

XML SCHEMA (XSD)

Introducción a XML

De DTD a XML Schema

```
<?xml version="1.0"?>
<note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>To write Tove!</body>
</note>
```

```
<!ELEMENT note (to, from,
heading, body)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT heading (#PCDATA)>
<!ELEMENT body (#PCDATA)>
```

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

  <xs:element name="note">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="to" type="xs:string"/>
        <xs:element name="from" type="xs:string"/>
        <xs:element name="heading" type="xs:string"/>
        <xs:element name="body" type="xs:string"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

</xs:schema>
```

Referencia al archivo de definición

```
<?xml version="1.0"?>

<!DOCTYPE note SYSTEM
"http://www.mysite.com/dtd/note.dtd">

<note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
```

```
<?xml version="1.0"?>
<note
xmlns="http://www.mysite.com"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.mysite.com note.xsd">
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
```

Elementos simples

- No puede contener otros elementos o atributos
- Puede contener únicamente “texto”
 - Tipos incluidos en la definición XML Schema (boolean, string, date, etc.), o
 - Un tipo personalizado que el usuario puede definir

Sintaxis de los elementos simples

```
<xs:element name="xxx" type="yyy"/>
```

- Tipos más comunes:

- xs:string
- xs:decimal
- xs:integer
- xs:boolean
- xs:date
- xs:time

Elementos simples: Ejemplo

```
<lastname>Aguilar</lastname>  
<age>36</age>  
<dateborn>1970-03-27</dateborn>
```

```
<xs:element name="lastname" type="xs:string"/>  
<xs:element name="age" type="xs:integer"/>  
<xs:element name="dateborn" type="xs:date"/>
```

Definición

```
<?xml version="1.0" encoding="UTF-8"?>  
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"  
  elementFormDefault="qualified">  
  
  <xs:element name="lastname" type="xs:string"/>  
  <xs:element name="age" type="xs:integer"/><xs:element  
    name="dateborn" type="xs:date"/>  
  
</xs:schema>
```

Valores default y fijo

- ❖ Elementos simples pueden tener un valor por defecto o un valor fijo especificado.

- ❖ Un valor por defecto se asigna automáticamente al elemento cuando no se especifica ningún otro valor.

- ❖ En el siguiente ejemplo el valor por defecto es "rojo":

```
<xs:element name="color" type="xs:string" default="red"/>
```

- ❖ Un valor fijo también se asigna automáticamente al elemento, y no se puede indicar otro valor.

- ❖ En el siguiente ejemplo el valor fijo es "rojo":

```
<xs:element name="color" type="xs:string" fixed="red"/>
```

Sintaxis de los Atributos

- ❖ Todos los atributos se declaran como tipos simples.
- ❖ Los elementos simples no pueden tener atributos. Si un elemento tiene atributos, se considera de tipo complejo.
- ❖ Pero el atributo siempre es declarado como un tipo simple.
- ❖ Sintaxis:

`<xs:attribute name="xxx" type="yyy"/>`

- Tipos más comunes:
 - xs:string
 - xs:decimal
 - xs:integer
 - xs:boolean
 - xs:date
 - xs:time

Atributos: Ejemplo

Uso:

```
<lastname lang="EN">Smith</lastname>
```

Definición:

```
<xs:attribute name="lang" type="xs:string"/>
```

Los atributos son opcionales por defecto. Para especificar que se requiere el atributo, utilice "use":

```
<xs:attribute name="lang" type="xs:string" default="EN"/>
```

```
<xs:attribute name="lang" type="xs:string" fixed="EN"/>
```

```
<xs:attribute name="lang" type="xs:string" use="required"/>
```

Default, fixed, optional/required

XSD: Restricciones o facetas

Las restricciones son usadas para definir valores aceptables para los elementos o atributos XML. Estas restricciones se llaman **facetas**.

- Restringiendo valores:

```
<xs:element name="age">
  <xs:simpleType>
    <xs:restriction base="xs:integer">
      <xs:minInclusive value="0"/>
      <xs:maxInclusive value="120"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

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

XSD: Restricciones en los tipos de datos

Constraint	Description
enumeration	Defines a list of acceptable values
fractionDigits	Specifies the maximum number of decimal places allowed. Must be equal to or greater than zero
length	Specifies the exact number of characters or list items allowed. Must be equal to or greater than zero
maxExclusive	Specifies the upper bounds for numeric values (the value must be less than this value)
maxInclusive	Specifies the upper bounds for numeric values (the value must be less than or equal to this value)
maxLength	Specifies the maximum number of characters or list items allowed. Must be equal to or greater than zero
minExclusive	Specifies the lower bounds for numeric values (the value must be greater than this value)
minInclusive	Specifies the lower bounds for numeric values (the value must be greater than or equal to this value)
minLength	Specifies the minimum number of characters or list items allowed. Must be equal to or greater than zero
pattern	Defines the exact sequence of characters that are acceptable
totalDigits	Specifies the exact number of digits allowed. Must be greater than zero
whiteSpace	Specifies how white space (line feeds, tabs, spaces, and carriage returns) is handled

XSD: Restricciones en un conjunto de valores

Para limitar el contenido de un elemento XML a un conjunto de valores aceptables, usamos la restricción enumeración. El siguiente ejemplo define un elemento llamado "car" con una restricción. Los únicos valores aceptables son: Audi, Golf, BMW:

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

El tipo carType puede ser utilizado por otros elementos, ya que no es parte del elemento de "coche".

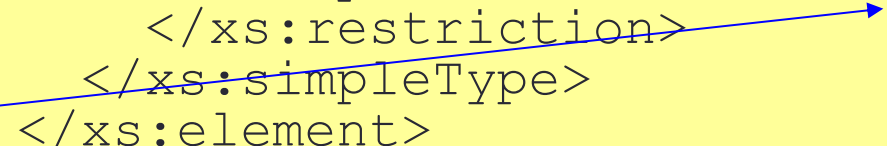
```
<xs:element name="car" type="carType"/>

<xs:simpleType name="carType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Audi"/>
    <xs:enumeration value="Golf"/>
    <xs:enumeration value="BMW"/>
  </xs:restriction>
</xs:simpleType>
```

XSD: Restricciones en una serie de valores

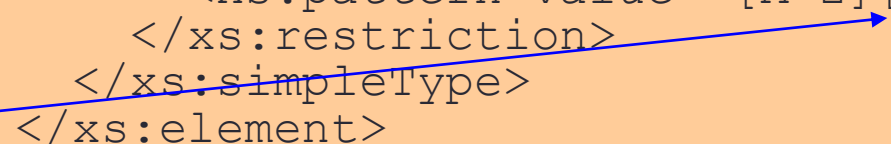
Uso de
patrones
restricción,
solo una letra
minúscula

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



Uso de
patrones
restricción,
tres letras
mayúsculas

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

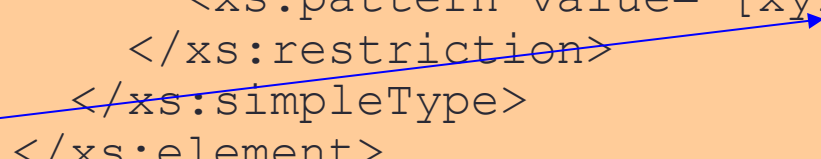


XSD: Restricciones en una serie de valores

```
<xs:element name="initials">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:pattern value="[a-zA-Z][a-zA-Z][a-zA-Z]"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

Uso de
patrones
restricción,
solo una letra
x, y, z

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



XSD: Restricciones en una serie de valores

Cinco dígitos en
secuencia

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

Cero o más
letras en
secuencia

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

XSD: Restricciones en una serie de valores

Conjunto de letras (uno o más), iniciando con una minúscula y una mayúscula

```
<xs:element name="letter">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:pattern value="([a-z][A-Z])+"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

Sólo uno male o female

```
<xs:element name="gender">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:pattern value="male|female"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```


XSD: Restricciones en una serie de valores

Exactamente 8
letras
minúsculas o
mayúsculas

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

Exactamente 8
letras
minúsculas,
mayúsculas o
números

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

XSD: Restricciones sobre los caracteres de espacio

Para especificar cómo los caracteres de espacio en blanco se deben manipular, usamos la restricción de espacio en blanco.

La restricción de espacio en blanco es ajustada a "preserve", lo que significa que el procesador XML no eliminará cualquier carácter de espacio en blanco

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

El procesador XML reemplazará todos los caracteres de espacios en blanco (saltos de línea, tabulaciones, espacios y retornos de carro) con espacios

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

XSD: Restricciones sobre los caracteres de espacio

El procesador XML removerá los espacios en blanco redundantes (saltos de línea, tabuladores, espacios son reemplazados con espacios; espacios en blanco antes y después de cada línea son removidos; múltiples espacios en blanco son reducidos a un solo espacio en blanco)

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

XSD: Restricciones sobre la longitud

Para limitar la longitud de un valor en un elemento, usamos las restricciones de length, maxLength y minLength.

```
<xs:element name="password">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:length value="8"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

El valor debe ser exactamente ocho caracteres:

```
<xs:element name="password">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:minLength value="5"/>
      <xs:maxLength value="8"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

el valor debe ser mínimo cinco caracteres y máximo ocho caracteres:

XSD: Elementos complejos

- Un elemento complejo es un elemento XML que contiene otros elementos y/o atributos.
- Existen cuatro tipos de elementos complejos:
 - Elementos vacíos
 - Elementos que contienen solamente otros elementos
 - Elementos que contienen solamente texto
 - Elementos que contienen tanto otros elementos como texto

XSD: Tipos de elementos complejos

Elementos vacíos

```
<product pid="1345"/>
```

Elementos que
contienen solamente
otros elementos

```
<employee>  
  <firstname>John</firstname>  
  <lastname>Smith</lastname>  
</employee>
```

Elementos que
contienen solamente
texto

```
<food type="dessert">Ice cream</food>
```

Elementos que
contienen tanto otros
elementos como
texto

```
<description>  
It happened on <date  
lang="norwegian">03.03.99</date> ....  
</description>
```

XSD: Definición de un elemento complejo

Employee contiene otros elementos complejos

```
<employee>
  <firstname>John</firstname>
  <lastname>Smith</lastname>
</employee>
```

Declarado directamente cada elemento

Los
elementos
hijos deben
aparecer en
el mismo
orden en que
se declaran.

```
<xs:element name="employee">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="firstname" type="xs:string"/>
      <xs:element name="lastname" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Definición

XSD: Definición de un elemento complejo

```
<employee>
  <firstname>John</firstname>
  <lastname>Smith</lastname>
</employee>
```

El elemento "employee" puede tener un atributo de tipo que hace referencia al nombre del tipo complejo a usar:

```
<xs:element name="employee" type="personinfo"/>

<xs:complexType name="personinfo">
  <xs:sequence>
    <xs:element name="firstname" type="xs:string"/>
    <xs:element name="lastname" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
```

Definición

XSD: Definición de un elemento complejo

Varios elementos pueden referir al mismo tipo complejo

```
<xs:element name="employee" type="fullpersoninfo"/>

<xs:complexType name="personinfo">
  <xs:sequence>
    <xs:element name="firstname" type="xs:string"/>
    <xs:element name="lastname" type="xs:string"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="fullpersoninfo">
  <xs:complexContent>
    <xs:extension base="personinfo">
      <xs:sequence>
        <xs:element name="address" type="xs:string"/>
        <xs:element name="city" type="xs:string"/>
        <xs:element name="country" type="xs:string"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Definición

XSD: Definición de un elemento complejo

También puede ser un elemento base complejo en un elemento complejo existente y añadir algunos elementos, como este:

```
<xs:element name="employee" type="fullpersoninfo"/>

<xs:complexType name="personinfo">
  <xs:sequence>
    <xs:element name="firstname" type="xs:string"/>
    <xs:element name="lastname" type="xs:string"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="fullpersoninfo">
  <xs:complexContent>
    <xs:extension base="personinfo">
      <xs:sequence>
        <xs:element name="address" type="xs:string"/>
        <xs:element name="city" type="xs:string"/>
        <xs:element name="country" type="xs:string"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

XSD: Elementos complejos vacíos

```
<product prodid="1345" />
```

```
<xs:element name="product">  
  <xs:complexType>  
    <xs:complexContent>  
      <xs:restriction base="xs:integer">  
        <xs:attribute name="prodid" type="xs:positiveInteger"/>  
      </xs:restriction>  
    </xs:complexContent>  
  </xs:complexType>  
</xs:element>
```

Definición

```
<xs:element name="product">  
  <xs:complexType>  
    <xs:attribute name="prodid" type="xs:positiveInteger"/>  
  </xs:complexType>  
</xs:element>
```

```
<xs:element name="product" type="prodtype"/>  
  
<xs:complexType name="prodtype">  
  <xs:attribute name="prodid" type="xs:positiveInteger"/>  
</xs:complexType>
```

XSD: Elementos complejos que contienen solamente elementos

```
<person>
  <firstname>John</firstname>
  <lastname>Smith</lastname>
</person>
```

```
<xs:element name="person">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="firstname" type="xs:string"/>
      <xs:element name="lastname" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Definición

```
<xs:element name="person" type="persontype"/>

<xs:complexType name="persontype">
  <xs:sequence>
    <xs:element name="firstname" type="xs:string"/>
    <xs:element name="lastname" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
```

XSD: Elementos complejos que contienen solamente texto

Este tipo sólo contiene contenido simple (texto y atributos), por lo tanto, añadimos un elemento `simpleContent` a todo el contenido. Al utilizar contenido simple, debe definir una extensión o una restricción en el elemento `simpleContent`.

```
<xs:element name="somename">
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base="basetype">
        ....
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>
```

Expandir o limitar el tipo base simple

O

```
<xs:element name="somename">
  <xs:complexType>
    <xs:simpleContent>
      <xs:restriction base="basetype">
        ....
      </xs:restriction>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>
```

XSD: Elementos complejos que contienen solamente texto

```
<shoesize country="france">35</shoesize>
```

```
<xs:element name="shoesize">  
  <xs:complexType>  
    <xs:simpleContent>  
      <xs:extension base="xs:integer">  
        <xs:attribute name="country" type="xs:string" />  
      </xs:extension>  
    </xs:simpleContent>  
  </xs:complexType>  
</xs:element>
```

Definición

```
<xs:element name="shoesize" type="shoetype"/>  
  
<xs:complexType name="shoetype">  
  <xs:simpleContent>  
    <xs:extension base="xs:integer">  
      <xs:attribute name="country" type="xs:string" />  
    </xs:extension>  
  </xs:simpleContent>  
</xs:complexType>
```

XSD: Elementos complejos mixtos

```
<letter>
  Dear Mr.<name>John Smith</name>.
  Your order <orderid>1032</orderid>
  will be shipped on <shipdate>2001-07-13</shipdate>.
</letter>
```

Hace posible que los
datos de tipo caracter
aparezcan entre los
elementos hijo de
"letter"

```
<xs:element name="letter">
  <xs:complexType mixed="true">
    <xs:sequence>
      <xs:element name="name" type="xs:string"/>
      <xs:element name="orderid" type="xs:positiveInteger"/>
      <xs:element name="shipdate" type="xs:date"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Definición

XSD: Elementos complejos mixtos

```
<letter>
  Dear Mr.<name>John Smith</name>.
  Your order <orderid>1032</orderid>
  will be shipped on <shipdate>2001-07-13</shipdate>.
</letter>
```

```
<xs:element name="letter" type="lettertype"/>

<xs:complexType name="lettertype" mixed="true">
  <xs:sequence>
    <xs:element name="name" type="xs:string"/>
    <xs:element name="orderid" type="xs:positiveInteger"/>
    <xs:element name="shipdate" type="xs:date"/>
  </xs:sequence>
</xs:complexType>
```

Definición

XSD: Indicadores

Existen siete tipos de indicadores:

- Indicadores de orden:
 - All
 - Choice
 - Sequence
- Indicadores de ocurrencia:
 - maxOccurs
 - minOccurs
- Indicadores de grupo:
 - Group name
 - attributeGroup name

Indicadores de orden: ALL

Especifica que los elementos hijo pueden aparecer en cualquier orden, y que cada elemento hijo puede ocurrir solamente una vez.

```
<xs:element name="person">
  <xs:complexType>
    <xs:all>
      <xs:element name="firstname" type="xs:string"/>
      <xs:element name="lastname" type="xs:string"/>
    </xs:all>
  </xs:complexType>
</xs:element>
```

Indicadores de orden: CHOICE

Especifica que los elementos hijo puede aparecer (uno o el otro).

```
<xs:element name="person">
  <xs:complexType>
    <xs:choice>
      <xs:element name="employee" type="employee"/>
      <xs:element name="member" type="member"/>
    </xs:choice>
  </xs:complexType>
</xs:element>
```

Indicadores de orden: SEQUENCE

Especifica que los elementos hijo deben aparecer en estricta secuencia, tal y como se han definido.

```
<xs:element name="person">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="firstname" type="xs:string"/>
      <xs:element name="lastname" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Indicadores de ocurrencia

Indicadores maxOccurs y minOccurs (número de veces que un elemento hijo puede ocurrir)

```
<xs:element name="person">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="full_name" type="xs:string"/>
      <xs:element name="child_name" type="xs:string"
        maxOccurs="10"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

```
<xs:element name="person">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="full_name" type="xs:string"/>
      <xs:element name="child_name" type="xs:string"
        maxOccurs="10" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

maxOccurs="unbounded"

XSD: Ejercicio

(Escribir archivo XSD para Familia.xml)

```
<?xml version="1.0" encoding="ISO-8859-1"?>
```

```
<persons xmlns:xsi="http://www.w3.org/2001/XMLSchema-  
instance"
```

```
xsi:noNamespaceSchemaLocation="familia.xsd">
```

```
<person>
```

```
  <full_name>Hege Refsnes</full_name>
```

```
  <child_name>Cecilie</child_name>
```

```
</person>
```

```
<person>
```

```
  <full_name>Tove Refsnes</full_name>
```

```
  <child_name>Hege</child_name>
```

```
  <child_name>Stale</child_name>
```

```
  <child_name>Jim</child_name>
```

```
  <child_name>Borge</child_name>
```

```
</person>
```

```
<person>
```

```
  <full_name>Stale Refsnes</full_name>
```

```
</person>
```

```
</persons>
```

XSD: Respuesta a Práctica 8

(Archivo Familia.xsd para Familia.xml)

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">

  <xs:element name="persons">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="person" maxOccurs="unbounded">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="full_name" type="xs:string"/>
              <xs:element name="child_name" type="xs:string"
minOccurs="0" maxOccurs="unbounded"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

</xs:schema>
```

Ejercicio

- Obtener el correspondiente archivo XSD

```
<?xml version="1.0"?>
<class>
  <student rollno="393">
    <firstname>Dinkar</firstname>
    <lastname>Kad</lastname>
    <nickname>Dinkar</nickname>
    <marks>85</marks>
  </student>
  <student rollno="493">
    <firstname>Vaneet</firstname>
    <lastname>Gupta</lastname>
    <nickname>Vinni</nickname>
    <marks>95</marks>
  </student>
  <student rollno="593">
    <firstname>Jasvir</firstname>
    <lastname>Singh</lastname>
    <nickname>Jazz</nickname>
    <marks>90</marks>
  </student>
</class>
```


Solución

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

  <xs:element name='class'>
    <xs:complexType>
      <xs:sequence>
        <xs:element name='student' type='StudentType'
          minOccurs='0' maxOccurs='unbounded' />
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:complexType name="StudentType">
    <xs:sequence>
      <xs:element name="firstname" type="xs:string"/>
      <xs:element name="lastname" type="xs:string"/>
      <xs:element name="nickname" type="xs:string"/>
      <xs:element name="marks" type="xs:positiveInteger"/>
    </xs:sequence>
    <xs:attribute name='rollno' type='xs:positiveInteger'/>
  </xs:complexType>
</xs:schema>
```