicos.

Actividad:

1. Añade un comentario **arriba del archivo usuario.xml**, luego prueba:

```
<!-- La edad es opcional -->
<alumno>
  <nombre>Laura</nombre>
  <curso>Lenguajes de Marcas</curso>
</alumno>
```

Pregunta:

- ¿El comentario aparece en el navegador?

2.4. CDATA

En los textos editoriales o en datos que contienen caracteres especiales (<, >, &, etc.), XML puede dar errores si se escriben directamente. CDATA permite incluir estos caracteres **sin que XML los interprete como código**.

Actividad:

1. Crea **texto.xml** y escribe el código siguiente

```
<texto>5 < 10 & 3 > 2</texto>
```

2. Abre el archivo en el navegador y observa.
3. Luego repite con CDATA y vuelve a abrir el archivo.

```
<texto><![CDATA[5 < 10 & 3 > 2]]></texto>
```

Pregunta:

- ¿Qué diferencia has visto?
- ¿Por qué crees que CDATA es útil al trabajar con textos que contienen caracteres especiales?

3. Estructura básica

Las editoriales a menudo envían información sobre libros en formato XML. Esto permite que diferentes aplicaciones (sitios web, bibliotecas digitales, sistemas de venta) lean y procesen los datos de manera automática y estructurada.

Actividad:

1. Crea un archivo llamado **libro.xml.**
2. Copia el código siguiente
3. Abre el archivo en un navegador o editor de XML y observa cómo se muestra.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<libro>
  <titulo>Programación en XML</titulo>
  <autor>Maria Pérez</autor>
  <capitulos>
    <capitulo numero="1">Introducción</capitulo>
    <capitulo numero="2">Sintaxis</capitulo>
  </capitulos>
</libro>
```

Preguntas:

- ¿Cuál es el nodo raíz de este archivo XML?
- ¿Qué crees que ocurriría si borras el cierre de un <capitulo> (por ejemplo, </capitulo>)?
- ¿Por qué es importante que todos los elementos tengan su cierre en XML?

4. Validación de XML

4.1. DTD (Document Type Definition)

Un DTD define la estructura del documento y las reglas que debe seguir.

Aquí tienes un ejemplo de DTD:

```
<!ELEMENT libro (titulo, autor, capitulos)>
<!ELEMENT titulo (#PCDATA)>
<!ELEMENT autor (#PCDATA)>
<!ELEMENT capitulos (capitulo+)>
<!ELEMENT capitulo (#PCDATA)>
<!ATTLIST capitulo numero CDATA #REQUIRED>
```

Actividad: Una editorial usa archivos XML para describir libros. Quiere asegurarse de que cada libro tenga al menos un capítulo. Para eso necesita un DTD que obligue esa regla.

1. Crea **libro.dtd** y escribe el código de arriba.

2. En tu **libro.xml**, enlázalo añadiendo esta línea arriba de la pagina:

```
<!DOCTYPE libro SYSTEM "libro.dtd">
```

3. Quita todos los elementos <capitulo> a dentro de <capítulos> y ábrelo

Preguntas:

- ¿Por qué da error cuando no hay capítulos?
- ¿Qué significa el "+"?
- ¿Cómo cambiarías el DTD si quisieras permitir 0 capítulos? (Pista: intenta * y ?)
- ¿En general, que pasa si un elemento del DTD no esta presente en el archivo xml?

4.2. XML Schema (XSD)

Es una alternativa más potente y moderna al DTD, basada en XML: Las administraciones públicas usan **XSD** para asegurarse de que los formularios online **tienen todos los datos correctos**, por ejemplo: número de documento, fecha, código postal, etc.

Actividad:

1. Crea **formulario.xsd** y copia el código del schema XSD siguiente:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

 <xsd:element name="capitulo">
  <xsd:complexType>
   <xsd:simpleContent>
    <xsd:extension base="xsd:string">
     <xsd:attribute name="numero" type="xsd:integer" use="required"/>
    </xsd:extension>
   </xsd:simpleContent>
  </xsd:complexType>
 </xsd:element>

</xsd:schema>
```

1. Observa el XML proporcionado y determina
   - Que elemento esta protegido por este schema ?
   - cuál atributo del elemento es requerido.

2. Crea **capitulo.xml** con el código siguiente:

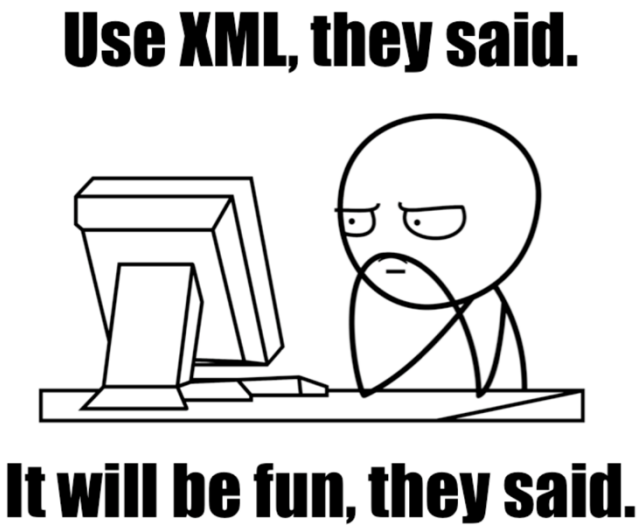
```
<?xml version="1.0" encoding="UTF-8"?>
<capitulo numero="1">Introducción</capitulo>
```

3. Valida tu código con la herramienta en línea: https://www.xmlvalidation.com/
   a. Pega el codigo de **capitulo.xml** y marca validate against external XML schema
   b. Haz clic en **validate** y pega el código de **formulario.xsd**
   c. Haz clic en **continue validation** y observa si hay errores

4. Ahora, borra el atributo requerido de tu código y vuelve a validar tu código.

Preguntas:

- ¿Dónde se declara que el atributo “numero” es obligatorio?
- ¿Qué pasa si lo quitan y validan?



5. Practicamos con ejemplos reales

5.1. Gestión de productos en un e-commerce

Los sitios web y tiendas en línea envían catálogos de productos en formato XML para que otras aplicaciones puedan leerlos fácilmente. Cada producto tiene un nombre, un precio y una categoría.

Actividad: Vas a crear un archivo productos.xml que contenga al menos tres productos y validar que la estructura es correcta.

1. Crea un archivo llamado **productos.xml**.
2. Copia la siguiente estructura inicial:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<productos>
  <producto id="101">
    <nombre>Televisor</nombre>
    <precio>450</precio>
    <categoria>Electrónica</categoria>
  </producto>
  <producto id="102">
    <nombre>Lavadora</nombre>
    <precio>300</precio>
    <categoria>Electrodomésticos</categoria>
  </producto>
</productos>
```

3. Añade un **tercer producto** con su propio id, nombre, precio y categoría.
4. Guarda el archivo y abre el XML en un navegador.

Preguntas:

- ¿Se muestra correctamente tu archivo XML?
- ¿Todos los productos tienen la misma estructura?
- ¿Qué pasaría si olvidaras algún elemento obligatorio (por ejemplo <nombre>)?

5.2. Gestión de productos en un Sistema

Muchas herramientas y sistemas usan archivos XML para almacenar configuraciones, por ejemplo:

- **Maven**: para proyectos Java
- **Tomcat**: para servidores web
- **Jenkins**: para automatización de tareas
- **Android**: para configuraciones de aplicaciones

Los archivos XML permiten definir parámetros de manera estructurada y fácil de leer para humanos y máquinas.

Actividad: Observa el siguiente ejemplo de archivo de configuración y analiza cada nodo y su función.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<configuracion>
  <baseDeDatos>
    <host>localhost</host>
    <usuario>admin</usuario>
    <password>1234</password>
  </baseDeDatos>
</configuracion>
```

Preguntas:

1. ¿Cuál de los nodos contiene información sensible y por qué?
2. ¿Por qué XML es adecuado para almacenar configuraciones de sistemas?
3. (Opcional) ¿Qué podrías cambiar en este archivo para que sea más seguro sin dejar de usar XML?

6. XML vs JSON

Característica	XML	JSON
Formato	Basado en etiquetas.	Basado en pares clave-valor.
Legibilidad	Más detallado y menos compacto.	Más legible y compacto.
Validación	DTD o XSD.	JSON Schema.
Uso	Sistemas complejos, interoperabilidad entre sistemas. ***Ejemplo:*** *Un proveedor envía un catálogo en XML; El backend lo convierte a JSON para el frontend.*	APIs, aplicaciones web y móviles. ***Ejemplo:*** *El backend devuelve productos en JSON que el frontend muestra en la tienda online.*