

XML

EXTENSIBLE MARKUP LANGUAGE - LENGUAJE DE MARCAS EXTENSIBLE.

- **Historia.**
- **Concepto:**
 - Lenguaje de marcado desarrollado por el consorcio W3C (World Wide Web Consortium).
 - Define un conjunto de reglas para la codificación de documentos que tanto personas, como máquinas pueden leer.
 - Para ello, utiliza etiquetas que definen la estructura del documento y también como debe ser éste transportado y almacenado.
- **Versiones:**
 - 1.0.
 - 1.1.
- **Función:**
 - Compartir o intercambiar datos entre sistemas.
- **Archivo XML.**
 - Archivos guardados con la extensión .xml, consistentes en archivos de texto sin formato que utilizan un conjunto de etiquetas personalizadas que agrupan los datos y que describen, tanto la estructura, como otras características del documento.
 - Un archivo XML incluye tanto etiquetas como datos, proporcionando las primeras, estructura a los segundos.
- **Diferencia con HTML.**
 - XML es extensible, es decir, no incorpora un lenguaje de marcado predefinido, sino que permite al usuario crear sus propias etiquetas para describir contenidos.
 - Dichas etiquetas son autodefinidas e ilimitadas.
- **Otros lenguajes basados en XML:**
 - XHTML, SVG, XVL, RSS, RDF, SMIL, ...
 - Lenguajes creados por el usuario.
- **Tecnologías XML.**
 - Servicios Web.
 - **SOAP.**
 - **WS Policy.**
 - **WSDL.**
 - **XML Signature.**
 - Especifica sintaxis y reglas para la creación de firmas digitales en documentos XML.
 - **XML Encryption.**
 - Especifica sintaxis y reglas para encriptar documentos XML.
 - Web semántica.
 - **RDF.**
 - **RDFa.**
 - **RDF-S.**

- **OWL.**
 - **RIF.**
 - **SPARQL.**
- Localización y Recorrido.
 - **XPath.**
 - Lenguaje de acceso que permite recorrer y procesar un documento XML.
- Consultas.
 - **XQuery.**
 - Lenguaje de consulta que permite acceder y manipular documentos XML, u otro tipo de contenidos que se puedan representar en formato XML (bases de datos relacionales, documentos ofimáticos, etc.).
- Transformación.
 - **XSLT.**
 - Lenguaje de transformación de documentos XML a otros formatos estén o no, basados XML.
 - **XSL Formatting Objects (XSL-FO).**
 - Permite formatear documentos XML para crear otros tipos de archivos como PDFs.
 - **DOM.**
 - **SAX.**
- Modelización y validación de datos.
 - **DTD.**
 - **XML Schema.**
 - Lenguaje que permite describir la estructura y las restricciones del contenido XML.
- **Características:**
 - Independencia Datos-Formato.
 - Simplicidad en intercambio de datos.
 - Simplicidad al cambiar de plataforma.
 - Formato texto plano.
 - Extensibilidad.
 - Interoperabilidad.
 - Compatibilidad con todo tipo de dispositivos.
 - Accesible para todo tipo de alfabetos y lenguajes.
 - Representación jerárquica y anidada.
- **Programas para la creación de un documento XML.**
 - Bloc de notas de Windows u otros sistemas operativos.
 - Editores de código (Notepad++, XML Copy Editor, VCode, etc.).
 - Hojas de cálculo como MS Excel.
 - IDE's de programación.

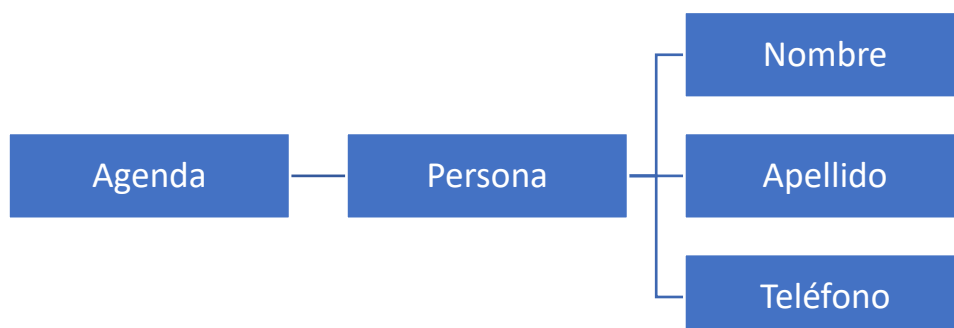
TERMINOLOGÍA.

- **Documento XML.**
 - Documento de texto plano.
- **Procesador XML.**
 - Analizador o Parser que lee un documento, analiza el contenido y pasa la información obtenida en un formato estructurado a la aplicación que la solicitó.
- **Caracteres.**
 - Se refiere a los juegos de caracteres en los que está codificado un documento XML (UTF-8, ISO-8859-1, etc.).
- **Marcas.**
 - Etiquetas.
 - Referencias a entidades.
 - Atributos.
- **Contenido.**
 - Todo lo que va encerrado entre etiquetas de apertura y cierre.
 - Un elemento puede tener como contenido otros elementos (elementos hijos).
- **Elementos.**
 - Componente lógico de un documento XML como, por ejemplo:
 - El contenido con su correspondiente etiqueta de apertura y cierre.
 - Una única etiqueta vacía.
- **Etiquetas.**
 - Identificador encerrado entre < y >.
 - Tipos:
 - **Apertura.**
 - <Etiqueta>.
 - Incluye su par de cierre.
 - **Cierre.**
 - </Etiqueta>.
 - Incluye su par de apertura.
 - **Vacías.**
 - <Etiqueta/>.
 - Ejemplos:
 - <cliente>...</cliente>.
 - <cliente/>.
 - Reglas para etiquetas:
 - XML distingue mayúsculas y minúsculas, siendo las minúsculas la forma común de escribir las etiquetas.
 - <nombre></Nombre> no son lo mismo.
 - Sólo puede haber un elemento raíz.
 - Las etiquetas pueden empezar por letra, dos puntos (:) y guion bajo (_), pero no por números, aunque si contenerlos:
 - <cliente> --- Correcto.
 - <_direccion1> --- Correcto.
 - <1precio> --- Incorrecto.
- **Atributos.**
 - Componente de las etiquetas que consta del par nombre - valor.

- Se puede incluir en las etiquetas de apertura y vacías, pero no en las de cierre.
- No puede repetirse su nombre en una misma etiqueta.
- Sintaxis:
 - Nombre del atributo = “Valor”.
- Ejemplo:
 - País = “España”.
- **Comentarios.**
 - Se especifican igual que en HTML, encerrados entre <!-- Comentario -->.
- **Declaración XML.**
 - Contenido encerrado entre los caracteres <? xml y ?>
 - Proporciona información sobre el propio documento.
 - Se incluye al principio del documento.
 - Atributos:
 - **version**:
 - Sirve para especificar la versión XML que se está empleando.
 - Sólo hay dos: 1.0 y 1.1, cuyas diferencias no significativas.
 - **encoding**:
 - Permite especificar el juego de caracteres con el que se ha codificado el documento XML.
 - El juego de caracteres se escribe en mayúsculas.
 - Es opcional, por defecto UTF-8.
 - **standalone**:
 - Atributo que incluye 2 valores: “yes” o “no”.
 - Especifica si el documento XML debe validarse con un archivo DTD externo (“no”) o un DTD interno (“yes”).
 - Sintaxis:
 - <?xml version=”versión” encoding=”JUEGO DE CARACTERES standalone = “yes | no”?>
 - Ejemplo:
 - <?xml version=”1.0” encoding=”UTF-8 standalone = ”no”?>

Estructura de anidamiento en árbol.

- Muestra los anidamientos entre etiquetas.
- Muestra cuales son las etiquetas hijas dentro de un anidamiento y cuál es la raíz.
- Ejemplo:
 - Agenda.
 - Persona.
 - Nombre.
 - Apellido.
 - Teléfono.



ESTRUCTURA DE UN ARCHIVO XML.

- **Prólogo.**
 - Opcional.
 - Equivale a la cabecera HTML.
 - **Contenido:**
 - Declaración XML.
 - DTD - Declaración de tipo de documento.
 - Comentarios.
 - Instrucciones de procesamiento.
- **Cuerpo:**
 - Obligatorio.
 - Contiene los datos o información junto con el marcado.
 - **Contenido:**
 - **Elementos o Etiquetas.**
 - Las hay de apertura y cierre. <>... </>
 - Pueden contener datos o estar vacíos.
 - **Atributos:**
 - Propiedades o características a incluir en las etiquetas que proporcionan más información acerca de éstas.
 - **Sintaxis:**
 - <etiqueta atributo1 = "valor" atributo2 = "valor" ...>Dato </etiqueta>
 - **Ejemplo:**
 - <nombre alias =" Pepito">Pepe</nombre>
 - **Ejemplo de un mismo contenido XML usado etiquetas o atributos:**
 - **Con etiquetas:**

```
<datos_cliente>
  <cliente>
    <nombre>Luis</nombre>
    <apellidos>Sanz García</apellidos>
  </cliente>
  <cliente>
    <nombre>Eva</nombre>
    <apellidos>Salgado López</apellidos>
  </cliente>
</datos_cliente>
```

- Con atributos:

```
<datos_cliente>
    <cliente nombre = "Luis" apellidos =" Sanz García" />
    <cliente nombre = "Eva" apellidos =" Salgado López" />
</datos_cliente>
```

DOCUMENTO XML BIEN FORMADO.

- Un documento está bien formado cuando respeta las reglas básicas de XML, si no, no es un documento XML.
- Reglas:
 - Contiene únicamente caracteres Unicode válidos.
 - Hay un solo elemento raíz que contiene al resto.
 - Los nombres de los elementos y de sus atributos no contienen espacios.
 - El primer carácter de un nombre de elemento o de atributo puede ser una letra, dos puntos (:) o subrayado (_).
 - El resto de caracteres pueden ser también números, guiones (-) o puntos (.).
 - Se pueden usar tildes en las etiquetas si el juego de caracteres lo permite.
 - Los caracteres "<" y "&" sólo se utilizan como comienzo de marcas, si se quieren utilizar como contenido hay que usar caracteres de escape o entidades de carácter.
 - **Entidades de carácter o internas:**

Referencia a entidad	Carácter
<	<
>	>
&	&
'	'
"	"

- Las etiquetas no pueden contener caracteres especiales, no pueden empezar por caracteres especiales, no pueden empezar por xml o XML y no pueden contener espacios en blanco.
 - Las etiquetas de apertura, de cierre y vacías están correctamente anidadas.
 - No falta ni sobra ninguna etiqueta de apertura o cierre.
 - Las etiquetas de cierre coinciden con las de apertura, incluso en mayúsculas-minúsculas.
 - Las etiquetas de cierre no contienen atributos.
 - No puede haber dos atributos con el mismo nombre en la misma etiqueta.
 - Los atributos deben tener un valor.
 - Los valores de los atributos están entre comillas simples o dobles.
 - No existen referencias en los valores de los atributos.
- Ejemplos Incorrectos:

```
<error>
    <nombre>
```

```

    Luis <Gómez> Sánchez. (El apellido no puede escribirse entre <> si no
    es una etiqueta. Correcto sería &lt;Gómez&gt;)
  </nombre>
</error>
<error>
  <precio divisa = Euros>200,00</precio> (Faltan comillas en el valor el atributo)
</error>
<error>
  <nombre#>
    Luis (Carácter # no permitido en el nombre de una etiqueta)
  </nombre#>
</error>
<error>
  <nombre>
    Luis (No distinción mayúsculas-minúsculas en etiqueta)
  </NOMBRE>
</error>
<error>
  <nombre>Luis </nombre>
  <teléfono>123456789 (Falta etiqueta de cierre <teléfono>)
</error>
<error>
  <nombre>Luis
  <teléfono>123456789</teléfono>
  </nombre> (Etiqueta mal anidada.)
</error>

```

DOCUMENTO XML VÁLIDO.

- Un documento XML puede estar bien formado, pero no ser válido.
- Para que sea válido de incluir lo siguiente:
 - Una referencia a una gramática.
 - Incluir sólo elementos y atributos definidos en esa gramática.
 - Cumplir las reglas gramaticales especificadas en la gramática.
- Existen varias formas de definir la gramática para documentos XML, las más utilizadas son:
 - DTD (Document Type Definition - Definición de Tipo de Documento).
 - XML Schema.

DTD.

Concepto y características.

- Documento que define la estructura de un documento XML.
- Se guarda en un archivo de texto con extensión *dtd*.
- Permite especificar las reglas sobre lo que se va, o no, a permitir a un documento XML.
- Indica qué elementos, atributos, entidades, notaciones, etc., pueden aparecer, así como su orden y número de veces en que pueden aparecer, cuáles pueden ser hijos de cuáles, etc.

- El procesador XML usa el documento DTD para comprobar si un documento XML es válido, es decir, si cumple las reglas especificadas en el DTD.
- Es el modelo antiguo, heredado del lenguaje SGML.
- El DTD describe:
 - **Elementos:**
 - Qué etiquetas están permitidas y cuál es su contenido.
 - **Estructura:**
 - Cuál es el orden de las etiquetas.
 - **Anidamiento:**
 - Qué etiquetas van incluidas dentro de otras etiquetas.

Ubicación.

- **Interna.**
 - El DTD está incluido en el documento XML.
 - Sintaxis:
 - <!DOCTYPE nombre [declaraciones]>
 - "nombre" debe coincidir con el nombre del elemento raíz del documento XML.
- **Externa.**
 - El DTD está incluido en un archivo independiente con la extensión .DTD.
 - En el archivo externo DTD no se incluye la sentencia <!DOCTYPE nombre...>
 - Sintaxis:
 - **En el propio equipo :**
 - <!DOCTYPE nombre SYSTEM “Ruta de Acceso/archivo.dtd”>
 - La ruta de acceso hasta el archivo puede ser absoluta o relativa.
 - Ejemplo:
 - <!DOCTYPE ventas SYSTEM "datos.dtd">
 - **En un equipo remoto con DTD publica:**
 - <!DOCTYPE nombre PUBLIC “URL/Ruta de Acceso/archivo.dtd”>
 - Ejemplo:
 - <!DOCTYPE ventas PUBLIC
 "https://www.tienda.es/archivos/datos.dtd">
- **Híbrida.**
 - Combinación interna-externa.
 - Sintaxis:
 - <!DOCTYPE nombre SYSTEM “Ruta de Acceso /archivo.dtd” [declaraciones]>

Declaraciones.

- **ELEMENTOS.**
 - Los elementos del documento se especifican con la declaración ELEMENT.
 - Sintaxis:
 - <!ELEMENT Nombre del elemento (Contenido)>
 - Contenido:
 - Expresión que define el contenido del elemento.
 - **Contenidos posibles:**

- **EMPTY.**
 - El elemento debe estar vacío, no debe tener ningún contenido.
 - Sintaxis:
 - <!ELEMENT Nombre del elemento EMPTY>
 - Ejemplos:
 - DTD.
 - <!DOCTYPE agenda [<!ELEMENT teléfono EMPTY>]>
 - XML.
 - <teléfono> 345678922</teléfono> // Error.
 - <teléfono></teléfono> // Correcto.
 - <teléfono/> // Correcto.
- **(#PCDATA).**
 - El elemento puede contener texto.
 - Sintaxis:
 - <!ELEMENT Nombre del elemento (#PCDATA)>
 - Ejemplos:
 - DTD.
 - <!DOCTYPE agenda [<!ELEMENT nombre (#PCDATA)>]>
 - XML.
 - <nombre> Virginia</nombre> // Correcto.
 - <nombre> <teléfono></teléfono> </nombre> // Error.
- **ANY.**
 - El elemento puede contener texto u otros elementos.
 - Sintaxis:
 - <!ELEMENT Nombre del elemento ANY>
 - Ejemplos:
 - DTD.
 - <!DOCTYPE agenda [<!ELEMENT nombre ANY>]>
 - XML.
 - <nombre> Virginia</nombre> // Correcto.
 - <nombre> <teléfono></teléfono> </nombre> //Correcto.
 - <nombre/> // Correcto.
- **Elementos.**
 - Pueden incluirse elementos indicando que éstos aparecerán anidados al que los incluye.
 - Sintaxis:
 - <!ELEMENT Nombre del elemento (Elemento1, Elemento2, ..., ElementoN)>
 - Ejemplos:
 - DTD.
 - <!DOCTYPE agenda [<!ELEMENT cliente (código, nombre)>]
<!ELEMENT código (#PCDATA)>
<!ELEMENT nombre(#PCDATA)>]>
 - XML.
 - <cliente>

```

        <código> 2785NT6</código>
        <nombre>Juan</nombre>
    </cliente>
    ○ <cliente>
        <nombre>Juan</nombre>
        <código> 2785NT6</código> // (Incorrecto. No
        están el mismo orden que en la declaración).
    </cliente>

```

▪ Conectores y Modificadores para elementos.

▪ **Coma (,).**

- Separador de elementos.
- El elemento contiene otros elementos en el orden especificado.
- Sintaxis y ejemplo anterior.

▪ **O lógico (|).**

- El elemento contiene uno de los elementos especificados.
- Sintaxis:
 - <!ELEMENT Nombre del elemento (Elemento1 | Elemento2)>
- Ejemplos:
 - DTD.
 - <!DOCTYPE agenda [<!ELEMENT cliente (código | nombre) >
 - <!ELEMENT código (#PCDATA)>
 - <!ELEMENT nombre(#PCDATA)>
 - XML.
 - <cliente>
 - <nombre>Juan</nombre>
 - </cliente>
 - <cliente>
 - <codigo>HYU789</codigo>
 - </cliente>

▪ **Interrogación (?).**

- El elemento puede aparecer una o ninguna vez.
- Sintaxis:
 - <!ELEMENT Nombre del elemento (Elemento1, Elemento2?)>
- Ejemplos:
 - DTD.
 - <!DOCTYPE agenda [<!ELEMENT cliente (código?, nombre) >
 - <!ELEMENT código (#PCDATA)>
 - <!ELEMENT nombre(#PCDATA)>
 - XML.

- `<cliente> //Correcto.`
`<nombre>Juan</nombre>`
`</cliente>`
- `<cliente>`
`<código> 2785NT6</código>`
`<nombre>Juan</nombre>`
`</cliente> // Correcto.`

▪ **Asterisco (*).**

- El elemento puede aparecer ninguna, una o varias veces.
- Sintaxis:
 - `<!ELEMENT Nombre del elemento (Elemento1*, Elemento2)>`
- Ejemplos:
 - DTD.
 - `<!DOCTYPE agenda [<!ELEMENT cliente (código, nombre*) >`
`<!ELEMENT código (#PCDATA)>`
`<!ELEMENT nombre(#PCDATA)>`
`]>`
 - XML.
 - `<cliente>`
`<código> 2785NT6</código>`
`<nombre>Juan</nombre>`
`</cliente> //Correcto`
 - `<cliente>`
`<código> 2785NT6</código>`
`</cliente> // Correcto.`

▪ **Signo más (+).**

- El elemento puede aparecer una o varias veces.
- Sintaxis:
 - `<!ELEMENT Nombre del elemento (Elemento1+, Elemento2)>`
- Ejemplos:
 - DTD.
 - `<!DOCTYPE agenda [<!ELEMENT cliente (código, nombre+) >`
`<!ELEMENT código (#PCDATA)>`
`<!ELEMENT nombre(#PCDATA)>`
`]>`
 - XML.
 - `<cliente>`
`<código> 2785NT6</código>`
`<nombre>Juan</nombre>`
`</cliente> //Correcto`
 - `<cliente>`
`<código> 2785NT6</código>`
`</cliente> // Incorrecto.`

- `<cliente>`
`<código> 2785NT6</código>`
`<nombre>Juan</nombre>`
`<nombre>Juanito</nombre>`
`</cliente> // Correcto.`

- **Paréntesis ().**

- Sirve para agrupar expresiones.
- Sintaxis:
 - `<!ELEMENT Nombre del elemento (Elemento1, (Elemento2, Elemento3))>`
- Ejemplos:
 - DTD.
 - `<!DOCTYPE agenda [<!ELEMENT cliente (código, (cuenta | nombre)) >`
`<!ELEMENT código (#PCDATA)>`
`<!ELEMENT nombre(#PCDATA)>`
`]>`
 - XML.
 - `<cliente>`
`<código> 2785NT6 </cuenta>`
`<cuenta> 1234 56 78 9378235 </cuenta>`
`</cliente> //Correcto`
 - `<cliente>`
`<código> 2785NT6</código>`
`<nombre>Juan</nombre>`
`</cliente> // Correcto.`

- **ATRIBUTOS.**

- Sintaxis:
 - **Un atributo para un elemento:**
 - `<!ATTLIST Nombre del Elemento Nombre del Atributo Tipo de Atributo Valor Inicial del Atributo >`
 - **Varios atributos para un elemento:**
 - Opción 1:
 - `<!ATTLIST Nombre del Elemento Nombre del Atributo1 Tipo de Atributo1 Valor Inicial del Atributo1 >`
 - `<!ATTLIST Nombre del Elemento Nombre del Atributo2 Tipo de Atributo2 Valor Inicial del Atributo2 >`
 - Opción 2:
 - `<!ATTLIST Nombre del Elemento`
`Nombre del Atributo1 Tipo de Atributo1 Valor Inicial del Atributo1`
`Nombre del Atributo1 Tipo de Atributo1 Valor Inicial del Atributo1>`
 - **Componentes:**
 - Nombre del elemento.
 - Elemento para el cual se crea el atributo.
 - Nombre del atributo.
 - Nombre que llevará el atributo.

- Tipo de atributo.
 - A continuación.
 - Valor inicial del atributo.
 - Valor predeterminado del atributo.
- Ejemplo:
 - <!ATTLIST nombre alias CDATA "" >
- **Tipos de Atributos.**
 - **Valores.**
 - El valor del atributo solo puede ser uno de los incluidos en una lista.
 - Los valores se incluyen entre paréntesis usando el modificador O Lógico (|).
 - Ejemplo:
 - DTD.
 - <!DOCTYPE letras [
 <!ELEMENT letra EMPTY>
 <!ATTLIST letra mayus (A| B |C) #REQUIRED >
]>
 - XML.
 - <letra mayus = "B" /> // Correcto.
 - <letra mayus = "T" /> // Incorrecto.
 - **CDATA.**
 - El atributo contiene cualquier carácter.
 - Ejemplo:
 - DTD.
 - <!DOCTYPE agenda [
 <!ELEMENT nombre EMPTY>
 <!ATTLIST nombre alias CDATA "" >
]>
 - XML.
 - <nombre alias = "Luismi" /> // Correcto.
 - <nombre alias = "Luismi #321" /> // Correcto.
 - <nombre/> // Incorrecto.
 - **NMTOKEN.**
 - El atributo sólo puede contener letras, números y los caracteres punto (.), guion medio (-), guion de subrayado (_) y dos puntos (:).
 - Ejemplo:
 - DTD.
 - <!DOCTYPE agenda [
 <!ELEMENT nombre EMPTY>
 <!ATTLIST nombre alias NMTOKEN "" >
]>
 - XML.
 - <nombre alias = "Luismi" /> // Correcto.
 - <nombre alias = "Luismi-321" /> // Correcto.
 - <nombre alias = "Luismi 321" /> // Incorrecto. Hay un espacio en blanco no permitido.
 - **NMTOKENS.**

- Igual que NMTOKEN incluyendo espacios en blanco.
- Ejemplo:
 - DTD.
 - `<!DOCTYPE agenda [
 <!ELEMENT nombre EMPTY>
 <!ATTLIST nombre alias NMTOKENS "" >
]>`
 - XML.
 - `<nombre alias = "Luismi" /> // Correcto.`
 - `<nombre alias = "Luismi-321" /> // Correcto.`
 - `<nombre alias = "Luismi 321" /> // Correcto.`
- **ID.**
 - El valor del atributo debe ser único.
 - Ejemplo:
 - DTD.
 - `<!DOCTYPE colores [
 <!ELEMENT color (#PCDATA)>
 <!ATTLIST color tipo ID #REQUIRED>
]>`
 - XML.
 - `<color tipo = "Cálido">Rojo</color> // Correcto.`
 - `<color tipo = "Cálido">Azul</color> // Incorrecto.`
- **IDREF.**
 - El valor del atributo debe coincidir con el valor de otro atributo.
 - Ejemplo:
 - DTD.
 - `<!DOCTYPE colores [
 <!ELEMENT color (#PCDATA)>
 <!ATTLIST color tipo ID #REQUIRED >
 <!ELEMENT tonalidad (#PCDATA)>
 <!ATTLIST tonalidad matiz IDREF #REQUIRED >
]>`
 - XML.
 - `<color tipo = "Cálido">Rojo</color> // Correcto.`
 - `<tonalidad matiz = "Cálido"> Anaranjada
</tonalidad> // Correcto.`
- **IDREFS.**
 - El valor del atributo es un conjunto de valores que coinciden con el valor ID de otros elementos.
 - Ejemplo:
 - DTD.
 - `<!DOCTYPE colores [
 <!ELEMENT color (#PCDATA)>
 <!ATTLIST color tipo ID #REQUIRED>
 <!ELEMENT tonalidad (#PCDATA)>
 <!ATTLIST tonalidad matiz IDREFS #REQUIRED>
]>`

- XML.
 - <color tipo = "Cálido">Rojo</color> // Correcto.
 - <color tipo = "Templado">Amarillo</color> // Correcto.
 - <tonalidad matiz = "Cálido Templado"> Anaranjada </tonalidad> // Correcto.
- **VALORES INICIALES.**
 - **Valor.**
 - Atributo con un valor predeterminado.
 - Sintaxis:
 - Tipo de Atributo "Valor predeterminado".
 - Ejemplo:
 - <!ATTLIST teléfono internacional CDATA "Desconocido">
 - **#REQUIRED.**
 - Atributo obligatorio.
 - No se especifica ningún valor predeterminado.
 - Sintaxis:
 - Tipo de Atributo #REQUIRED.
 - Ejemplo:
 - <!ATTLIST teléfono internacional CDATA #REQUIRED>
 - **#IMPLIED.**
 - Atributo opcional.
 - No se especifica ningún valor predeterminado.
 - Sintaxis:
 - Tipo de Atributo #IMPLIED.
 - Ejemplo:
 - <!ATTLIST teléfono internacional CDATA #IMPLIED>
 - **#FIXED.**
 - Atributo con un valor fijo.
 - Sintaxis:
 - #FIXED "Valor fijo"
 - Ejemplo:
 - <!ATTLIST teléfono internacional CDATA #FIXED "34">

XML SCHEMA.

Concepto y características.

- Es un modelo creado por el W3C como sucesor de DTD.
- Utiliza el XSD (XML Schema Definition) como lenguaje para describir la estructura y las restricciones de un documento XML.
- Con el lenguaje XSD se puede comprobar la validez de un documento XML, es decir, definir su estructura: qué elementos, qué tipos de datos, que atributos, en qué orden, cuántas veces se repiten, etc.

DOCUMENTO XML SCHEMA.

- Contiene los componentes y elementos que va a servir para validar un documento XML.

- Se guarda en un archivo con la extensión .xsd.

Elemento raíz del documento XML Schema.

- Todo esquema XML incluye el elemento raíz schema.
- Sintaxis:
 - `<xs:schema>...</xs:schema>`
- Esta etiqueta está definida en el estándar XSD y es la que permite definir esquemas siguiendo el estándar W3C.
- **Atributos.**
 - **xmlns** (XML Namespace).
 - Especifica el espacio de nombres para el esquema.
 - **Espacio de nombres:**
 - Un espacio de nombres XML es un conjunto de nombres identificados mediante una URL.
 - Proporciona elementos y atributos con nombre único para resolver la ambigüedad entre elementos o atributos que se llamen igual.
 - Sintaxis:
 - `xmlns:prefijo = URL` el espacio de nombres.
 - Ejemplo:
 - `<xs:schema xmlns:xs="http://www.w3c.org/2001/XMLSchema">`
 - Esto indica que los elementos y tipos de datos utilizados en el esquema provienen del espacio de nombres 'http://www.w3.org/2001/XMLSchema'
 - **targetNamespace.**
 - Especifica un espacio de nombres propio.
 - Sintaxis:
 - `targetNamespace: URL` del espacio de nombres propio.
 - Ejemplo:
 - `targetNamesapace:"http://www.miWeb.es/Esquema">`
 - **elementFormDefault.**
 - Indica si los elementos deben o no estar certificados por el espacio de nombre cuando se utilicen en el documento instancia XML.
 - Sintaxis:
 - `elementFormDefault = "Valores"`.
 - Valores:
 - **qualified**
 - Los elementos de los documentos XML que apunten a un espacio de nombres determinado, deben estar cualificados con un prefijo.
 - **unqualified.**
 - Los elementos no deben estar cualificados con un prefijo.
 - Valor por defecto.
 - **attributeFormDefault.**

- Indica si los atributos deben o no estar certificados por el espacio de nombre cuando se utilicen en el documento instancia XML.
- Sintaxis:
 - attributeFormDefault = “Valores” .
 - Valores:
 - **qualified**
 - Los atributos de los documentos XML que apunten a un espacio de nombres determinado, deben estar cualificados con un prefijo.
 - **unqualified.**
 - Los atributos no deben estar cualificados con un prefijo.
 - Valor por defecto.

DOCUMENTO INSTANCIA XML.

- Es el documento XML que llama o referencia al esquema XSD que va a servir para validarlo.
- Dicho documento tiene la siguiente estructura: cabecera y cuerpo.

CABECERA.

- **Declaración XML.**
 - Deben empezar por una declaración XML:
 - Sintaxis:
 - <?xml version="versión" encoding="JUEGO DE CARACTERES?>
 - Ejemplo:
 - <?xml version="1.0" encoding="UTF-8"?>
- **Elemento raíz.**
 - Elemento que engloba al resto de elemento del documento y que debe aparecer también en el XSD.
 - Sintaxis:
 - <elemento raíz atributos> ... </elemento raíz>
 - Atributos:
 - **xmlns:xsi.**
 - Atributo que permite declarar el espacio de nombres del esquema XSD.
 - Indica que queremos utilizar los elementos definidos en un determinado esquema.
 - Sintaxis:
 - xmlns:xsi = “URL”.
 - Ejemplo:
 - xmlns:xsi = <http://www.w3.org/2001/XMLSchema-instance>
 - indica que se van a los elementos definidos en <http://www.w3.org/2001/XMLSchema-instance>.
 - **noNamespaceSchemaLocation.**
 - Atributo que permite vincular el documento XML con el esquema local XSD.

- Especifica el nombre del archivo XSD que contiene las reglas de validación.
- Sintaxis:
 - "xsi:noNamespaceSchemaLocation = "Ruta de acceso/Nombre del archivo.xsd".
- Ejemplo:
 - "xsi:noNamespaceSchemaLocation = "clientes.xsd".
-

CUERPO.

- Contiene los elementos y atributos junto con los datos o información asociada a ellos.
- Incluye distintos tipos de **declaraciones y componentes**:

ELEMENTOS COMPLEJOS. (Complex Types).

- Elemento que contiene otros elementos o atributos anidados.
- Los elementos del documento XML se especifican con la declaración *element*.
- Sintaxis:
 - <xs:element name="Nombre del Elemento"/> (Elementos simples)
 - <xs:element name="Nombre del Elemento"> ... </xs:element> (Elementos complejos)
- Ejemplo:
 - <xs:element name="Libro"/>
 - <xs:element name="Libro"> ... </xs:element>
- Si, además, incluye otros elementos anidados, se dice que es un elemento complejo e incluye la declaración *complexType* para especificar los elementos que lo componen, así como, otro tipo de características.
- Sintaxis:
 - <xs:complexType> Declaración de elementos anidados </xs:complexType>
- Ejemplo:

```
<xs:element name="Libro">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="Título" type="xsd:string"/>
      <xs:element name="Autor" type="xsd:string" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

- **Indicadores de orden.**
 - Especifican el orden de aparición de los elementos anidados.
 - Tipos:
 - **Secuencia.**
 - Define el orden exacto de aparición de los elementos.
 - Deben aparecer todos, y en el mismo orden en que se especifican.

- Sintaxis:
 - `<xs:sequence>...</xs:sequence>`
- **Alternativa.**
 - Define una serie de elementos entre los cuales sólo se puede elegir o debe aparecer uno de ellos.
 - El elemento que define la selección, así como sus elementos contenidos deben ser opcionales, es decir, deben incluir el atributo *minOccurs* con valor igual a 0, mientras que el valor de *maxOccurs* puede ser cualquiera que se necesite.
 - Sintaxis:
 - `<xs:choice>...</xs:choice>`
 - Ejemplo:

```
<xs:choice minOccurs = "0" maxOccurs = "5">
  <xs:element name="Título" type="xsd:string" minOccurs = "0"/>
  <xs:element name="Autor" type="xsd:string" minOccurs = "0"/>
</xs:choice>
```
- **Todos.**
 - No es obligatorio que aparezcan todos los elementos y además pueden hacerlo en cualquier orden.
 - Sintaxis:
 - `<xs:all>...</xs:all>`

ELEMENTOS SIMPLES. (Simple Types).

- Elementos que no contienen otros elementos.
- Los elementos del documento se especifican con la declaración `xs:element`.
- Incluyen un nombre, un tipo de datos y otros ajustes, como si son obligatorios o tienen valores por defecto.
- Sintaxis:
 - `<xs:element name = "Nombre del elemento" type = "Tipo de datos" Otros ajustes/>`
 - **Tipos de datos.** (Más comunes).
 - `xs:string`.
 - `xs:decimal`.
 - `xs:int`.
 - `xs:integer`.
 - `xs:boolean` (Valores true / false).
 - `xs:date` (Formato Año-Mes-Día).
 - `xs:time` (Formato Hora : Minutos : Segundos).
 - **Otros.**
 - `xs:positiveInteger`.
 - `xs:negativeInteger`.
 - `xs:dateTime` (Formato Año-Mes-Día Hora : Minutos : Segundos).
 - `xs:anyURI` (Indicar una URL).
 - **Otros ajustes.**
 - **Valores por defecto** (opcional).

- Valor que se asigna automáticamente al elemento si no se especifica otro.
 - Sintaxis:
 - `<xs:element name ="Nombre del elemento" type = "Tipo de datos" default "Valor por defecto">`
 - Ejemplo:
 - `<xs:element name ="email" type = "xs:string" default "Desconocido">`
- **Valor fijo.** (opcional).
 - Valor que se asigna automáticamente al elemento y que no se puede cambiar por otro.
 - Sintaxis:
 - `<xs:element name ="Nombre del elemento" type = "Tipo de datos" fixed "Valor fijo">`
 - Ejemplo:
 - `<xs:element name ="idioma" type = "xs:string" fixed "Español">`
- **Ocurrencias.** (opcional).
 - Se puede especificar el mínimo y máximo número de ocurrencia que pueden darse para un elemento.
 - Valores máximo y mínimo por defecto: 1.
 - Si se quiere que el máximo valor sea indefinido, el valor de `maxOccurs` debe ser *unbounded*.
 - `minOccurs = 0` indica que el elemento es opcional y puede no aparecer.
 - Sintaxis:
 - `<xs:element name ="Nombre del elemento" type = "Tipo de datos" minOccurs ="Valor" maxOccurs ="Valor">`
 - Ejemplo:
 - `<xs:element name ="hora" type = "xs:time" minOccurs ="1" maxOccurs ="50">`
- Ejemplos:
 - **XSD.**
 - `<xs:element name="nombre" type="xs:string" maxOccurs="4"/>`
`<xs:element name="edad" type="xs:integer" default "00"/>`
`<xs:element name="fecha_nacimiento" type="xs:date"/>`
 - **XML.**

```
<nombre>Luis</nombre>
<edad>24</edad>
<fecha_nacimiento>1997-09-10</fecha_nacimiento>
```
- **ATRIBUTOS.**
 - Se declaran igual que un elemento simple.
 - Un elemento simple no puede llevar atributos y si los lleva, se considera un elemento complejo.
 - Los elementos del documento se especifican con la declaración `xs:attribute`.

- Incluyen un nombre, un tipo de datos y otros ajustes, como si son obligatorios o tienen valores por defecto.
- **Tipos de datos.**
 - xs:string.
 - xs:decimal.
 - xs:int.
 - xs:integer.
 - xs:positiveInteger.
 - xs:negativeInteger.
 - xs:boolean (Valores true / false).
 - xs:date (Formato Año-Mes-Día)
 - xs:dateTime (Formato Año-Mes-Día Hora : Minutos : Segundos).
 - xs:time (Formato Hora : Minutos : Segundos).
 - xs:anyURI (Indicar una URL).
- Sintaxis:
 - <xs:attribute name = "Nombre del atributo" type = "Tipo de datos" Otros ajustes>
 - Ejemplos:
 - **XSD.**
 - <xs:attribute name="idioma" type="xs:string"/>
 - **XML.**
 - <título idioma = "inglés">Hamlet</título>
- **Otros ajustes.**
 - **Valores por defecto.**
 - Valor que se asigna automáticamente al atributo si no se especifica otro.
 - Sintaxis:
 - <xs:attribute name = "Nombre del atributo" type = "Tipo de datos" default "Valor por defecto">
 - Ejemplo:
 - <xs:attribute name = "Idioma" type = "xs:string" default "ES">
 - **Valor fijo.**
 - Valor que se asigna automáticamente al atributo y que no se puede cambiar por otro.
 - Sintaxis:
 - <xs:attribute name = "Nombre del atributo" type = "Tipo de datos" fixed "Valor fijo">
 - Ejemplo:
 - <xs:attribute name = "Idioma" type = "xs:string" fixed "Español">
 - **use.**
 - Un atributo puede ser opcional, obligatorio o no parecer en el elemento.
 - Valores:
 - **Optional.**
 - Atributo opcional.
 - Valor por defecto.
 - **Required.**
 - El atributo es obligatorio.

- **Prohibited.**
 - El atributo no debe aparecer en el documento XML.
 - Sintaxis:
 - `<xs: attribute name = " Nombre del atributo" type = "Tipo de datos" use "Valor">`
 - Ejemplo:
 - `<xs:attribute name ="Idioma" type = "xs:string" use ="required">`
- **EXTENSIONES XSD.**
 - Sirven para extender y por tanto ampliar las características de un elemento simple o complejo.
 - Por ejemplo, se puede usar para añadir un atributo a un elemento simple.
 - Sintaxis:

```

<xs: extensión base = "Tipo de datos del elemento que se extiende">
    <características extendidas o añadidas con su propio tipo de datos>
</xs: extensión>

```
 - Ejemplo:

```

<xs:extension base="xs:decimal">
    <xs:attribute name="divisa" type="xs:string"/>
</xs:extension>

```
- **simpleContent.**
 - Permite definir restricciones o extensiones a elementos que solo contienen datos, es decir, no contienen a otros elementos.
 - Si a un elemento simple, que solo contienen datos, se le añade o extiende por ejemplo un atributo, se convierte en un elemento complejo, por lo que hay que utilizar la etiqueta `<complexType>` también.
 - Sintaxis:

```

<xs:element elemento simple>
    <xs:complexType>
        <xs:simpleContent>
            <xs:extension>
                <Características añadidas o extendidas>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
</xs:element>

```
 - Ejemplo:

```

<xs:element name="saldo" maxOccurs="unbounded">
    <xs:complexType>
        <xs:simpleContent>
            <xs:extension base="xs:decimal"> // tipo de dato del elemento saldo
                <xs:attribute name="divisa" type="xs:string"/> // tipo de dato del
atributo divisa.
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
</xs:element>

```

- **simpleType.**

- Además de los tipos de datos predefinidos incorporados a XSD, se pueden definir tipos para elementos o atributos definidos por el usuario y también para crear restricciones (ver más adelante para estas últimas).
- Sintaxis para tipos nuevos:

```
<xs:element name="Nombre del elemento" type = "Nuevo tipo a crear">
  <xs:simpleType name = "Nombre del nuevo tipo a crear">
    <xs:restriction base=" Tipo de datos para el nuevo tipo a crear">
      <xs:tipo de restricción value = "valor"/> // Opcional.
    </xs:restriction>
  </xs:simpleType>
```

- Ejemplo:

```
<xs:element name="bonoloto" type = "numerosPermitidos">
  <xs:simpleType name = "numeroPermitidos">
    <xs:restriction base="xs:integer">
      <xs:minInclusive = "1"/>
      <xs:maxInclusive = "49"/>
    </xs:restriction>
  </xs:simpleType>
```

- **RESTRICCIONES o FACETAS XSD.**

- Especifican las condiciones que deben cumplir los datos.
- Las restricciones se establecen para distintos aspectos denominados facetas.
- Sintaxis:

```
<xs:restriction base=" Tipo de datos del elemento al que se le aplica la
restricción">
  <xs:tipo de restricción value = "valor"/>
</xs:restriction>
```

- Para crear una restricción, dentro de la etiqueta *simpleType* y para el elemento al cual se le aplica la restricción, se incluye una etiqueta *restriction* con las restricciones.

- Sintaxis:

```
<xs:element name="Nombre del elemento">
  <xs:simpleType>
    <xs:restriction base=" Tipo de datos del elemento al que se le aplica la
restricción">
      <xs:tipo de restricción value = "valor"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

- Ejemplo:

```
<xs:element name = "valores">
  <xs:simpleType>
    <xs:restriction base="xs:integer">
```

```

<xs:minInclusive value="20"/>
<xs:maxInclusive value="100"/>
</xs:restriction>
</xs:simpleType>
</xs:element>

```

- **Tipos de restricciones.**

- **Patrones.**

- Permiten el uso de expresiones y caracteres especiales para crear un patrón que especifica qué caracteres están permitidos.
 - Expresiones y caracteres especiales para crear patrones:

- **[]:**

- Especifica que valores son permitidos.
 - Se pueden incluir los valores permitidos exactos o, si son consecutivos, establecer un intervalo entre 2 valores extremos separados con un guion.
 - Varios juegos de corchetes representan a varios caracteres distintos.
 - Ejemplos:
 - [abc] Permitido *a*, *b* o *c*. Equivaldría a [a-c].
 - [35689] Permitido 3, 5, 6, 8 o 9.
 - [A-Z] Permitidas todas las letras mayúsculas.
 - [A-L] Permitidas todas las letras mayúsculas entre la A y la L.
 - [0-9] Sólo números.
 - [A-Za-z] Todas las letras mayúsculas y minúsculas.
 - [A-Z][149][a-m] 3 caracteres, el primero una letra mayúscula cualquiera, el segundo un dígito de entre los mostrados y el tercero, una minúscula entre las letras *a* y *m*.
 - C9e
 - D2a

- **{ }:**

- Especifica el total de caracteres o dígitos a incluir.
 - Sintaxis:
 - {número}
 - {número inicial, número final}
 - Ejemplo:
 - [0-9] {9} // El número 823542352 es correcto ya que el número total son 9 dígitos.
 - [0-9] {3,9} // El valor 98375 es correcto ya que el número mínimo son 3 dígitos.
 - [a-z] {5,20} // Por ejemplo, *ana* no es válido por tener menos de 5 caracteres, mientras que *antonio* si, por tener más de 5 caracteres y no pasarse de 20.
 - [0-9]{5} < = > [0-9] [0-9] [0-9] [0-9] [0-9] // Equivalentes para un código postal.

- **():**

- Permiten agrupar caracteres para indicar a cuáles les afecta un determinado cuantificador.

- **Cuantificadores.**

- Para expresar el número de ocurrencias de un valor se usan los siguientes caracteres:
 - *:
 - Indica que los valores introducidos pueden aparecer ninguna, una o varias veces.
 - +:
 - Indica que los valores introducidos pueden aparecer una o varias veces.
 - ?:
 - Indica que los valores introducidos pueden aparecer ninguna o una vez.
- Ejemplos:
 - [A-Z]* // El dato introducido puede tener ninguna, una o varias letras mayúsculas.
 - [1-9]([a-z])+ // El dato introducido debe empezar por un número comprendido entre el 3 y el 9, e ir seguido por un número variable de letras minúsculas. Una minúscula al menos.
 - (-)?[0-9]+ // Número con al menos un dígito que puede ser positivo o negativo.
- Sintaxis:
 - <xs:pattern value="patrón"/>
- Ejemplo:
 - Restricción para especificar con un patrón que un DNI debe contener 9 caracteres, siendo los 8 primeros dígitos y el último, una letra mayúscula.

```
<xs:element name="dni">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:pattern value="[0-9]{8}[A-Z]{1}"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```
- **minExclusive.**
 - Especifica el mínimo valor no incluido que admite un elemento.
 - Sintaxis:
 - <xs:minExclusive = "valor"/>
 - Ejemplo:
 - <xs:minExclusive = "10"/> El valor 11 está permitido, pero no el 10.
- **maxExclusive.**
 - Especifica el máximo valor no incluido que admite un elemento.
 - Sintaxis:
 - <xs:maxExclusive = "valor"/>
 - Ejemplo:
 - <xs:maxExclusive = "100"/> El valor 99 está permitido, pero no el 1100.

- **minInclusive.**
 - Especifica el mínimo valor incluido que admite un elemento.
 - Sintaxis:
 - `<xs:minInclusive = "valor"/>`
 - Ejemplo:
 - `<xs:minInclusive = "10"/>` El valor 10 está permitido.
- **maxInclusive.**
 - Especifica el máximo valor incluido que admite un elemento.
 - Sintaxis:
 - `<xs:maxInclusive = "valor"/>`
 - Ejemplo:
 - `<xs:maxInclusive = "100"/>` El valor 100 está permitido.
- **totalDigits.**
 - Especifica el número máximo de dígitos que puede tener una cifra.
 - Sintaxis:
 - `<xs:totalDigits = "valor"/>`
 - Ejemplo:
 - `<xs:totalDigits = "4"/>` El valor 450 es válido, pero no el número 12560.
- **fractionDigits.**
 - Especifica el número máximo de decimales que puede tener una cifra.
 - Sintaxis:
 - `<xs:fractionDigits = "valor"/>`
 - Ejemplo:
 - `<xs:fractionDigits = "2"/>` El valor 45,23 es válido, pero no el número 45,9823.
- **length.**
 - Especifica el número exacto de caracteres. Puede usarse con textos y números.
 - Sintaxis:
 - `<xs:length = "valor"/>`
 - Ejemplo:
 - `<xs:length = "6"/>` Toledo es correcto, Ávila o Zaragoza no.
- **minLength.**
 - Especifica el mínimo número de caracteres. Puede usarse con textos y números.
 - Sintaxis:
 - `<xs:minLength = "valor"/>`
 - Ejemplo:
 - `<xs:minLength = "6"/>` Toledo y Zaragoza son correctos, Ávila no.
- **maxLength.**
 - Especifica el máximo número de caracteres. Puede usarse con textos y números.
 - Sintaxis:
 - `<xs:maxLength = "valor"/>`
 - Ejemplo:
 - `<xs:maxLength = "6"/>` Zaragoza no es correcto, Ávila sí.

○ **enumeration.**

- Permite crear una lista de valores permitidos.
- Sintaxis:
 - <xs:enumeration = “valor 1”/>
 - <xs:enumeration = “valor 2”/>
 - <xs:enumeration = “valor N”/>
- Ejemplo:
 - <xs:enumeration = “rojo”/>
 - <xs:enumeration = “verde”/>
 - <xs:enumeration = “azul”/>

CARACTERES ESPECIALES PARA CREAR PATRONES.

PATRÓN	SIGNIFICADO	EJEMPLO
.	Cualquier carácter.	;
\w	Cualquier letra.	M
\d	Un dígito.	7
\D	Cualquier carácter que no sea un dígito.	j
\s	Espacios (tabulador, espacio, salto línea).	
\S	Cualquier carácter que no sea un espacio.	C
\d{n}	Cifra con número de dígitos n exacto.	3856
\d{n,m}	Numero de dígitos comprendido ente n y m.	91
\d{n,}	Número de dígitos comprendido ente n y cualquier valor máximo.	3500
[xyz]	Uno de los 3 caracteres.	Y
[A-Z]	Uno de los caracteres en mayúsculas entre la A y la Z.	M
[^abc]	Ninguno del conjunto de caracteres.	z
[F-J-[H]]	Eliminar un carácter de un tramo.	G
(a b)	Elección entre dos expresiones.	a
b?	Una o ninguna aparición del carácter precedente.	b
1*	Ninguna, una o apariciones de los caracteres precedentes.	111
(cd)+	Una o más apariciones de los caracteres precedentes. Los paréntesis permiten agrupar caracteres.	cdcd