RELACIÓN 2 - SCHEMAS

1. Define un elemento llamado "edad" con una restricción. El valor de la edad no puede ser menor que 0 o mayor que 120.

```
<xs:element name="edad">
  <xs:simpleType>
    <xs:restriction base="xs:integer">
      <xs:minInclusive value="0"/>

      <xs:maxInclusive value="120"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

2. Define un elemento llamado "coche" con una restricción. Los únicos valores aceptables son: Audi, Golf y BMW.

```
<xs:element name="coche">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:enumeration value="Audi"/>

      <xs:enumeration value="Golf"/>

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

3. Define un elemento llamado "letra" restricción. El único valor aceptable es de tres de las letras mayúsculas de la A a la Z.

```
<xs:element name="letra">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:pattern value="[A-Z]{3}"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

4. Define un elemento llamado "prodid" con una restricción. El único valor aceptable es de cinco dígitos en una secuencia, y cada dígito debe estar en un rango de 0 a 9.

```
<xs:element name="prodid">
  <xs:simpleType>
    <xs:restriction base="xs:integer">
      <xs:pattern value="[0-9]{5}"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

5. Define un elemento llamado "letra" con una restricción. El único valor aceptable es cero o más ocurrencias de letras minúsculas de la a a la z.

```
<xs:element name="letra">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:pattern value="[a-z]*"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

6. Define un elemento llamado "contraseña" con una restricción. Debe haber exactamente ocho caracteres en una fila y los caracteres deben estar en minúsculas o en mayúsculas (desde la a/A a la z/Z) o ser números del 0 al 9.

```
<xs:element name="contraseña">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:pattern value="[a-zA-Z0-9]{8}"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

7. Define un elemento llamado "dirección" con una restricción. La restricción de espacio en blanco en ajustado a "preservar". No eliminará los caracteres de espacio en blanco.

```
<xs:element name="dirección">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:whiteSpace value="preserve"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

8. Define otro elemento llamado "contraseña" con una restricción. El valor debe ser como mínimo de cinco caracteres y un máximo de ocho caracteres.

```
<xs:element name="contraseña">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:minLength value="5"/>

      <xs:maxLength value="8"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

9. Rellena los huecos que faltan en la siguiente parte del esquema:

```
<xs:element name="bola">
  <xs:complexType>
    <xs:attribute name="numero" type="_____"/>
  </xs:complexType>
</xs:element>

<xs:simpleType name="numeroDeBola">
  <xs:_____ base="xs:positiveInteger">
    <xs:_____ value="1"/>

    <xs:_____ value="90"/>
  </xs:restriction>
</xs:_____>
```

Respuesta:

```
<xs:element name="bola">
  <xs:complexType>
    <xs:attribute name="numero" type="numeroDeBola"/>
  </xs:complexType>
</xs:element>

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

    <xs:maxInclusive value="90"/>
  </xs:restriction>
</xs:simpleType>
```

10. Dado el siguiente esquema, responde a las siguientes preguntas:

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

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="agenda">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="contacto" maxoccurs="unbounded">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="nombre" type="xs:string"/>

              <xs:element name="telefono" type="xs:string" minoccurs="0"
maxoccurs="2"/>

              <xs:element name="telefono" type="unsignedInt"/>

              <xs:element name="edad" type="xs:unsignedByte"/>

              <xs:element name="email" type="xs:string" minoccurs="1"
maxoccurs="3"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

a. ¿Podrías añadir 7 contactos más a la agenda?

Sí, se pueden añadir 7 contactos a la agenda. Por ejemplo:

```
<agenda>
  <contacto>
    <nombre>Carla López</nombre>

    <telefono>911-11-22-33</telefono>

    <telefono>912-22-33-44</telefono>

    <edad>25</edad>

    <email>carla.lopez@example.com</email>
  </contacto>

  <contacto>
    <nombre>Pablo Sánchez</nombre>

    <telefono>913-33-44-55</telefono>

    <edad>35</edad>

    <email>pablo.sanchez@example.com</email>
  </contacto>
```

```
<contacto>
  <nombre>Sofia Rodríguez</nombre>

  <telefono>914-44-55-66</telefono>

  <telefono>915-55-66-77</telefono>

  <edad>30</edad>

  <email>sofia.rodriguez@example.com</email>
</contacto>

<contacto>
  <nombre>Miguel Álvarez</nombre>

  <telefono>916-66-77-88</telefono>

  <edad>45</edad>

  <email>miguel.alvarez@example.com</email>
</contacto>

<contacto>
  <nombre>Paula Gómez</nombre>

  <telefono>917-77-88-99</telefono>

  <edad>28</edad>

  <email>paula.gomez@example.com</email>
</contacto>

<contacto>
  <nombre>Adrián Fernández</nombre>

  <telefono>918-88-99-00</telefono>

  <telefono>919-99-00-11</telefono>

  <edad>32</edad>

  <email>adrian.fernandez@example.com</email>
</contacto>

<contacto>
  <nombre>Julia García</nombre>

  <telefono>920-00-11-22</telefono>

  <edad>37</edad>

  <email>julia.garcia@example.com</email>
</contacto>
```

```
</agenda>
```

b. Si un usuario deja el campo de teléfono sin rellenar, ¿estaría bien?

En el esquema XML proporcionado, el campo "telefono" tiene un atributo "minoccurs" con valor "0", lo que indica que no es obligatorio que se proporcione un número de teléfono para cada contacto. Por lo tanto, si un usuario deja el campo "telefono" sin rellenar en un formulario o en una aplicación que utiliza este esquema, no habría ningún problema en términos de validación del esquema XML.

c. Si el usuario Andrés dispone de dos teléfonos móviles y uno fijo, ¿podrían añadirse todos en XML?

Sí, en el esquema XML proporcionado, el campo "telefono" tiene un atributo "maxoccurs" con un valor de "2", lo que significa que se pueden proporcionar hasta dos valores para el campo "telefono". Además, se definen dos elementos de teléfono distintos en el esquema XML: uno con el tipo de datos "xs:string" y otro con el tipo de datos "xs:unsignedInt". Por lo tanto, es posible proporcionar dos números de teléfono móvil y un número de teléfono fijo para el contacto Andrés.

Por ejemplo, una posible instancia XML que represente los datos de contacto de Andrés con dos números de teléfono móvil y uno fijo sería:

```
<agenda>
  <contacto>
    <nombre>Andrés</nombre>

    <telefono>910-123-456</telefono>

    <telefono>911-987-654</telefono>

    <telefono>912-123-456</telefono>

    <edad>30</edad>

    <email>andres@example.com</email>
  </contacto>
</agenda>
```

d. ¿Cuántos correos electrónicos, como máximo y mínimo, hay que introducir?

En el esquema XML proporcionado, el campo "email" tiene un atributo "minoccurs" con valor "1" y un atributo "maxoccurs" con valor "3". Esto significa que cada contacto debe tener al menos un correo electrónico ("minoccurs"="1") y puede tener hasta tres correos electrónicos ("maxoccurs"="3").

Por lo tanto, al crear una instancia XML que siga este esquema, debe proporcionarse al menos un correo electrónico para cada contacto, y se puede proporcionar hasta un máximo de tres correos electrónicos para cada contacto. Si se proporcionan más de tres correos electrónicos para un contacto en una instancia XML, la validación del esquema XML fallará.

e. ¿De qué tipo es el campo edad? ¿Qué significa? ¿Qué valores podrá tomar según el tipo?

El campo "edad" en el esquema XML proporcionado tiene el tipo de datos "xs:unsignedByte". Este es un tipo de datos numérico que puede almacenar valores enteros no negativos entre 0 y 255.

El tipo "xs:unsignedByte" es una subclase de "xs:unsignedShort", que a su vez es una subclase de "xs:unsignedInt", que es una subclase de "xs:decimal". Esto significa que el tipo "xs:unsignedByte" hereda las propiedades y restricciones de sus clases padre, como la capacidad de realizar operaciones matemáticas y de comparación.

En el esquema XML proporcionado, el campo "edad" representa la edad del contacto en años. Al utilizar el tipo de datos "xs:unsignedByte", se garantiza que el valor de edad no puede ser negativo y no puede ser mayor de 255.

Por lo tanto, los valores que puede tomar el campo "edad" en una instancia XML que siga este esquema deben ser números enteros no negativos entre 0 y 255.