Práctica: XML y XSD

Lenguaje de Marcas y Sistemas de Gestión (LMSG)

Marcos Ruiz García

25 de febrero de 2021

Índice

1. Validación de un documento XML con XSD **3**
1.1. Validar un documento XML . . . 3
1.2. Validar un sitemap XML . . . 4
1.3. Mensaje entre personas . . . 5

2. Elementos simples **6**
2.1. Definición de elementos simples . . . 6
2.2. Puerta cerrada y ventana abierta . . . 6

3. Atributos **6**
3.1. Fichas de personas . . . 6

4. Restricciones **7**
4.1. Edad entre 0 y 130 años . . . 7
4.2. Precios de tres dígitos . . . 7
4.3. Tipo de vehículo . . . 8
4.4. Iniciales de personas famosas . . . 8
4.5. Iniciales al revés . . . 9
4.6. Respuestas admitidas . . . 10
4.7. Números y letras . . . 11
4.8. Escribir expresiones regulares . . . 11
4.9. Letras admitidas . . . 12
4.10. Longitud fija de una clave . . . 13
4.11. Longitud mínima y máxima de una clave . . . 13

5. Extensiones **13**
5.1. Información de persona ampliada . . . 13
5.2. Precios de artículos . . . 15

5.3. Información de ubicaciones 16
5.4. Colores de muebles 17

6. Elementos complejos **18**
6.1. Números del bingo 18
6.2. Información de personas en contenido mixto 19

7. Indicadores **19**
7.1. Panel de vuelos 19
7.2. Factura 20
7.3. Registro de conexiones 23
7.4. Personal de departamentos 25

8. Entrega **28**

Introducción

Tabla 1: Información básica de la práctica a realizar

Duración	- horas
Título	XML y XSD
Unidad Didáctica	
Módulo	Lenguaje de Marcas y Sistemas de Gestión (LMSG)
Ciclo Formativo	Desarrollo de Aplicaciones Multiplataforma (DAM)
Autor	Marcos Ruiz García

1. Validación de un documento XML con XSD

1.1. Validar un documento XML

Dado el documento `marcadores.xml` (Código 1) y el archivo `marcadores.xsd` (Código 2), comprobar con XML Copy Editor que `marcadores.xml` es válido.

Código 1: Archivo `marcadores.xml`

```
<?xml version="1.0" encoding="UTF-8"?>
<marcadores xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:noNamespaceSchemaLocation="marcadores.xsd">
  <pagina>
    <nombre>Abrirllave</nombre>
    <descripcion>Tutoriales de informática.</descripcion>
    <url>http://www.abrirllave.com/</url>
  </pagina>
  <pagina>
    <nombre>Wikipedia</nombre>
    <descripcion>La enciclopedia libre.</descripcion>
    <url>http://www.wikipedia.org/</url>
  </pagina>
  <pagina>
    <nombre>W3C</nombre>
    <descripcion>World Wide Web Consortium.</descripcion>
    <url>http://www.w3.org/</url>
  </pagina>
</marcadores>
```

Código 2: Archivo `marcadores.xsd`

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="marcadores">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="pagina" maxOccurs="unbounded">
          <xs:complexType>
```

```
            <xs:sequence>
              <xs:element name="nombre" type="xs:string"/>
              <xs:element name="descripcion" type="xs:string"/>
              <xs:element name="url" type="xs:string"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

1.2. Validar un sitemap XML

Dado el siguiente `sitemap.xml` (Código 3), comprobar con un validador online de sitemaps XML que es válido.

Código 3: Archivo `sitemap.xml`

```
<?xml version="1.0" encoding="UTF-8"?>
<urlset xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.sitemaps.org/schemas/sitemap/0.9
           http://www.sitemaps.org/schemas/sitemap/0.9/sitemap.xsd"
        xmlns="http://www.sitemaps.org/schemas/sitemap/0.9">
  <url>
    <loc>http://www.ejemplos-de-abrirllave.com/</loc>
    <lastmod>2016-09-30</lastmod>
    <priority>0.8</priority>
  </url>
  <url>
    <loc>http://www.ejemplos-de-abrirllave.com/contactar.html</loc>
    <lastmod>2016-09-30</lastmod>
    <priority>0.3</priority>
  </url>
  <url>
    <loc>http://www.ejemplos-de-abrirllave.com/productos/impresora.html</
       loc>
    <lastmod>2016-09-30</lastmod>
    <priority>0.5</priority>
  </url>
  <url>
    <loc>http://www.ejemplos-de-abrirllave.com/productos/monitor.html</loc>
    <lastmod>2016-09-30</lastmod>
    <priority>0.5</priority>
  </url>
  <url>
    <loc>http://www.ejemplos-de-abrirllave.com/productos/teclado.html</loc>
    <lastmod>2016-09-30</lastmod>
    <priority>0.5</priority>
  </url>
</urlset>
```

1.3. Mensaje entre personas

Dado el siguiente archivo `mensaje.xsd` (Código 4). Nótese que, en la definición del elemento `mensaje`, se ha referenciado al elemento `asunto` utilizando el atributo `ref`), corregir los errores cometidos en el documento XML (Código 5), para que sea válido. Para ello, no modificar la cuarta línea de código.

Código 4: Archivo `mensaje.xsd`

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           targetNamespace="http://www.abrirllave.com/mensaje"
           xmlns:me="http://www.abrirllave.com/mensaje">

  <xs:element name="mensaje">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="origen" type="me:datosOrigenDestino"/>
        <xs:element name="destino" type="me:datosOrigenDestino"/>
        <xs:element ref="me:asunto"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:element name="asunto" type="xs:string"/>

  <xs:complexType name="datosOrigenDestino">
    <xs:sequence>
      <xs:element name="nombre" type="xs:string"/>
      <xs:element name="ciudad" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>

</xs:schema>
```

Código 5: Archivo `mensaje.xml`

```
<?xml version="1.0"?>
<mensaje xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.abrirllave.com/mensaje mensaje.xsd"
         xmlns:men="http://www.abrirllave.com/mensaje">
  <origen>
    <nombre>Ana Sanz Tin</nombre>
    <ciudad>Pamplona</ciudad>
  </origen>
  <destino>
    <nombre>Iker Rubio Mol</nombre>
    <ciudad>Sevilla</ciudad>
  </destino>
  <asunto>¡Vuelve pronto!</asunto>
</mensaje>
```

2. Elementos simples

2.1. Definición de elementos simples

Para los elementos del Código 6, escribir sus definiciones de elementos simples correspondientes.

Código 6: Elementos simples

```
<ciudad>Roma</ciudad>
<fecha-de-nacimiento>1996-12-18</fecha-de-nacimiento>
<hora>18:29:45</hora>
<nota>7.5</nota>
<apto>true</apto>
```

2.2. Puerta cerrada y ventana abierta

Definir 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`.

3. Atributos

3.1. Fichas de personas

Dado el documento XML del Código 7, escribir el contenido del archivo `fichas.xsd` que permita validarlo.

Código 7: Archivo `fichas.xml`

```
<?xml version="1.0" encoding="UTF-8"?>
<fichas xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:noNamespaceSchemaLocation="fichas.xsd">
  <ficha numero="1">
    <nombre>Ana Sanz Tin</nombre>
    <edad>22</edad>
  </ficha>
  <ficha numero="2">
    <nombre>Iker Rubio Mol</nombre>
    <edad>23</edad>
  </ficha>
</fichas>
```

4. Restricciones

4.1. Edad entre 0 y 130 años

Dado el siguiente documento XML del Código 8. Escribir el contenido del archivo `fichas.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.

Código 8: Archivo `edad.xml`

```
<?xml version="1.0" encoding="UTF-8"?>
<fichas xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:noNamespaceSchemaLocation="edad.xml">
  <ficha numero="1">
    <nombre>Ana Sanz Tin</nombre>
    <edad>22</edad>
  </ficha>
  <ficha numero="2">
    <nombre>Iker Rubio Mol</nombre>
    <edad>23</edad>
  </ficha>
</fichas>
```

4.2. Precios de tres dígitos

Dado el siguiente documento XML (Código 9). Escribir el contenido del archivo "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.

Código 9: Archivo `precios.xml`

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

```
  <precio>2.6</precio>
  <precio>4.95</precio>
  <precio>187</precio>
</precios>
```

4.3. Tipo de vehículo

Dada la definición del Código 15. ¿De qué otro modo se puede definir el elemento `vehículo` y un `xs:simleType` 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?

Código 10: Definición

```
<xs:element name="vehiculo">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:enumeration value="barco"/>
      <xs:enumeration value="bicicleta"/>
      <xs:enumeration value="coche"/>
      <xs:enumeration value="tren"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

4.4. Iniciales de personas famosas

Dado el archivo `iniciales.xsd` (Código 11). Corregid los errores cometidos en el documento XML del Código 12 para que sea válido.

Código 11: Archivo `iniciales.xsd`

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="fichas">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="ficha" maxOccurs="unbounded">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="nombre" type="xs:string"/>
              <xs:element name="iniciales">
                <xs:simpleType>
                  <xs:restriction base="xs:string">
                    <xs:pattern value="[A-Z][A-Z][A-Z]"/>
                  </xs:restriction>
                </xs:simpleType>
              </xs:element>
              <xs:element name="edad" type="xs:integer"/>
            </xs:sequence>
```

```
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Código 12: Archivo `iniciales.xml`

```
<?xml version="1.0" encoding="UTF-8"?>
<fichas xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:noNamespaceSchemaLocation="iniciales.xsd">
  <ficha>
    <nombre>Antonio Machado Ruiz</nombre>
    <iniciales>AMR</iniciales>
    <edad>22</edad>
  </ficha>
  <ficha>
    <nombre>Mario Moreno</nombre>
    <iniciales>MM</iniciales>
    <edad>23</edad>
  </ficha>
  <ficha>
    <iniciales>ALO</iniciales>
    <nombre>Ada Lovelace</nombre>
    <edad>24</edad>
  </ficha>
  <ficha>
    <nombre>pablo ruiz picasso</nombre>
    <iniciales>prp</iniciales>
    <edad>24</edad>
  </ficha>
</fichas>
```

4.5. Iniciales al revés

Dado el archivo `inicialesAlReves.xml` (Código 11).Realizar los cambios necesarios en el documento XSD del Código 13 para que sea válido.

Código 13: Archivo `inicialesAlReves.xsd`

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="fichas">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="ficha" maxOccurs="unbounded">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="nombre" type="xs:string"/>
              <xs:element name="iniciales">
```

```
                <xs:simpleType>
                  <xs:restriction base="xs:string">
                    <xs:pattern value="[A-Z][A-Z][A-Z]"/>
                  </xs:restriction>
                </xs:simpleType>
              </xs:element>
              <xs:element name="edad" type="xs:integer"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Código 14: Archivo `inicialesAlReves.xml`

```
<?xml version="1.0" encoding="UTF-8"?>
<fichas xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:noNamespaceSchemaLocation="inicialesAlReves.xsd">
  <ficha>
    <nombre>Ana Sanz Tin</nombre>
    <iniciales>AST</iniciales>
    <edad>22</edad>
    <iniciales-al-reves>TSA</iniciales-al-reves>
  </ficha>
  <ficha>
    <nombre>Iker Rubio Mol</nombre>
    <iniciales>IRM</iniciales>
    <edad>23</edad>
    <iniciales-al-reves>MRI</iniciales-al-reves>
  </ficha>
</fichas>
```

4.6. Respuestas admitidas

En el siguiente ejemplo se define un elemento llamado `respuesta` con la restricción de que el único valor aceptable es una de las siguientes letras: `A`, `B`, `C`, `D`, `E`.

- En vez de escribiendo `[ABCDE]`, ¿de qué otras formas se podría especificar la misma restricción?.
- Sin hacer uso de `xs:pattern`, ¿de qué otro modo podríamos especificar lo mismo que lo expresado con `<xs:pattern value="[ABCDE]/>`?

Código 15: Definición

```
<xs:element name="respuesta">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:pattern value="[ABCDE]"/>
```

```
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

4.7. Números y letras

Dado el siguiente documento XML (Código 16). Escribir el contenido del archivo `numerosYLetras.xsd` que permita validarlo, 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: `X`, `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.

Código 16: Archivo `numerosYLetras.xml`

```
<?xml version="1.0" encoding="UTF-8"?>
<fichas xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:noNamespaceSchemaLocation="numerosYLetras.xsd">
  <ficha numero="01" letra="Z">
    <codigo>11</codigo>
    <nombre>Ana Sanz Tin</nombre>
  </ficha>
  <ficha numero="02">
    <codigo>12</codigo>
    <nombre>Iker Rubio Mol</nombre>
  </ficha>
</fichas>
```

4.8. Escribir expresiones regulares

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

Teniendo en cuenta, solamente, los símbolos mostrados en la tabla anterior, escribir 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).

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

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`, `ABBBBBC`.
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`...

4.9. Letras admitidas

En el siguiente ejemplo 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" (Código 17).

Nota: 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.

Código 17: Definición

```
<xs:element name="letras">
  <xs:simpleType>
    <xs:restriction base="xs:string">
```

```
      <xs:pattern value="([a-z])*"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

4.10. Longitud fija de una clave

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.

4.11. Longitud mínima y máxima de una clave

Dado el siguiente documento XML (Código 18). Escribir el contenido del archivo `fichas.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.

Código 18: Archivo `longitud.xml`

```
<?xml version="1.0" encoding="UTF-8"?>
<fichas xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="longitud.xsd">
   <ficha>
      <nombre>Ana Sanz Tin</nombre>
      <clave>G8w2</clave>
   </ficha>
   <ficha>
      <nombre>Iker Rubio Mol</nombre>
      <clave>ag32Ue7AFF</clave>
   </ficha>
</fichas>
```

5. Extensiones

5.1. Información de persona ampliada

Dado el archivo `fichas.xsd` (Código 19)

Código 19: Archivo `fichas.xsd`

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

  <xs:element name="fichas">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="ficha" type="infoPersonaAmpliada" maxOccurs="
            unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:complexType name="infoPersonaAmpliada">
    <xs:complexContent>
      <xs:extension base="infoPersona">
        <xs:sequence>
          <xs:element name="ciudad" type="xs:string"/>
          <xs:element name="pais" type="xs:string"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <xs:complexType name="infoPersona">
    <xs:sequence>
      <xs:element name="nombre" type="xs:string"/>
      <xs:element name="edad" type="edadPersona"/>
    </xs:sequence>
    <xs:attribute name="numero" type="xs:integer"/>
  </xs:complexType>

  <xs:simpleType name="edadPersona">
     <xs:restriction base="xs:integer">
        <xs:minExclusive value="-1"/>
        <xs:maxExclusive value="131"/>
     </xs:restriction>
  </xs:simpleType>

</xs:schema>
```

Añadir, al archivo `fichas.xsd`, la definición de un nuevo elemento `complexType` llamado `infoPersonaAmpliada2` que amplíe la definición de `infoPersonaAmpliada`, permitiendo validar el siguiente documento XML:

Código 20: Archivo `fichas.xml`

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

  <xs:element name="fichas">
```

```
    <xs:complexType>
      <xs:sequence>
        <xs:element name="ficha" type="infoPersonaAmpliada" maxOccurs="
            unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:complexType name="infoPersonaAmpliada">
    <xs:complexContent>
      <xs:extension base="infoPersona">
        <xs:sequence>
          <xs:element name="ciudad" type="xs:string"/>
          <xs:element name="pais" type="xs:string"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <xs:complexType name="infoPersona">
    <xs:sequence>
      <xs:element name="nombre" type="xs:string"/>
      <xs:element name="edad" type="edadPersona"/>
    </xs:sequence>
    <xs:attribute name="numero" type="xs:integer"/>
  </xs:complexType>

  <xs:simpleType name="edadPersona">
     <xs:restriction base="xs:integer">
        <xs:minExclusive value="-1"/>
        <xs:maxExclusive value="131"/>
     </xs:restriction>
  </xs:simpleType>

</xs:schema>
```

5.2. Precios de artículos

Dado el archivo `precios.xsd`:

Código 21: Archivo `precios.xsd`

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="precios">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="precio" maxOccurs="unbounded">
          <xs:complexType>
            <xs:simpleContent>
              <xs:extension base="xs:decimal">
```

```
                <xs:attribute name="moneda" type="xs:string" />
              </xs:extension>
            </xs:simpleContent>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Realizar los cambios necesarios en el archivo `precios.xsd` para que permita validar el siguiente documento XML:

Código 22: Archivo `precios.xml`

```
<?xml version="1.0" encoding="UTF-8"?>
<articulos xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="articulos.xsd">
   <articulo>
      <nombre>Mesa</nombre>
      <precio moneda="Euro">50</precio>
   </articulo>
   <articulo>
      <nombre>Silla</nombre>
      <precio moneda="Dólar">78.99</precio>
   </articulo>
</articulos>
```

5.3. Información de ubicaciones

Dado el archivo `ubicaciones.xsd`:

Código 23: Archivo `ubicaciones.xsd`

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

  <xs:element name="ubicaciones">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="ubicacion" type="direccion" maxOccurs="unbounded"
           />
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:simpleType name="direccion">
    <xs:restriction base="xs:string">
      <xs:enumeration value="norte"/>
      <xs:enumeration value="sur"/>
      <xs:enumeration value="este"/>
```

```
      <xs:enumeration value="oeste"/>
    </xs:restriction>
  </xs:simpleType>

</xs:schema>
```

Añadir, al archivo `ubicaciones.xsd`, la definición de un nuevo elemento `complexType` llamado `infoUbicacion` que amplíe la definición de `direccion`, permitiendo validar el siguiente documento XML:

Código 24: Archivo `ubicaciones.xml`

```
<?xml version="1.0" encoding="UTF-8"?>
<ubicaciones xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="ubicaciones.xsd">
   <ubicacion metros="32">norte</ubicacion>
   <ubicacion metros="25">este</ubicacion>
   <ubicacion metros="64">este</ubicacion>
</ubicaciones>
```

5.4. Colores de muebles

Haciendo uso del siguiente código:

Código 25: Definición de `tipoColorMueble`

```
<xs:complexType name="tipoColorMueble">
  <xs:simpleContent>
    <xs:extension base="tipoMueble">
      <xs:attribute name="color">
        <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:enumeration value="blanco"/>
          <xs:enumeration value="gris"/>
          <xs:enumeration value="negro"/>
          <xs:enumeration value="wengue"/>
        </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>

<xs:simpleType name="tipoMueble">
  <xs:restriction base="xs:string">
    <xs:enumeration value="armario"/>
    <xs:enumeration value="mesa"/>
    <xs:enumeration value="silla"/>
  </xs:restriction>
</xs:simpleType>
```

Escribir el contenido del archivo `muebles.xsd` que permita validar el siguiente documento XML:

Código 26: Archivo `muebles.xml`

```
<?xml version="1.0" encoding="UTF-8"?>
<muebles xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="muebles.xsd">
   <mueble color="blanco">mesa</mueble>
   <mueble color="gris">silla</mueble>
</muebles>
```

6. Elementos complejos

6.1. Números del bingo

Dado el archivo `bingo.xsd` (Código 27), escribir el código de un documento XML que pueda ser validado por `bingo.xsd` y almacene los números 17, 23 y 65. Teniendo en cuenta que `xs:positiveInteger` es un tipo de dato predefinido derivado, que admite números enteros positivos mayores que cero.

Código 27: Archivo `bingo.xsd`

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

  <xs:element name="bingo">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="bola" maxOccurs="unbounded">
          <xs:complexType>
            <xs:attribute name="numero" type="numeroDeBola"/>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:simpleType name="numeroDeBola">
     <xs:restriction base="xs:positiveInteger">
        <xs:minInclusive value="1"/>
        <xs:maxExclusive value="90"/>
     </xs:restriction>
  </xs:simpleType>

</xs:schema>
```

6.2. Información de personas en contenido mixto

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

- "Eva vive en París y tiene 25 años."
- "Giovanni vive en Florencia y tiene 26 años."

Código 28: Archivo `personas.xsd`

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="personas">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="persona" maxOccurs="unbounded">
          <xs:complexType mixed="true">
            <xs:sequence>
              <xs:element name="nombre" type="xs:string"/>
              <xs:element name="ciudad" type="xs:string"/>
              <xs:element name="edad" type="xs:positiveInteger"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

7. Indicadores

7.1. Panel de vuelos

Si para representar la información de vuelos ficticia (Tabla 2) se ha escrito el documento XML del Código 29. Escribir el código del archivo `aeropuerto.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`, `hora-llegada`, `hora-salida` y `fecha`, cada uno de ellos debe definirse del tipo que se considere más apropiado, de entre los proporcionados por XML Schema.

Tabla 2: Información de vuelos ficticia

Código	Diario	Origen	Destino	Hora salida	Hora llegada	Estado
V22	SI	New York	Chicago	9:30	11:30	R
V23	NO	New York	Miami	10:15	11:15	C

Código 29: Archivo `aeropuerto.xml`

```
<?xml version="1.0" encoding="UTF-8"?>
<aeropuerto xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="aeropuerto.xsd">
   <nombre>JFK</nombre>
   <vuelos>
      <vuelo código="V22" estado="R">
         <diario />
         <origen>New York</origen>
         <destino>Chicago</destino>
         <hora-salida>09:30:00</hora-salida>
         <hora-llegada>11:30:00</hora-llegada>
      </vuelo>
      <vuelo código="V23" estado="C">
         <origen>New York</origen>
         <destino>Miami</destino>
         <hora-salida>10:15:00</hora-salida>
         <hora-llegada>11:15:00</hora-llegada>
      </vuelo>
   </vuelos>
   <fecha>2013-12-20</fecha>
</aeropuerto>
```

7.2. Factura

Si para representar la información contenida en la factura ficticia representada en la Tabla 3.

se ha escrito el siguiente documento XML:

Código 30: Archivo `factura.xml`

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

Tabla 3: Factura ficticia

FACTURA NÚMERO 27 – FECHA: 18/12/2013					
DATOS EMISOR:			**DATOS CLIENTE:**		
Librería Pérez			Biblioteca Txantrea		
CIF: 44555666B			CIF: 33111222A		
Teléfono: 777888999			Teléfono: 333999444		
DETALLE FACTURA:					
CÓDIGO-ARTÍCULO	TIPO	DESCRIPCIÓN	CANTIDAD	OFERTA	PVP
AW7	Libro	Analítica Web 2.0	1	SI	25.12€
CP5	DVD	Curso de HTML	2	NO	30.5€
IMPORTE:					86.12€

```
<factura xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="factura.xsd"
número="27" fecha="2013-12-18" moneda="Euro">
   <datos-emisor>
      <nombre>Librería Pérez</nombre>
      <cif>44555666-B</cif>
      <teléfono>777888999</teléfono>
   </datos-emisor>
   <datos-cliente>
      <nombre>Biblioteca Txantrea</nombre>
      <cif>33111222-A</cif>
      <teléfono>333999444</teléfono>
   </datos-cliente>
   <detalle-factura importe="86.12">
      <línea código-artículo="AW7" tipo="Libro">
         <descripción>Analítica Web 2.0</descripción>
         <cantidad>1</cantidad>
         <oferta />
         <pvp>25.12</pvp>
      </línea>
      <línea código-artículo="CP5" tipo="DVD">
         <descripción>Curso de HTML</descripción>
         <cantidad>2</cantidad>
         <pvp>30.5</pvp>
      </línea>
   </detalle-factura>
</factura>
```

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

- Exceptuando los elementos `datos-emisor`, `datos-cliente` y `detalle-factura`, 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`.

7.3. Registro de conexiones

Si para representar la información de conexiones ficticia de la Tabla 4 se ha escrito el documento XML del Código 31. Escribir el código del archivo `registro.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 `apellidos-y-nombre`, `email` y `conexiones`. Por otro lado, `empleado` ha de ser de otro tipo llamado `tipoPersonaAmpliado`, definido como una extensión de `tipoPersona` añadiéndosele 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 `apellidos-y-nombre` 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`.

Tabla 4: Información de conexiones 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&11:18:31&25
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)

Código 31: Archivo `registro.xml`

```
<?xml version="1.0" encoding="UTF-8"?>
<registro xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="registro.xsd">
  <usuario identificador="U123">
    <apellidos-y-nombre>Sanz Tapia,Ana</apellidos-y-nombre>
    <email>asanz@jmail.com</email>
    <conexiones>
      <conexión fecha="2014-02-23" hora="19:15:40">122</conexión>
      <conexión fecha="2014-02-23" hora="20:30:22">617</conexión>
      <conexión fecha="2014-02-24" hora="11:18:31">25</conexión>
    </conexiones>
  </usuario>
  <usuario identificador="U96">
    <apellidos-y-nombre>Ruiz Hierro,Pedro</apellidos-y-nombre>
    <email>pruiz@jotmail.com</email>
    <conexiones>
      <conexión fecha="2014-02-25" hora="20:33:55">390</conexión>
    </conexiones>
  </usuario>
  <empleado identificador="E4">
    <apellidos-y-nombre>Vera Gil,Marta</apellidos-y-nombre>
    <email>mvera@yajoo.es</email>
    <conexiones/>
    <departamento>
      <marketing />
    </departamento>
  </empleado>
</registro>
```

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

que:

7.4. Personal de departamentos

Si para representar la información ficticia de la Figura 1 se ha escrito el documento XML del Código 32. Escribir el código del archivo `personal.xsd` que permita validarlo, teniendo en cuenta que:

- Todos los elementos y atributos son obligatorios, a menos que se indique lo contrario.
- Los elementos `datos-generales` y `datos-departamentos` pueden aparecer indistintamente uno antes que el otro.
- Excepto para los hijos directos de los elementos `personal`, `datos-generales` 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 `nombre-empresa`, `número-trabajadores` (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 `nombre-empresa` y el elemento `nombre-y-apellidos` 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 `nombre-y-apellidos`, `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`.

Figura 1: Información de personal ficticia

INFORMACIÓN DEL PERSONAL DE LOS DEPARTAMENTOS A FECHA 20-10-2013

NOMBRE: Navarra 4 Internet **NÚMERO DE TRABAJADORES:** 6
MULTINACIONAL: NO **MONEDA:** Euro **SECTOR:** Tecnología

CÓDIGO	DEPARTAMENTO	NOMBRE Y APELLIDOS	BAJA	SALARIO	USUARIO	CLAVE
ADMIN	Administración	Ana Sanz Ruiz (Jefe)	NO	4700.58	anasan	3%abZKi6
		Juan Gil Rus (Empleado)	SI	2200.58		
		Lucas López Tapia (Empleado)	NO	2215.65		
INFOR	Informática	Isabel Gómez Pérez (Jefe)	SI	5200.77	isabelgo	8$abcdHj
		Oscar Lee Blesa (Empleado)	NO	3109.26		
MARKE	Marketing	Luis Mar Herreros (Jefe)	NO	5111.09	invitado	2$xlzY#@

Código 32: Archivo `personal.xml`

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

```
<personal fecha="2013-10-20" moneda="Euro" multinacional="false" xmlns:xsi=
   "http://www.w3.org/2001/XMLSchema-instance"
   xsi:noNamespaceSchemaLocation="personal.xsd">
  <datos-generales>
    <nombre-empresa>Navarra 4 Internet</nombre-empresa>
    <número-trabajadores>6</número-trabajadores>
    <sector>
      <tecnología/>
    </sector>
  </datos-generales>
  <datos-departamentos>
    <departamento código="ADMIN" nombre-departamento="Administración">
      <jefe>
        <nombre-y-apellidos>Ana Sanz Ruiz</nombre-y-apellidos>
        <salario>4700.58</salario>
        <clave usuario="anasan">3%abZKi6</clave>
      </jefe>
      <empleado>
        <nombre-y-apellidos>Juan Gil Rus</nombre-y-apellidos>
        <baja/>
        <salario>2200.58</salario>
      </empleado>
      <empleado>
        <nombre-y-apellidos>Lucas López Tapia</nombre-y-apellidos>
        <salario>2215.65</salario>
      </empleado>
    </departamento>
    <departamento código="INFOR" nombre-departamento="Informática">
      <jefe>
        <nombre-y-apellidos>Isabel Gómez Pérez</nombre-y-apellidos>
        <baja/>
        <salario>5200.77</salario>
        <clave usuario="isabelgo">8$abcdHj</clave>
      </jefe>
      <empleado>
        <nombre-y-apellidos>Oscar Lee Blesa</nombre-y-apellidos>
        <salario>3109.26</salario>
      </empleado>
    </departamento>
    <departamento código="MARKE" nombre-departamento="Marketing">
      <jefe>
        <nombre-y-apellidos>Luis Mar Herreros</nombre-y-apellidos>
        <salario>5111.09</salario>
        <clave>2$xlzY#@</clave>
      </jefe>
    </departamento>
  </datos-departamentos>
</personal>
```

8. Entrega

La entrega de la práctica será un enlace a vuestro repositorio público remoto con la tarea realizada. Este repositorio puede estar en GitHub, Bitbucket, GitLab, etc.

Tras la entrega de la práctica se realizará un test para comprobar que se han adquirido los conocimientos pertinentes y se deberá defender.

El día de la entrega de la práctica será el mismo día del test.

Referencias

Ejercicios resueltos de XSD (XML Schema) | Tutorial de XSD | Abrirllave.com. (2021, Feb). Descargado de `https://www.abrirllave.com/xsd/ejercicios-resueltos.php` ([Online; accessed 19. Feb. 2021])