

Tarea: XSD

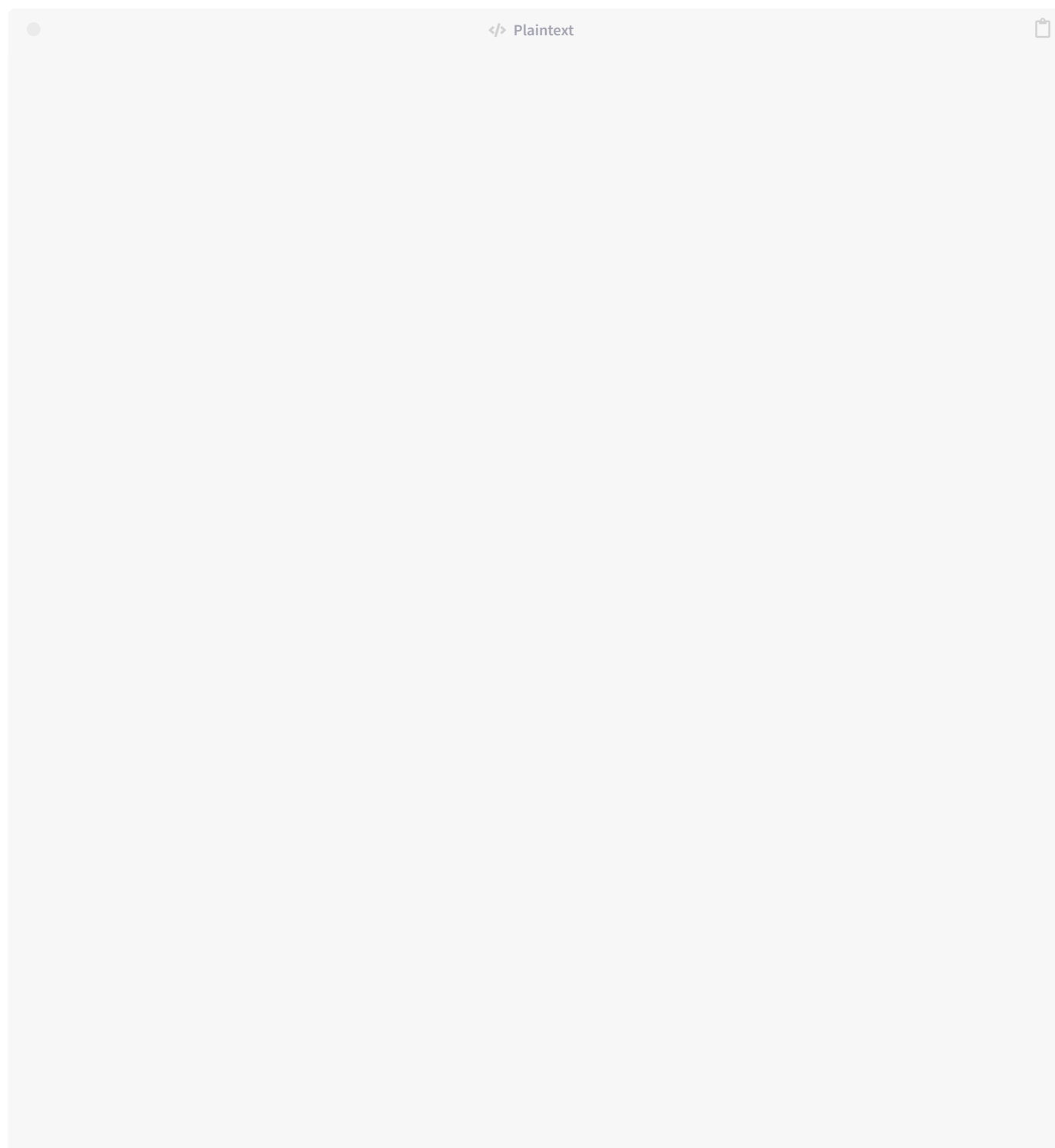
Marcos Ruiz

hace 20 días • Actualizado hace 3 días • 19 minutos lectura

Entrega y presentación

La entrega será en formato ZIP manteniendo la estructura de carpetas original. Leer [Entrega y presentación de tareas](#).

Ejemplo de entrega:



```
mruizg_t10.zip
├──mruizg_a01_validarMarcadores.xml
├──mruizg_a01_validarMarcadores.xsd
├──mruizg_a02_validarSitemap.xml
├──mruizg_a03_mensaje.xml
├──mruizg_a03_mensaje.xsd
├──mruizg_a04_definicionElementosSimples.xml.txt
├──mruizg_a04_definicionElementosSimples.xsd.txt
├──mruizg_a05_puertaCerradaVentanaAbierta.xsd.txt
├──mruizg_a06_fichas.xml
├──mruizg_a06_fichas.xsd
├──mruizg_a07_edad.xml
├──mruizg_a07_edad.xsd
├──mruizg_a08_precios.xml
├──mruizg_a08_precios.xsd
├──mruizg_a09_vehiculo.xsd.txt
├──mruizg_a10_iniciales.xml
├──mruizg_a10_iniciales.xsd
├──mruizg_a11_inicialesAlReves.xml
├──mruizg_a11_inicialesAlReves.xsd
├──mruizg_a12_respuestasAdmitidas.xsd.txt
├──mruizg_a13_numerosYLetras.xml
├──mruizg_a13_numerosYLetras.xsd
├──mruizg_a14_expresionesRegulares.txt
├──mruizg_a14_expresionesRegulares.xml
├──mruizg_a14_expresionesRegulares.xsd
├──mruizg_a15_letrasAdmitidas.xsd.txt
├──mruizg_a16_longitudFija.xsd.txt
├──mruizg_a17_longitudMinMax.xml
├──mruizg_a17_longitudMinMax.xsd
├──mruizg_a18_infoPersona.xml
├──mruizg_a18_infoPersona.xsd
├──mruizg_a19_preciosArticulos.xml
├──mruizg_a19_preciosArticulos.xsd
├──mruizg_a20_infoUbicaciones.xml
├──mruizg_a20_infoUbicaciones.xsd
├──mruizg_a21_coloresMuebles.xml
├──mruizg_a21_coloresMuebles.xsd
├──mruizg_a22_numerosBingo.xml
├──mruizg_a22_numerosBingo.xsd
├──mruizg_a23_infoPersonasMixto.xml
├──mruizg_a23_infoPersonasMixto.xsd
├──mruizg_a24_panelVuelos.xml
├──mruizg_a24_panelVuelos.xsd
├──mruizg_a25_factura.xml
├──mruizg_a25_factura.xsd
├──mruizg_a26_registroConexiones.xml
├──mruizg_a26_registroConexiones.xsd
├──mruizg_a27_personalDepartamentos.xml
└──mruizg_a27_personalDepartamentos.xsd
```



El ejemplo de entrega solo es una referencia de entrega, si se necesita eliminar o crear algún fichero o carpeta siéntete libre de hacerlo.

Pregunta 1: ► ¿Por qué hay ficheros .xml.txt?

Pregunta 2: ¿Puede un fichero XSD ser .xml?

Calificación

La tarea se calificará con una nota de 0 a 10.

Duración: - h

Actividades

Descarga [enunciado_tarea_xsd.zip](#) y realiza las siguientes actividades.

! Recuerda que NO hay que entregar ningún documento PDF contestando a las preguntas.

Actividad 1: (Voluntaria) Validar un documento XML

Comprobar con Visual Studio Code o XML Copy Editor que `a01_validarMarcadores.xml` es válido haciendo uso de `a01_validarMarcadores.xsd`.

Actividad 2: (Voluntaria) Validar un sitemap XML

Comprobar con un validador online de sitemaps XML que `a02_validarSitemap.xml` es válido.

Actividad 3: Mensaje entre personas

Dado el siguiente archivo `a03_mensaje.xsd`. Corregir los errores cometidos en el documento XML `a03_mensaje.xml`, para que sea válido. Para ello, no modificar la cuarta línea de código de `a03_mensaje.xml`:

```
</> XML
xmlns:men="marcosruiz.github.io/assets/img/tarea-xsd">
```

Actividad 4: Definición de elementos simples

Para los elementos del `a04_definicionElementosSimples.xml.txt` escribir sus definiciones de elementos simples correspondientes en `a04_definicionElementosSimples.xsd.txt`.

Actividad 5: Puerta cerrada y ventana abierta

Definir en `a05_puertaCerradaVentanaAbierta.xsd.txt` un elemento llamado `puertaCerrada` de tipo lógico, que por defecto tenga el valor “falso”, y otro elemento llamado `ventanaAbierta` también de tipo lógico, que tenga asignado el valor fijo “verdadero”.

Actividad 6: Fichas de personas

Dado el documento XML `a06_fichas.xml` escribir el contenido del archivo `a06_fichas.xsd` que permita validarlo.

Actividad 7: Edad entre 0 y 130 años

Dado el documento XML `a07_edad.xml` escribir el contenido del archivo `a07_edad.xsd` que permita validarlo, teniendo en cuenta que se debe definir la “edad” con la restricción de que el valor que tome no pueda ser menor que 0 ni mayor que 130. Además, en vez de `xs:minInclusive` y `xs:maxInclusive`, se debe utilizar:

- `xs:minExclusive` que sirve para especificar que el valor debe ser mayor que el indicado.
- `xs:maxExclusive` que sirve para especificar que el valor debe ser menor que el indicado.

Actividad 8: Precios de tres dígitos

Dado el documento XML `a08_precios.xml` escribir el contenido del archivo `a08_precios.xsd` que permita validarlo, teniendo en cuenta que el elemento “precio” puede tomar por valor un número que contenga tres dígitos como máximo y, de ellos, solamente dos pueden ser decimales. Para ello, escribir una restricción que no podrá ser utilizada por otros elementos y, por otra parte, haga uso de:

- `xs:totalDigits` que sirve para especificar el número máximo de dígitos que puede tener un número, incluyendo a los decimales.
- `xs:fractionDigits` que sirve para especificar el número máximo de decimales que puede tener un número.

Actividad 9: Tipo de vehículo

Dada la definición de `a09_vehiculo.xsd.txt` escribe otro modo de definir el elemento “vehiculo” y un `xs:simpleType` llamado `tipoDeVehiculo` que restringiese a “barco”, “bicicleta”, “coche” y “tren” como los únicos valores aceptables para el vehículo, de forma que dicho tipo pudiera ser también utilizado por otros elementos.

Actividad 10: Iniciales de personas famosas

Dado el archivo `a10_iniciales.xsd` corregir los errores cometidos en `a10_iniciales.xml` para que sea válido.

Actividad 11: Iniciales al revés

Realizar los cambios necesarios en `a11_inicialesAlReves.xsd`, para que `a11_inicialesAlReves.xml` sea válido.

! Hay que tener en cuenta que el elemento `inicialesAlReves` debe ser del mismo tipo que `iniciales`, y dicho tipo solamente deberá definirse una vez.

Pregunta 3: Si se quisiera permitir que las tres letras de iniciales e `inicialesAlReves`, sean minúsculas o mayúsculas indistintamente; en lugar de `[A-Z][A-Z][A-Z]` ¿qué se podría escribir?

Actividad 12: Respuestas admitidas

En `a12_respuestasAdmitidas.xsd.txt` se define un elemento llamado “respuesta” con la restricción de que el único valor aceptable es una de las siguientes letras: “A”, “B”, “C” o “D”. Escribe otra forma de especificar la misma restricción.

Pregunta 4: Sin hacer uso de `xs:pattern`, ¿de qué otro modo podríamos especificar lo mismo que lo expresado con `<xs:pattern value="[ABCD]"/>` ?

Actividad 13: Números y letras

Escribir el contenido del archivo `a13_numerosYLetras.xsd` que permita validar `a13_numerosYLetras.xml`, teniendo en cuenta que:

- Tanto el atributo `numero` como el elemento “código” utilizan la misma restricción que solamente les permite tomar un valor entero expresado con dos dígitos comprendidos entre “00” y “19”.
- El atributo `letra` puede tomar por valor una de las siguientes letras: “Y” o “Z”. - La restricción debe definirse de forma que solamente pueda ser utilizada por dicho atributo.
- Para cada ficha se tiene que indicar un número, obligatoriamente. Sin embargo, la letra es opcional.

Actividad 14: (Voluntaria) Escribir expresiones regulares

En las expresiones regulares se pueden utilizar –entre otros– los siguientes símbolos:

Símbolos	Significado
<code>.</code>	Cualquier carácter.
<code>\d</code>	Cualquier dígito del <code>0</code> al <code>9</code> .
<code>\D</code>	Cualquier carácter que no sea un dígito del <code>0</code> al <code>9</code> .
<code>x*</code>	<code>x</code> puede aparecer cero o más veces.
<code>x+</code>	<code>x</code> debe aparecer al menos una vez.
<code>x?</code>	<code>x</code> puede aparecer una vez o no aparecer.
<code>[abc]</code> o <code>[a\b\ c]</code>	Cualquier carácter indicado entre los corchetes: <code>a</code> , <code>b</code> o <code>c</code> .
<code>[a-z]</code>	Cualquier carácter de la <code>a</code> a la <code>z</code> .
<code>x{n}</code>	<code>x</code> debe aparecer <code>n</code> veces.
<code>x{n,m}</code>	<code>x</code> debe aparecer entre <code>n</code> y <code>m</code> veces.
<code>x{n,}</code>	<code>x</code> debe aparecer al menos <code>n</code> veces.

Teniendo en cuenta, solamente, los símbolos mostrados en la tabla anterior, escribir en `a14_expresionesRegulares.txt` las posibles expresiones regulares que permitan representar los siguientes valores:

1. “Capítulo 0”, “Capítulo 1”, “Capítulo 2”... “Capítulo 9”. (Solo se permite un dígito).
2. “Capítulo 0”, “Capítulo 1”, “Capítulo 2”... “Capítulo 99”. (Uno o dos dígitos).
3. “Capítulo 1”, “Capítulo 2”, “Capítulo 3”... “Capítulo 99”. (No se permite “Capítulo 0”).
4. “Capítulo 0”, “Capítulo 1”, “Capítulo 2”... “Capítulo 99”... “Capítulo 100”... (Uno o más dígitos).
5. Cualquier valor de dos caracteres, cuyo primer carácter sea distinto de un dígito (0-9) y cuyo segundo carácter sea “Z”:
“aZ”... “zZ”, “AZ”... “ZZ”, “?Z”, “=Z”, “*Z”...
6. “ABBC”, “ABBBC”, “ABBBBC”, “ABBBBBBC”.
7. El primer carácter debe ser “R”. A continuación, deben aparecer obligatoriamente dos o más “S”. Finalmente, puede aparecer o no, un dígito del 3 al 8: “RSS”, “RSSS”... “RSS3”... “RSS8”, “RSSS3”... “RSSS8”... “RSSSSSSSSSS7”...
8. Cualquier valor que contenga en primer lugar “COD”, después tres dígitos (0-9) y, finalmente, uno o más caracteres cualesquiera: “COD645pera”, “COD646manzana”...

Dado el documento XML `a14_expresionesRegulares.xml` escribir el contenido del archivo `a14_expresionesRegulares.xsd` que permita validarlo utilizando las expresiones regulares escritas en este ejercicio.

Actividad 15: Letras admitidas

En `a15_letrasAdmitidas.xsd.txt` se define un elemento llamado “letras” con la restricción de que puede tomar por valor cero o más (*) letras minúsculas de la “a” a la “z”:

! Los paréntesis de la expresión regular se pueden omitir, escribiendo simplemente: `[a-z]*`

Realizar los cambios necesarios en el código del ejemplo anterior para que “letras” pueda tomar por valor uno o más pares (+) de letras, y cada par de letras deberá estar formado por una letra mayúscula seguida de otra minúscula. Por ejemplo, “HoLa” sería admitido, pero no lo sería “Hola”, “HOLA”, “hola”, etc.

Actividad 16: Longitud fija de una clave

En `a16_longitudFija.xsd.txt` definir un elemento “clave” que pueda tomar por valor exactamente diez caracteres, los cuales podrán ser letras mayúsculas o minúsculas de la “a” a la “z”, o dígitos del “0” al “9”. Por ejemplo, serán válidos los valores siguientes: “abcde12345”, “Clave55ABC”, “1A2b3c4D5f”, etc.

Actividad 17: Longitud mínima y máxima de una clave

Dado el documento `a17_longitudMinMax.xml` escribir el contenido del archivo `a17_longitudMinMax.xsd` que permita validarlo, teniendo en cuenta que el elemento “clave” debe poder tomar por valor un mínimo de cuatro caracteres y un máximo de diez. Dichos caracteres pueden ser indistintamente letras mayúsculas o minúsculas de la “a” a la “z”, o dígitos del “0” al “9”. La restricción solamente podrá aplicarse al elemento “clave”.

Para ello, se debe utilizar `xs:pattern` y también:

- `xs:minLength` que permite especificar la longitud mínima.
- `xs:maxLength` que permite especificar la longitud máxima.

Actividad 18: Información de persona ampliada

Dado el archivo `a18_infoPersona.xsd` añadir la definición de un nuevo elemento `complexType` llamado “infoPersonaAmpliada2” que amplíe la definición de “infoPersonaAmpliada”, permitiendo validar el documento `a18_infoPersona.xml`.

Actividad 19: Precios de artículos

Realizar los cambios necesarios en el archivo `a19_preciosArticulos.xsd` para que permita validar el documento XML `a19_preciosArticulos.xml`.

Actividad 20: Información de ubicaciones

Añadir, al archivo `a20_infoUbicaciones.xsd`, la definición de un nuevo elemento `complexType` llamado “infoUbicacion” que amplíe la definición de “direccion”, permitiendo validar el documento `a20_infoUbicaciones.xml`.

Actividad 21: Colores de muebles

Haciendo uso del siguiente código:

```
</> XML
1  <xs:complexType name="tipoColorMueble">
2    <xs:simpleContent>
3      <xs:extension base="tipoMueble">
4        <xs:attribute name="color">
5          <xs:simpleType>
6            <xs:restriction base="xs:string">
7              <xs:enumeration value="blanco" />
8              <xs:enumeration value="gris" />
9              <xs:enumeration value="negro" />
10             <xs:enumeration value="azul" />
11            </xs:restriction>
12          </xs:simpleType>
13        </xs:attribute>
14      </xs:extension>
15    </xs:simpleContent>
16  </xs:complexType>
17
18  <xs:simpleType name="tipoMueble">
19    <xs:restriction base="xs:string">
20      <xs:enumeration value="armario" />
21      <xs:enumeration value="mesa" />
22      <xs:enumeration value="silla" />
23    </xs:restriction>
24  </xs:simpleType>
```

Escribir el contenido del archivo `a21_coloresMuebles.xsd` que permita validar el documento `a21_coloresMuebles.xml`.

Actividad 22: Números del bingo

Escribir el código de un documento XML en `a22_numerosBingo.xml` que pueda ser validado por `a22_numerosBingo.xsd` y almacene los números 17, 23 y 65.

i `xs:positiveInteger` es un tipo de dato predefinido derivado, que admite números enteros positivos mayores que cero.

Actividad 23: Información de personas en contenido mixto

Utilizando los elementos “nombre”, “ciudad” y “edad”, escribir el código de un documento XML (`a23_infoPersonasMixto.xml`) que pueda ser validado por `a23_infoPersonasMixto.xsd` y que almacene la siguiente información:

- “Eva vive en París y tiene 25 años.”
- “Giovani vive en Florencia y tiene 25 años.”

Actividad 24: Panel de vuelos

Se ha escrito el documento XML `a24_panelVuelos.xml` para representar la siguiente información ficticia:

Código	Diario	Origen	Destino	Hora salida	Hora llegada	Estado
--------	--------	--------	---------	-------------	--------------	--------

V22	SI	New York	Chicago	9:35	11:35	R
V23	NO	New York	Miami	10:15	11:15	C

Escribir el código del archivo **a24_panelVuelos.xsd** que permita validarlo, teniendo en cuenta que:

- No debe utilizarse ni `group` ni `attributeGroup`.
- El `nombre` del aeropuerto, los `vuelos` y la `fecha` pueden aparecer en distinto orden.
- Se tiene que indicar que el `código` ha de ser único (esto se puede hacer definiéndolo de tipo `xs:ID`) y obligatorio para cada vuelo.
- Haciendo uso `pattern` indicar que los posibles estados de un vuelo son `C` (Cancelado), `E` (En hora), `R` (Retrasado). Dicha restricción sólo debe poder ser utilizada por el atributo `estado`. El valor por defecto debe ser `E`.
- Debe permitirse aparecer desde cero hasta ilimitados elementos `vuelo` y, para cada uno de ellos, se tiene que guardar la información en el mismo orden en el que aparece en el panel.
- Para indicar si un vuelo es `diario`, se debe utilizar un elemento vacío que, respecto a cada vuelo, podrá aparecer (en el caso de sí ser diario) o no aparecer (en el caso contrario).
- Respecto a los elementos `nombre`, `origen`, `destino`, `horaLlegada`, `horaSalida` y `fecha`, cada uno de ellos debe definirse del tipo que se considere más apropiado, de entre los proporcionados por XML Schema.

Actividad 25: Factura

Se ha escrito el documento XML **a25_factura.xml** para representar la información contenida en la siguiente factura ficticia:

FACTURA NÚMERO 27 – FECHA: 18/12/2013:

DATOS EMISOR:	DATOS CLIENTE:
Librería García	Biblioteca Txantrea
CIF: 44555666B	CIF: 33111222A
Teléfono: 777888999	Teléfono: 333999444

Detalle de la factura:

CÓDIGO-ARTÍCULO	TIPO	DESCRIPCIÓN	CANTIDAD	OFERTA	PVP
AW7	Libro	Analítica Web 2.0	1	SI	35€
CP5	Varios	Curso de HTML	2	NO	40€
IMPORTE:					95€

Escribir el código del archivo **a25_factura.xsd** que permita validarlo, teniendo en cuenta que:

- Exceptuando los elementos `datosEmisor`, `datosCliente` y `detalleFactura`, que no tienen porqué aparecer en este orden, el resto de elementos representados en el documento XML, sí deben escribirse obligatoriamente en el orden en el que aparecen.
- Excepto para los hijos directos del elemento `factura`, siempre que sea posible agrupar al menos dos elementos o dos atributos, se debe usar `group` o `attributeGroup`, respectivamente.
- Respecto al número de la `factura` (que debe ser un valor entero mayor que 0) y su `fecha` de emisión (de tipo `xs:date`),

hay que indicar que son atributos obligatorios.

- El atributo `moneda` debe indicarse que es un valor fijo.
- Los `nombres` del emisor y cliente, así como, la `descripción` de cada artículo, deben ser del mismo tipo, al que llamaremos `tipoTexto`, y donde debe indicarse que los valores admitidos para dichos elementos pueden ser cadenas de un mínimo de 5 caracteres y un máximo de 20.
- Haciendo uso `pattern` hay que indicar que el valor del `cif` debe estar formado por una cadena de ocho dígitos del 0 al 9, seguidos de un guión “-” y una letra mayúscula de la “A” a la “Z”. Dicha restricción sólo debe poder ser utilizada por el elemento `cif`.
- Haciendo uso `pattern` hay que indicar que el valor del `teléfono` debe estar formado por una cadena de nueve dígitos del 0 al 9. Dicha restricción sólo debe poder ser utilizada por el `teléfono`.
- Al menos tiene que aparecer una `línea` de detalle y como máximo 15.
- El `importe` debe indicarse que es obligatorio.
- El `importe` y el `pvp` deben ser del mismo tipo, al que llamaremos `tipoPrecio`, y donde debe indicarse, sin hacer uso de `pattern`, que los valores admitidos por este tipo pueden ser números decimales mayores que 0, pero no mayores que 999. Además, dichos valores podrán contener cinco dígitos como máximo y, de ellos, sólo dos podrán ser decimales.
- El `código` del `artículo` ha de ser único y obligatorio para cada artículo.
- Sin hacer uso `pattern` indicar que los posibles tipos de un artículo son `Libro`, `DVD` o `Varios`, no permitiéndose otro valor. Para ello, se debe definir un tipo de dato llamado `tipoArtículo`, que debe poder ser utilizado por otros atributos o elementos. Ahora bien, hay que tener en cuenta que este atributo es opcional.
- La `cantidad` de artículos indicada en cada línea, debe ser un valor entero mayor que 0.
- Para indicar si un artículo está de `oferta`, se debe utilizar un elemento vacío que, respecto a cada artículo, podrá aparecer (en el caso de sí estar de oferta) o no aparecer (en el caso contrario).
- No hay que definir más tipos de datos que los especificados en el ejercicio: `tipoTexto`, `tipoArtículo` y `tipoPrecio`.

Actividad 26: Registro de conexiones

Se ha escrito el documento XML `a26_registroConexiones.xml` para representar la siguiente información ficticia:

REGISTRO DE CONEXIONES DE USUARIOS Y EMPLEADOS DE UNA EMPRESA

USUARIOS:

IDENTIFICADOR	NOMBRE APELLIDOS EMAIL	CONEXIONES (FECHA HORA TIEMPO)
U123	Ana&Sanz Tapia&asanz@jmail.com	2014-02-23&19:15:40&122 2014-02-23&20:30:22&617 2014-02-24&11:18:
U96	Pedro&Ruiz Hierro&pruiz@jotmail.com	2014-02-25&20:33:55&390

EMPLEADOS:

IDENTIFICADOR	NOMBRE APELLIDOS EMAIL DEPARTAMENTO	CONEXIONES (FECHA HORA TIEMPO)
E4	Marta&Vera Gil&mvera@yajoo.es&Marketing	(Ninguna)

Escribir el código del archivo `a26_registroConexiones.xsd` que permita validarlo, teniendo en cuenta que:

- Todos los elementos y atributos son obligatorios, a menos que se indique lo contrario.
- Siempre que sea posible agrupar al menos dos elementos o dos atributos, se debe utilizar `group` o `attributeGroup`,

respectivamente.

- Pueden aparecer de cero a ilimitados `usuarios` y, a continuación, de cero a ilimitados `empleados`.
- `Usuario` debe ser de un tipo definido por nosotros al que llamaremos `tipoPersona`, donde hay que definir los elementos `apellidosYNombre`, `email` y `conexiones`. Por otro lado, empleado ha de ser de otro tipo llamado `tipoPersonaAmpliado`, definido como una extensión de `tipoPersona` añadiéndole el elemento `departamento`. El orden en que tienen que aparecer los elementos hijo de `usuario` y `empleado`, debe ser obligatoriamente el escrito en el documento XML.
- El valor del `identificador` debe ser una cadena formada por una letra “U” o “E” mayúscula, seguida de uno a cinco dígitos del 0 al 9.
- El valor del elemento `apellidosYNombre` debe ser una cadena de entre 1 a 30 caracteres (de la “a” a la “z”, mayúsculas o minúsculas, o el carácter espacio en blanco) para los apellidos, seguida del carácter coma “,” y de entre 1 a otras 20 letras (de la “a” a la “z”, también mayúsculas o minúsculas, o el carácter espacio en blanco) para el nombre.
- El valor del email puede ser una cadena formada por 1 a 15 caracteres de la “a” a la “z”, seguida del carácter “@”, seguido de entre 1 a otras 25 letras de la “a” a la “z”, seguidas del carácter punto “.” y de entre otras 2 a 4 letras de la “a” a la “z”.
- De cada usuario y empleado se reflejan sus `conexiones`, indicando para cada `conexión` la cantidad de segundos que duró, que debe ser un número entero mayor que cero. Hay que tener en cuenta que, como se puede ver en el documento XML, pueden aparecer desde cero hasta ilimitados elementos `conexión`.
- Respecto a los atributos `fecha` y `hora`, cada uno de ellos debe definirse del tipo que se considere más apropiado, de entre los proporcionados por XML Schema.
- Los posibles `departamentos` de la empresa a los que puede pertenecer un `empleado` son `administración`, `informática` o `marketing`. De tal forma que, para cada empleado, sólo uno de ellos debe escribirse en el documento XML mediante un elemento vacío, como en este caso se ha escrito `<marketing />`.
- No hay que definir más tipos de datos que los indicados en el ejercicio: `tipoPersona` y `tipoPersonaAmpliado`.

Actividad 27: (Voluntaria) Personal de departamentos

Se ha escrito el documento `a27_personalDepartamentos.xml` para representar la siguiente información ficticia:



Figura 1: Información del personal de los departamentos

Escribir el código del archivo `a27_personalDepartamentos.xsd` que permita validarlo, teniendo en cuenta que:

- Todos los elementos y atributos son obligatorios, a menos que se indique lo contrario.
- Los elementos `datosGenerales` y `datosDepartamentos` pueden aparecer indistintamente uno antes que el otro.
- Excepto para los hijos directos de los elementos `personal`, `datosGenerales` y `departamento`, siempre que sea posible agrupar al menos dos elementos o dos atributos, se debe utilizar `group` o `attributeGroup`.
- Los datos generales de la empresa deben ser de un tipo definido por nosotros al que llamaremos `tipoDatosGenerales`, donde hay que definir los elementos `nombreEmpresa`, `númeroTrabajadores` (que debe ser un valor entero mayor que 0) y `sector`. Estos elementos deben escribirse en dicho orden en el documento XML.
- El atributo `fecha` debe definirse del tipo que se considere más apropiado, de entre los proporcionados por XML Schema.
- El atributo `multinacional` indica si la empresa lo es, o no, con un valor lógico.
- El atributo `moneda` debe indicarse que es un valor fijo. Pero, no es obligatorio.
- El elemento `nombreEmpresa` y el elemento `nombreYApellidos` de los trabajadores, deben ser del mismo tipo, al que llamaremos `tipoTexto`, y donde debe indicarse que los valores admitidos para dichos elementos pueden ser cadenas de un mínimo de 1 carácter y un máximo de 40.
- Los posibles `sectores` son `educación`, `finanzas` o `tecnología`. De tal forma que, sólo uno de ellos debe escribirse en el


archivo XML mediante un elemento vacío, como en este caso se ha escrito `<tecnología />` .

- Se tiene que indicar que el `código` de cada departamento ha de ser único.
- Sin hacer uso `pattern` indicar que los posibles `nombres de departamento` son `Administración` , `Informática` , `Marketing` o `Recursos humanos` , no permitiéndose otro valor. Para ello, se debe definir un tipo de dato llamado `tipoDepartamento` , que debe poder ser utilizado por otros atributos o elementos.
- `Empleado` (en cada departamento puede haber de 0 a 3) debe ser de un tipo definido por nosotros al que llamaremos `tipoEmpleado` , donde hay que definir los posibles valores que pueden tener los elementos `nombreYApellidos` , `baja` y `salario` (que deberán escribirse en ese orden en el documento XML). Por otro lado, jefe (obligatoriamente habrá 1 por departamento) ha de ser de otro tipo llamado `tipoJefe` , definido como una extensión de `tipoEmpleado` añadiéndole el elemento clave.
- De cada `departamento` (pueden haber de 1 a ilimitados), primero debe escribirse el `jefe` y, después, los `empleados` que hubiese.
- Para indicar si un trabajador está de `baja` , se debe utilizar un elemento vacío, que podrá aparecer (en el caso de que sí esté de baja) o no aparecer (en el caso contrario).
- Sin hacer de uso `pattern` hay que indicar que el valor del `salario` debe ser un número decimal mayor que 1000, pero no mayor que 9999. Además, dicho valor podrá contener 6 dígitos como máximo y, de ellos, sólo dos podrán ser decimales.
- La `clave` debe ser de un tipo definido por nosotros al que llamaremos `tipoClave` , y donde debe indicarse, que los valores admitidos por este tipo pueden ser cadenas de ocho caracteres donde el primero debe ser un dígito del 0 al 9; el segundo debe ser un carácter distinto a un dígito; después, pueden aparecer de 2 a 4 letras de la “a” a la “z”; posteriormente, podrá aparecer, o no, una letra de la “A” a la “Z”; a continuación, tendrá que estar una de estas tres letras mayúsculas (K, Y, H); y finalmente, podrá aparecer de 0 a 3 caracteres cualesquiera.
- `Usuario` no es un atributo obligatorio. Ahora bien, si se escribe, debe estar formado por un mínimo de 6 caracteres y un máximo de 8 (hay que escribir esta restricción sin hacer uso de `pattern`). Por otro lado, se debe indicar “invitado” como su valor por defecto.
- No hay que definir en el schema más tipos de datos que los indicados en el ejercicio: `tipoDatosGenerales` , `tipoDepartamento` , `tipoEmpleado` , `tipoJefe` , `tipoTexto` y `tipoClave` .

Bibliografía

- <https://www.abrirllave.com/xsd/ejercicios-resueltos.php>

 [Desarrollo de Aplicaciones Multiplataforma](#) , [Lenguajes de Marcas y Sistemas de Gestión de Información](#)

 [desarrollo de aplicaciones multiplataforma](#) [lenguajes de marcas y sistemas de gestión de información](#)
[administración de sistemas informáticos de red](#) [práctica](#) [tarea](#) [dam](#) [daw](#) [asir](#)

Otros artículos

[hace 5 meses](#)

Tarea: Introducción a XML

[Entrega y presentación](#) La entrega será en formato PDF. [Leer Entrega y](#)

[hace 4 meses](#)

Tarea: El formato SVG

[Entrega y presentación](#) La entrega será en formato SVG. [Calificación](#) La tarea se

[hace 4 meses](#)

Tarea: Calculadora con HTML y CSS

[Entrega y presentación](#) La entrega será en formato ZIP. [Leer Entrega y](#)

MÁS VIEJO
Tarea: DTD

MÁS NUEVO
Tarea: Pandas, KNN y Regresión Lineal

0 reacciones



0 comentarios

Escribir

Previsualizar

Aa

Iniciar sesión para comentar

Iniciar sesión con GitHub