SCHEMAS

Documentación generada automáticamente

Tamara

None

Table of contents

1. Lenguaje de Marcas (LMSGI): 3
1.1 UD04-Schemas 3
1.1.1 CURSO 2025-2026 3
2. UND4-SCHEMAS (XSD) 4
2.1 Lenguaje de Marcas (LMSGI): Schemas 4
2.2 Tipos de datos 5
2.3 Facetas de los tipos de datos 6
2.4 Elementos del lenguaje 8
2.5 Definición de tipos de datos XML Schema 10
2.6 Asociación con documentos XML 12
2.7 Documentación del esquema 13
2.8 Herramientas de creación y validación 14
3. CONSEJOS 15
3.1 Cosas a tener en cuenta: 15
3.2 Proceso de validación de XML contra XSD 15

1. Lenguaje de Marcas (LMSGI):

1.1 UD04-Schemas

1.1.1 CURSO 2025-2026

Descargar toda la documentación en PDF

2. UND4-SCHEMAS (XSD)

2.1 Lenguaje de Marcas (LMSGI): Schemas

- Los DTD permiten diseñar un vocabulario para ficheros XML, pero, ¿qué sucede cuando los valores de los elementos y atributos de esos ficheros han de corresponder a datos de un tipo determinado, o cumplir determinadas restricciones que no pueden reflejarse en los DTD? Para ello se definen XML Schemas.
- ¿También se definen en ficheros planos? Si, ya que son documentos XML, pero en este caso la extensión de los archivos es xsd, motivo por el cual también se les denomina documentos XSD.
- Los elementos XML que se utilizan para generar un esquema han de pertenecer al espacio de nombre XML Schema, que es: http://www.w3.org/2001/XMLSchema.
- El ejemplar de estos ficheros es xs:schema, contiene declaraciones para todos los elementos y atributos que puedan aparecer en un documento XML asociado válido. Los elementos hijos inmediatos de este ejemplar son xs:element que nos permiten crear globalmente un elemento. Esto significa que el elemento creado puede ser el ejemplar del documento XML asociado.

2.2 Tipos de datos

- Son los distintos valores que puede tomar el atributo type cuando se declara un elemento o un atributo y representan el tipo de dato que tendrá el elemento o atributo asociado a ese type en el documento XML.

Algunos de estos valores predefinidos son:

- **string**, se corresponde con una cadena de caracteres UNICODE.
- **boolean**, representa valores lógicos, es decir que solo pueden tomar dos valores, true o false.
- **integer**, número entero positivo o negativo.
- **positiveInteger**, número entero positivo.
- **negativeInteger**, número entero negativo.
- **decimal**, número decimal, por ejemplo, 8,97.
- **dateTime**, representa una fecha y hora absolutas.
- **duration**, representa una duración de tiempo expresado en años, meses, días, horas, minutos segundos. El formato utilizado es: PnYnMnDTnHnMnS. Por ejemplo para representar una duración de 2 años, 4 meses, 3 días, 5 horas, 6 minutos y 10 segundos habría que poner: P2Y4M3DT5H6M7S. Se pueden omitir los valores nulos, luego una duración de 2 años será P2Y. Para indicar una duración negativa se pone un signo – precediendo a la P.
- **time**, hora en el formato hh:mm:ss.
- **date**, fecha en formato CCYY–MM–DD.
- **gYearMonth**, representa un mes de un año determinado mediante el formato CCYY–MM.
- **gYear**, indica un año gregoriano, el formato usado es CCYY.
- **gMothDay**, representa un día de un mes mediante el formato –MM–DD.
- **gDay**, indica el ordinal del día del mes mediante el formato –DD, es decir el 4º día del mes será –04.
- **gMonth**, representa el mes mediante el formato –MM. Por ejemplo, febrero es –02.
- **anyURI**, representa una URI.
- **language**, representa los identificadores de lenguaje, sus valores están definidos en RFC 1766.
- **ID, IDREF, ENTITY, NOTATION, MTOKEN**.Representan lo mismo que en los DTD's En este enlace encontrarás los tipos de datos admitidos por el estándar. XML Schema Tipos de datos https://www.w3.org/TR/xmlschema-2/

2.3 Facetas de los tipos de datos

¿Cuáles son las restricciones que podemos aplicar sobre los valores de los datos de un elemento o atributo?

Están definidos por las facetas, que solo pueden aplicarse sobre tipos simples utilizando el elemento xs:restriction. Se expresan como un elemento dentro de una restricción y se pueden combinar para lograr restringir más el valor del elemento. Son, entre otros:

- **length, minLength, maxLentgh**: Longitud del tipo de datos.
- **enumeration**: Restringe a un determinado conjunto de valores. ***Ejercicio resuelto:*** Creación de una cadena de texto con una longitud máxima de 9 caracteres y dos valores posibles.

Ejemplo de uso

```
<xs:simpleType name="estado">
    <xs:restriction base="xs:string">
    <xs:maxLength value="9"/>
    <xs:enumeration value="conectado"/>
    <xs:maxLength value="ocupado"/>
    </xs:restriction>
</xs:simpleType>
```

- **whitespace:** Define el tratamiento de espacios (preserve/replace, collapse).

Ejercicio resuelto: Creación de un elemento en el que se respetan los espacios tal y como se han introducido.

Ejemplo de uso

```
<xs:simpleType name="nombre">
    <xs:restriction base="xs:string">
    <xs:whitespac value="preserve"/>
    </xs:restriction>
</xs:simpleType>
```

- **(max/min)(In/Ex)clusive:** Límites superiores/inferiores del tipo de datos. Cuando son Inclusive el valor que se determine es parte del conjunto de valores válidos para el dato, mientras que cuando se utiliza Exclusive, el valor dado no pertenece al conjunto de valores válidos.
- **totalDigits, fractionDigits:** número de dígitos totales y decimales de un número decimal. ***Ejercicio resuelto:*** Creación de un elemento calificaciones de dos dígitos cuyo valor es un número entero comprendido entre 1 y 10, ambos inclusive.

Ejemplo de uso

```
<xs:simpleType name="calificaciones">
    <xs:restriction base="xs:integer">
    <xs:totalDigits value="2"/>
    <xs:minExclusive value="0"/>
    <xs:maxInclusive value="10"/>
    </xs:restriction>
</xs:simpleType>
```

- **pattern:** Permite construir máscaras que han de cumplir los datos de un elemento. La siguiente tabla muestra algunos de los caracteres que tienen un significado especial para la generación de las máscaras.

Elementos para hacer patrones.			
Patron	**Significado**	**Patron**	**Significado**
[A-Z a-z]	Letra.	**AB**	Cadena que es la concatenación de las cadenas A y B.
[A-Z]	Letra mayúscula.	**A?**	Cero o una vez la cadena A.
[a-z]	Letra minúscula.	**A+**	Una o más veces la cadena A.
[0-9]	Dígitos decimales.	**A***	Cero o más veces la cadena A.
\D	Cualquier carácter excepto un dígito decimal.	**[abcd]**	Alguno de los caracteres que están entre corchetes.
(A)	Cadena que coincide con A.	**[^abcd]**	Cualquier carácter que no esté entre corchetes.
A | B	Cadena que es igual a la cadena A o a la B.	**\t**	Tabulación.

Ejercicio resuelto: El ejemplo siguiente muestra la utilización de pattern para crear la máscara de un DNI.

Ejemplo de uso

```
<xs:simpleType name="dni">
    <xs:restriction base="xs:string">
    <xs:pattern value="[0-9] [0-9] [0-9] [0-9] [0-9] [0-9] [0-9] [0-9] [0-9] [A-z]"/>
    </xs:restriction>
</xs:simpleType>
```

Descargar toda la documentación en PDF

2.4 Elementos del lenguaje

Algunos de los más usados son:

- Esquema, **xs:schema**, contiene la definición del esquema.
- Tipos complejos, **xs:complexType**, define tipos complejos.
- Tipos simples, **xs:simpleType**, permite definir un tipo simple restringiendo sus valores.
- Restricciones, **xs:restriction**, permite establecer una restricción sobre un elemento de tipo base.
- Agrupaciones, **xs:group**, permite nombrar agrupaciones de elementos y de atributos para hacer referencia a ellas.
- Secuencias, **xs:secuence**, permite construir elementos complejos mediante la enumeración de los que les forman.
- Alternativa, **xs:choice**, representa alternativas, hay que tener en cuenta que es una o-exclusiva.
- Contenido mixto, definido dando valor true al atributo mixed del elemento **xs:complexType**, permite mezclar texto con elementos.
- Secuencias no ordenadas, **xs:all**, representa a todos los elementos en cualquier orden.

Ejercicio resuelto: Esquema correspondiente a un documento XML para estructurar la información personal sobre alumnos.

!!! example "Ejemplo de uso"

```xml
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
 <!--Elemento raíz-->
  <xs:element name="alumno" type="datosAlum"/>

  <!--Definición del tipo datosAlum-->
   <xs:complexType name="datosAlum">
    <xs:sequence>
     <xs:element name="alumno" type="datos" minOccurs="1" maxOccurs="unbounded"/>
    </xs:sequence>
   </xs:complexType>

    <!--Definición del tipo datos-->
   <xs:complexType name="datos">
    <xs:sequence>
     <xs:element name="nombre" type="xs:string" minOccurs="1" maxOccurs="1"/>
     <xs:element name="apellidos" type="xs:string" minOccurs="1" maxOccurs="1"/>
     <xs:element name="direccion" type="datosDireccion" minOccurs="1" maxOccurs="1"/>
     <xs:element name="contactar" type="datosContactar" minOccurs="1" maxOccurs="1"/>
    </xs:sequence>

    <!--Atributos del elemento usuario-->
     <xs:attribute name="id" type="xs:string"/>
    </xs:complexType>
    <!--Definición del tipo datosDireccion-->
   <xs:complexType name="datosDireccion">
    <xs:sequence>
     <xs:element name="domicilio" type="xs:string" minOccurs="0" maxOccurs="1"/>
     <xs:element name="codigo postal" type="xs:string" minOccurs="1" maxOccurs="1">
     <xs:complexType>
      <xs:attribute name="cp" type="xs:string"/>
     </xs:complexType>
    </xs:element>
    <xs:element name="localidad" type="xs:string" minOccurs="0" maxOccurs="1"/>
     <xs:element name="provincia" type="xs:string" minOccurs="0" maxOccurs="1"/>
    </xs:sequence>
   </xs:complexType>

    <!--Definición del tipo datosContactar-->
   <xs:complexType name="datosContactar">
    <xs:sequence>
     <xs:element name="telf_casa" type="xs:string" minOccurs="0" maxOccurs="1"/>
     <xs:element name="telf_movil" type="xs:string" minOccurs="0" maxOccurs="1"/>
     <xs:element name="telf_trabajo" type="xs:string" minOccurs="0" maxOccurs="1"/>
     <xs:element name="email" minOccurs="0" maxOccurs="1">
     <xs:complexType>
      <xs:attribute name="href" type="xs:string"/>
     </xs:complexType>
      </xs:element>
    </xs:sequence>
   </xs:complexType>
</xs:schema>
```

2.5 Definición de tipos de datos XML Schema

En los DTD se diferencia entre los elementos terminales y los no terminales ¿en este caso también? Si, este lenguaje permite trabajar tanto con datos simples como con estructuras de datos complejos, es decir compuestos por el anidamiento de otros datos simples o compuestos.

Tipos de datos simples: Estos datos se suelen definir para hacer una restricción sobre un tipo de datos XDS ya definido y establece el rango de valores que puede tomar.

Ejercicio resuelto: Creación de un elemento simple de nombre edad que representa la edad de un alumno de la ESO, por tanto su rango está entre los 12 y los 18 años.

Ejemplo de uso

```
<xs:simpleType name="edad">
    <xs:restriction base="xs:positiveInteger">
    <xs:maxExclusive value="10"/>
    </xs:restriction>
</xs:simpleType>
```

Ejercicio resuelto: Creación de una lista con los días de la semana en letras.

```
<?xml version="1.0" encoding="UTF-8"?>
<catalogo xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="ejemplo_Lista.xsd">
 <dias>Lunes Martes Miercoles Jueves Viernes Sabado Domingo</dias>
</catalogo>
```

Ejemplo de uso

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

<!--Elemento raíz "catalogo"-->
  <xs:element name="catalogo">
    <xs:complexType>
      <xs:sequence>
      <!--Elemento raíz "dias" que utiliza el tipo "DiasType"-->
        <xs:element name="dias" type="DiasType" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <!--Definición del tipo  "DiasType" como una lista de Strings-->
  <xs:simpleType name="DiasType">
    <xs:list itemType ="xs:string"/>
  </xs:simpleType>
</xs:schema>
```

- **Tipos de datos compuestos:** El elemento **xsd:complexType** permite definir estructuras complejas de datos. Su contenido son las declaraciones de elementos y atributos, o referencias a elementos y atributos declarados de forma global. Para determinar el orden en que estos elementos aparecen en el documento XML se utiliza el elemento . ***Ejercicio resuelto:*** Creación de un elemento compuesto de nombre alumno, formado por los elementos nombre, apellidos, web personal.

Ejemplo de uso

```
<xs:complexType  name="alumno">
      <xs:sequence>
        <xs:element name="nombre" type="xs:string" minOccurs="1" maxOccurs="unbounded"/>
        <xs:element name="apellidos" type="xs:string" minOccurs="1" maxOccurs="unbounded"/>
        <xs:element name="web" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
            <xs:complexType>
                <xs:attribute name="href" type="xs:string"/>
            </xs:complexType>
      </xs:sequence>
    </xs:complexType>
```

2.6 Asociación con documentos XML

Una vez que tenemos creado el fichero XSD ¿cómo lo asociamos a un fichero XML?

El modo de asociar un esquema a un documento XML es un espacio de nombres al ejemplar del documento, donde se indica la ruta de localización de los ficheros esquema mediante su URI, precedida del prefijo " xsi:"

Ejemplo de como asoscir un xsd a un xml

2.7 Documentación del esquema

Una vez que hemos visto como crear un esquema vamos a ver el modo de incorporar cierta documentación (quién es el autor, limitaciones de derechos de autor, utilidad del esquema, etc.) al mismo.

- Podemos pensar que un método para añadir esta información es utilizar comentarios. El problema es que los analizadores no garantizan que los comentarios no se modifiquen al procesar los documentos y por tanto, que los datos añadidos no se pierdan en algún proceso de transformación del documento.
- En lugar de usar los comentarios, XML Schema tiene definido un elemento xs:annotation que permite guardar información adicional. Este elemento a su vez puede contener una combinación de otros dos que son:
- xs:documentation, además de contener elementos de esquema puede contener elementos XML bien estructurados. También permite determinar el idioma del documento mediante el atributo xml:lang.
- xs:appinfo, se diferencia muy poco del elemento anterior, aunque lo que se pretendió inicialmente era que xs:documentation fuese legible para los usuarios y que xs:appinfo guardase información para los programas de software.

También es usado para generar una ayuda contextual para cada elemento declarado en el esquema.

Ejercicio resuelto: Ejemplo de documentación de un esquema.

Ejemplo de uso

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" >
  <xs:annotation>
    <xs:documentation xml:lang="es-es">
       Materiales para formación e-Learning
        <modulo>Lenguaje de marcas y ssitemas de gestión de información</modulo>
        <fecha_creacion>2011</fecha_creacion>
        <autos>Nuky La Bruji</autos>
    </xs:documentation>
</xs:annotation>
<xs:element name="lmsgi" type="xs:string">
  <xs:annotation>
    <xs:appinfo>
        <text_de_ayuda>Se debe de introducir el nombre completo del tema</text_de_ayuda>
    </xs:appinfo>
  </xs:annotation>
  </xs:element>
</xs:schema>
```

Descargar toda la documentación en PDF

2.8 Herramientas de creación y validación

Para crear y validar los documentos XML y los esquemas basta con un editor de texto plano y un navegador.Pero existen aplicaciones que permiten al usuario visualizar, validar y editar documentos en el lenguaje XML. Algunos de estos productos son:

- Editix XML Editor (Gratuito).
- Microsoft Core XML Services (MSXML) (Gratuito).
- XMLFox Advance.
- Altova XML Spy Edición Estándar.
- Editor XML xmlBlueprint.
- Editor Gráfico XSD y XML (XML Studio) (Gratuito).
- Estudio XML Líquido (Gratuito).
- Stylus Studio 2001 (Gratuito).
- Oxygen XML Editor.
- Exchanger XML Editor. ❓

3. CONSEJOS

3.1 Cosas a tener en cuenta:

> ⚠ **¡Atención!**
>
> Asegúrate de validar tu XML antes de enviarlo.

> ⚠ **¡Atención!**
>
> Se validan los xml no los xsd.

> 🔥 **Consejo útil**
>
> Utiliza las tabulaciones y los comentarios, te seran de gran ayuda.

3.2 Proceso de validación de XML contra XSD

A continuación se muestra el diagrama de flujo que representa el proceso de validación de un archivo XML utilizando un archivo XSD:

```
graph TD
  A[Inicio] --> B[Leer archivo XML]
  B --> C[Leer archivo XSD]
  C --> D[Validar XML contra XSD]
  D -->|Validación exitosa| E[Generar reporte de éxito]
  D -->|Error de validación| F[Generar reporte de error]
  E --> G[Finalizar]
  F --> G[Finalizar]
```

📄 **Descargar toda la documentación en PDF**