

**TALLER No. 1****Fecha:** 09 de febrero de 2018

Sistemas de recuperación de información

**Profesor:** Jaime Alberto Guzmán Luna**Monitor:** Alejandro Villa Velez

---

Aprender y dominar los conceptos básicos sobre los documentos XML, como crear y comprobar la correcta formación de los archivos XML, además de aprender acerca de los Esquemas XML (XMLS) y la validación de un documento XML de acuerdo a su esquema

---

## 1. Introducción a los documentos XML

XML es un lenguaje de marcado o etiquetas como los es HTML, es recomendado por la W3C, obtiene sus siglas de eXtensible Markup Language. Está diseñado para transportar o almacenar datos más no para mostrar datos.

HTML y XML, a pesar de parecer tener un origen común, tienen objetivos totalmente distintos. Por un lado HTML fue creado para mostrar información de acuerdo a unas etiquetas pre-definidas que son interpretadas por un navegador, mientras que por otro lado XML fue creado para almacenar y transportar información y no para mostrarla (las etiquetas en XML son totalmente transparentes para el navegador).

## 2. Documentos XML usando Notepad++

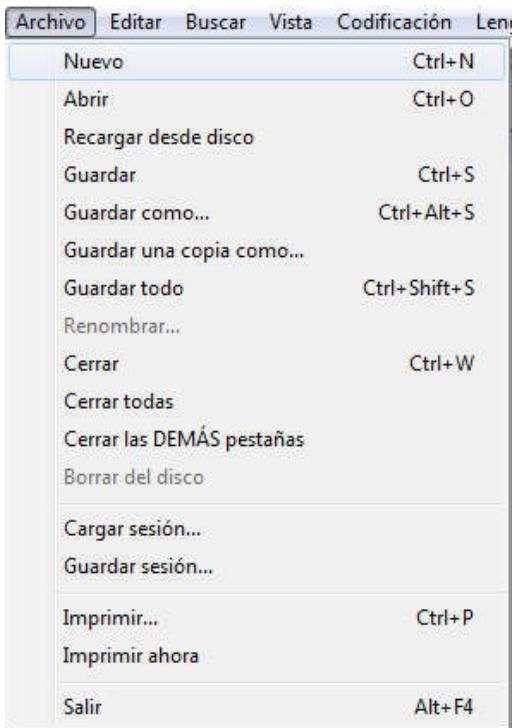
Notepad++ es una herramienta bastante versátil y potente no solo para crear o editar archivos XML sino para modificar casi tipo cualquier archivo escrito en texto plano.

Notepad++ se puede descargar desde: <http://notepad-plus-plus.org/>

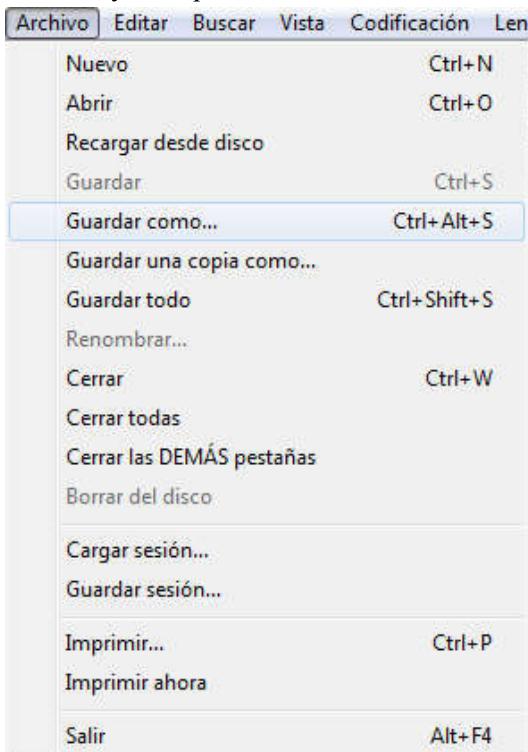
Para poder verificar si el documento está bien formado y es válido, se debe instalar un complemento para Notepad++ llamado XML Tools, disponible en <http://sourceforge.net/projects/npp-plugins/files/XML%20Tools/>.

Para crear un documento XML con Notepad++, bastará simplemente con guardar nuestro documento con el tipo correspondiente de la siguiente forma:

Creamos un nuevo documento en blanco



Una vez haya sido creado el nuevo documento en blanco podremos guardarlo asignándole un nombre y un tipo de documento haciendo clic en Archivo -> Guardar como...



Una vez hemos dado clic sobre la opción Guardar, se nos mostrará una ventana pidiéndonos el  
Universidad Nacional de Colombia –Sede Medellín-

nombre del archivo y el tipo del documento, allí le pondremos como nombre Hotel y como tipo de documento pondremos eXtensible Markup Language, dar clic en guardar y tendremos finalmente el documento XML listo para ser editado.

### Ejercicio 1

Transcribir el siguiente documento XML en el editor Notepad++, luego guardarlo

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!-- SISTEMAS DE RECUPERACION DE INFORMACION -->
<hoteles>
    <hotel>
        <nombre>La paz</nombre>
        <telefono>0180006789</telefono>
        <direccion calle="34" numero="29">avenida sur</direccion>
        <ciudad>medellin</ciudad>
        <estrellas>5</estrellas>
    </hotel>
    <hotel>
        <nombre>El calamar</nombre>
        <telefono>09787878787</telefono>
        <direccion calle="34" numero="29">Transversal 30</direccion>
        <ciudad>medellin</ciudad>
        <estrellas>2</estrellas>
    </hotel>
    <hotel>
        <nombre>Hotel Palmira</nombre>
        <telefono>89989897437</telefono>
        <direccion calle="34" numero="29">diagonal 34</direccion>
        <ciudad>medellin</ciudad>
        <estrellas>3</estrellas>
    </hotel>
    <hotel>
        <nombre>Oasis</nombre>
        <telefono>678907643433</telefono>
        <direccion calle="34" numero="29">avenida norte</direccion>
        <ciudad>medellin</ciudad>
        <estrellas>4</estrellas>
    </hotel>
</hoteles>
```

### 3. Componentes principales de un documento XML

Todo documento XML tiene 3 componentes principales que son: Prologo o encabezado, un nodo raíz y un contenido. Un archivo XML correcto debe tener por lo menos un encabezado y el nodo raíz, el contenido son los demás nodos que se desprenden del nodo raíz. En el siguiente archivo tenemos el código XML correspondiente a la descripción de una persona en XML.

- **Encabezado:** El encabezado o prologo, indican el tipo de documento al cual corresponde el

archivo, puede ser html, xml, doc, etc, en este caso es XML. Además contiene varios atributos; los más comunes son version y encoding que indican la versión de XML usada y la codificación del texto al interior del documento (respectivamente).

- **Nodo Raíz:** El nodo raíz es el elemento principal al interior del archivo XML (entiéndase por elemento todo aquel que empiece y termine por una misma etiqueta), este indica el comienzo y el final del documento, cualquier línea adicional después del cierre del nodo raíz es considerada un error. El nodo raíz, puede contener una indicación de la ubicación del esquema que rige la validez del documento, al igual que referencias a espacios de nombre (namespaces).
- **Contenido:** El contenido es el código que se encuentra principalmente al interior del nodo raíz, en caso de que el nodo raíz tenga referencia a un esquema XML, el contenido debe respetar y cumplir la estructura dictada por dicho esquema.

#### 4. Validación de sintaxis en un Documento XML

Para poder verificar que nuestro documento XML esté bien formado (que su sintaxis sea correcta), debemos activar en el Notepad++ algunas características del plugin instalado (XMLTools). Para esto, simplemente hacemos clic en Plugins --> XML Tools y se activan las opciones Enable XML Syntax auto check y opcionalmente Tag auto close.

Adicionalmente a la herramienta Notepad++, la W3C tiene en línea un validador de documentos en XML, que analiza la sintaxis e informa los errores que se produjeron en el análisis, el validador está disponible en la siguiente dirección: [http://validator.w3.org/#validate\\_by\\_upload](http://validator.w3.org/#validate_by_upload).

Validar el documento anterior y verificar que está bien formado.

#### Ejercicio 2

El siguiente documento XML está mal formado, verifíquelo tanto en el Notepad++ como en la página de la W3C: identifique las fallas mediante el uso del validador y luego corrija el documento, por último asegúrese de validarla de nuevo y cheque que es un documento bien formado.

```
<?xml version="\1.0">
<--Sistemas de Recuperacion-->
<mensaje>
<remite.>
<nombre id=8976>Alfredo Reino</nombre>
<email>alf@ibium.com</email>
</remite>
<destinatario>
<nommbre>
<nombre id=0976>Bill Clinton</nombre>
<email>president@whitehouse.gov
</destinatario>
<asunto">Hola Bill</asunto>
<texto.>
<<parrafo>¿Hola qué tal? Hace <enfasis->mucho</enfasis> que no escribes. A ver
si llamas y quedamos para tomar algo.</parrafo>
```

```
</texto>
</mensaje>
</mensaje>
```

## 5. Atributos y Elementos en XML

Como se mencionó anteriormente, un elemento en XML es todo aquello que tenga una etiqueta de inicio y una etiqueta de cierre.

Un atributo es un componente perteneciente a un elemento y proporciona información adicional acerca de ese elemento. Los atributos van siempre en la etiqueta de inicio de un elemento y su valor se debe escribir entre comillas simples o dobles.

No existen normas acerca de cuándo usar un elemento y cuando usar un atributo. Sin embargo de la práctica se ha determinado que es preferible evitar usar atributos en la medida de lo posible por lo siguiente:

- Los atributos no pueden tener múltiples valores mientras que los elementos si pueden
- Los atributos no pueden contener estructuras en árbol, mientras que los elementos si pueden

En general los atributos son difíciles de leer y de mantener. Es recomendable usar los elementos para datos y los atributos para dar información sin relación con los datos.

### Ejercicio 3

Retomar el documento XML de los hoteles y representar la calle y el número como elementos y el teléfono como atributo.

## 6. Espacios de Nombre (Namespace) en XML

Es muy posible que durante el desarrollo de un documento XML, nos encontremos frente a la dificultad de poder diferenciar dos elementos que se llamen igual pero que pertenecen a contextos totalmente diferentes, por ejemplo diferenciar entre el peso (propiedad de un cuerpo), el peso (moneda colombiana) y el peso (moneda argentina). Para solucionar este inconveniente tenemos a nuestra disposición los espacios de nombre. Los espacios de nombre pueden ser declarados como atributos del elemento raíz o como atributo del elemento que hará uso del mismo.

La sintaxis para declarar un namespace como atributo al interior de un elemento es la siguiente:  
xmlns:prefijo="URI"

De acuerdo a lo anterior, el problema propuesto podría solucionarse de la siguiente forma:

```

<?xml version="1.0" encoding="iso-8859-1" ?>
<Pesos xmlns:col="http://es.wikipedia.org/wiki/Peso_colombiano"
        xmlns:arg="http://es.wikipedia.org/wiki/Peso_de_Argentina">
    <col:Peso>
        <!-- Valor en dólares -->
        <col:Valor>0.000564</col:Valor>
    </col:Peso>
    <Peso>Propiedad Física de los cuerpos.</Peso>
    <arg:Peso>
        <arg:Valor>0.230840</arg:Valor>
    </arg:Peso>
</Pesos>

```

**Nota:** El URI del namespace no es utilizado por el analizador para buscar información, es usado únicamente para darle unicidad al namespace. Sin embargo, generalmente se utiliza como URI una dirección que contenga información referente al namespace

#### Ejercicio 4

Responda las siguientes preguntas:

A. ¿Cuál es la sintaxis correcta de la declaración que define la versión de xml?

- <xml version="1.0" />
- <?xml version="1.0" />
- <?xml version="1.0"?>

B. ¿Es el siguiente un docuemnto xlm bien formado?

```

<?xml version="1.0"?>
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>

```

- Si
- No

C. ¿Es el siguiente un docuemnto xlm bien formado?

```

<?xml version="1.0"?>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>

```

- Si
- No

D. ¿Cuál afirmación es falsa?

- Todos los elementos XML deben estar correctamente cerrados
- Todos los elementos XML deben estar en letras en minúscula
- Todos los documentos XML deben tener una etiqueta raíz
- Todos los elementos XML deben estar correctamente anidados
- Las etiquetas XML distinguen entre mayúsculas y minúsculas

E. XML conserva espacios en blanco

- Si
- No

F. ¿Es el siguiente un documento xml bien formado?

```
<?xml version="1.0"?>
<note>
<to age="29">Tove</to>
<from>Jani</from>
</note>
```

- Si
- No

G. ¿Es el siguiente un documento xml bien formado?

```
<?xml version="1.0"?>
<note>
<to age=29>Tove</to>
<from>Jani</from>
</note>
```

- Si
- No

H. Los elementos XML no pueden ser vacíos

- Falso
- Verdadero

I. ¿Cuál no es un nombre correcto para un elemento xml?

- <1dollar>
- <h1>
- <Note>
- All 3 names are incorrect

J. ¿Cuál no es un nombre correcto para un elemento xml?

- All 3 names are incorrect
- <NAME>
- <first name>

- <age>
- K. Los atributos XML deben estar siempre encerrados entre comillas
  - Verdadero
  - Falso
- L. ¿Para qué cierta sección del documento XML sea ignorada, cual es la sintaxis correcta?
  - <CDATA> Text to be ignored </CDATA>
  - <xml:CDATA[ Text to be ignored ]>
  - <![CDATA[ Text to be ignored ]]>
  - <PCDATA> Text to be ignored </PCDATA>
- M. ¿Cuál afirmación es verdadera?
  - Los atributos siempre deben estar presentes
  - Los atributos siempre deben aparecer en un orden definido
  - Ninguna de las dos es correcta
  - Las dos son correctas
- N. ¿Cuál de los siguientes fragmentos XML está bien formado?
  - <customer id=3456><name>John Smith</name></customer>
  - <customer id="3456"><address/><zip code="3456"/></customer>
- O. ¿Qué es una instancia XML?
  - Un documento XML
  - Un elemento XML
  - Un atributo XML

## 7. Esquemas XML (XMLS)

Existen herramientas para determinar si un documento XML está bien formado, es decir que su sintaxis sea correcta, sin embargo no tenemos aún una manera de comprobar si un archivo XML, cumple con nuestras necesidades. Supongamos por ejemplo que nosotros no somos los únicos encargados de desarrollar documentos XML al interior de nuestro grupo, claramente cada integrante del grupo puede hacer su documento XML de la manera que mejor le parezca y tendrían como resultado muchos documentos descriptivos acerca de lo mismo pero de maneras probablemente muy distintas. Para solucionar este tipo de problemas, surge XML Schema, que nos permite establecer normas sobre un documento XML que lo referencia. Un Esquema XML, entre muchas cosas, nos permite determinar por ejemplo, los elementos y atributos que debe contener el documento, el orden en que estos deben aparecer, determinar si un elemento es único, entre muchas cosas más. Veamos entonces algunos de los principales conceptos de un XML Schema y como referenciarlos desde un documento XML.

## 8. Creación de un Esquema XML

Al igual que un documento XML cualquiera, los esquemas deben tener un encabezado, un elemento raíz y un contenido, sin embargo, dado que están basados en un estándar de la W3C para esquemas, sus elementos son un poco más restringidos.

- **El encabezado:** El encabezado es similar al de cualquier elemento XML, de manera que no contendrá elementos ni declaraciones adicionales. Dado que un esquema XML no es una instancia, no es necesario declarar el encoding.
- **El nodo raíz:** Este componente tiene una importancia vital dentro del esquema XML. El nombre de este elemento debe ser schema y es necesario entre sus atributos crear las referencias a los elementos de un XMLSchema, declarar el targetNamespace, el namespace por defecto y elementFormDefault.
  - o **Referencia a los elementos:** Para hacer referencia a los componentes o elementos estándar de un XML Schema, es necesario declarar un namespace llamado xsd que apunte a la URI “<http://www.w3.org/2001/XMLSchema>”.
  - o **El targetNamespace:** El espacio de nombre objetivo, indica que los elementos definidos a lo largo del esquema, estarán en el namespace con la URI indicada, en efecto este URI puede ser la ubicación de un archivo o un elemento en la web.
  - o **El namespace por defecto:** Este componente es un atributo llamado xmlns cuyo valor indica el espacio de nombre usado por defecto, generalmente el valor de este atributo es igual al del targetNamespace.
  - o **El elementFormDefault:** Este atributo tiene como valor “qualified”, este valor indica que los nombres de los elementos de la instancia del esquema deben obligatoriamente usar el espacio de nombre.

De este modo, el elemento raíz de un XML Schema, podría quedar de la siguiente forma:

```
<xs:schema elementFormDefault="qualified"
            targetNamespace="http://www.onsheld.co.uk/ns1"
            xmlns:xs="http://www.w3.org/2001/XMLSchema">
```

- **El contenido:** El contenido de un Schema XML, está dado por los elementos estándar de un esquema XML estipulados por la W3C, y por eso cada elemento debe ser creado usando el namespace que hace referencia a los elementos de un Schema XML.

## 9. Tipos de datos en XML Schema:

En los esquemas XML, es posible determinar el tipo de dato de un elemento cualquiera, existen tipos simples y tipos complejos.

**Tipos Simples:** Son los tipos de datos básicos disponibles para un esquema, entre estos están: string, boolean, integer, decimal, time, date, anyURI, etc.

La declaración de un tipo de elemento simple sería así:

- <xsd:element name="Cadena" type="xs:string"/>
- <xsd:element name="Numero" type="xs:integer"/>
- <xsd:element name="direccion\_web" type="xs:anyURI"/>

También podemos crear nuestros propios tipos de datos simples:

```
<xsd:simpleType name="meses">
    <xsd:restriction base="xsd:string">
        <xsd:enumeration value="Jan"/>
        <xsd:enumeration value="Feb"/>
        <xsd:enumeration value="Mar"/>
        <xsd:enumeration value="Apr"/>
        <xsd:enumeration value="May"/>
        <xsd:enumeration value="Jun"/>
        <xsd:enumeration value="Jul"/>
        <xsd:enumeration value="Aug"/>
        <xsd:enumeration value="Sep"/>
        <xsd:enumeration value="Oct"/>
        <xsd:enumeration value="Nov"/>
        <xsd:enumeration value="Dec"/>
    </xsd:restriction>
</xsd:simpleType>
```

**Tipos Complejos:** Los tipos de dato complejos, son elementos que pueden tener atributos de cualquier tipo, sub-elementos que a su vez pueden ser simples o también compuestos

La estructura de un elemento complejo podría ser así:

```
<xs:element name="userdetails">
    <xs:complexType>
        <xs:sequence>
            <xs:element type="xs:string" name="username"/>
            <xs:element type="xs:string" name="password"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>
```

## 10. Indicadores de ocurrencia para XML Schema:

Los esquemas XML, nos permiten determinar la cantidad mínima y/o máxima de veces que un elemento cualquiera puede aparecer en un documento XML. Existen dos indicadores de ocurrencia que son, minOccurs y maxOccurs:

**Indicador de Ocurrencia minOccurs:** Como su nombre lo indica, nos permite determinar el

número mínimo de veces que un elemento puede aparecer en la estructura del documento.

**Indicador de Ocurrencia maxOccurs:** Como su nombre lo indica, nos permite determinar el número mínimo de veces que un elemento puede aparecer en la estructura del documento.

```
<xsd:element name="Memoria" type="RAM" minOccurs="1"/>
<xsd:element name="Procesador" type="CPU" maxOccurs="1"/>
```

## 11. Indicadores de orden para XML Schema:

Los esquemas XML, nos permiten determinar los elementos que pueden aparecer en un documento XML, podemos determinar el orden en que estos aparezcan o dar unos posibles elementos que puedan componer el documento. Existen tres indicadores de orden basicos que son, all, sequence y choice.

**Indicador de orden all:** Este indicador exige que los elementos que se declaren dentro de este deben aparecer en la instancia, sin importar el orden.

```
<xsd:element name="PC" type="Computador"/>
<xsd:complexType name="Computador">
    <xsd:all>
        <xsd:element name="Procesador" type="CPU" maxOccurs="1"/>
        <xsd:element name="Memoria" type="RAM" minOccurs="1"/>
        <xsd:element name="Disco" type="HDD" minOccurs="1"/>
        <xsd:element name="Board" type="Main_Board" maxOccurs="1"/>
    </xsd:all>
</xsd:complexType>
```

**Indicador de orden Sequence:** Este indicador exige que los elementos que se declaren dentro de este deben aparecer en el orden establecido

```
<xsd:element name="Aplicacion" type="Software"/>
<xsd:complexType name="Software">
    <xsd:sequence>
        <xsd:element name="Name" type="xs:string"/>
        <xsd:element name="Version" type="xs:integer"/>
        <xsd:element name="Date" type="xs:date"/>
    </xsd:sequence>
</xsd:complexType>
```

**Indicador de orden Choice:** Este indicador indica que, de los elementos que se declaren dentro de este deben solo uno puede ser utilizado.

```

<xsd:element name="tipo" type="tipo_dcto"/>
<xsd:complexType name="tipo_dcto">
    <xsd:choice>
        <xsd:element name="TI" type="Tarjeta_identidad"/>
        <xsd:element name="CC" type="Cedula"/>
    </xsd:choice>
</xsd:complexType>

```

### Ejercicio 5

Marcar un XML esquema que defina una familia, la familia está compuesta por un conjunto de exactamente de 4 personas, cada persona debe contener un nombre, un apellido y al menos 2 nombres de los hijos de esa persona.

Puede usar el siguiente validador, que además tiene varios ejemplos:

[http://www.utilities-online.info/xsdvalidation/#.U-AEI\\_mBN8E](http://www.utilities-online.info/xsdvalidation/#.U-AEI_mBN8E)

### 12. Referenciación de un esquema XML desde una instancia

Evidentemente, no tendría sentido desarrollar un esquema xml, si no existiera una forma automática de comprobar si una instancia cumple con los requisitos dados por el esquema, para esto, debemos indicar que nuestro documento XML es una instancia e indicar la ubicación del esquema XML. Esto se debe indicar en el nodo raíz del documento, podría ser como se muestra a continuación:

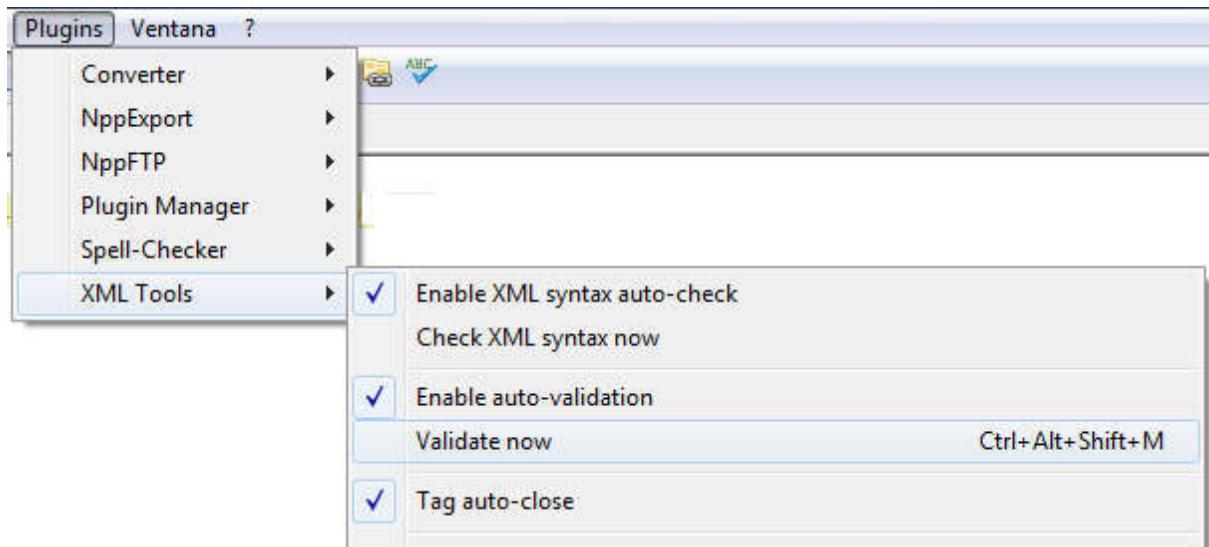
```

<?xml version="1.0"?>
<userdetails xmlns="http://www.onsheld.co.uk/ns1"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
              xsi:schemaLocation="file:///D:/Dropbox/Unal/Monitoria/SRIW/2015/Taller1/userdetails2.xsd">
    <username>Fred</username>
    <password>pass123</password>
</userdetails>

```

### 13. Validación de un documento XML

Para validar un documento XML, correctamente formado usando Notepad++, debemos habilitar algunas características del plugin. Para esto vamos a plugins -> XML Tools y damos clic sobre validate now. Podemos también activar la auto validación para que lo haga cada vez que se guarden los cambios.



**Ejercicio 6:** Realizar un documento XML que tenga como modelo de datos el Esquema XML realizado en el ejercicio 5 y valídelo.

## 14. JSON

JSON (JavaScript Object Notation) es un formato para el intercambio de datos, básicamente JSON describe los datos con una sintaxis dedicada que se usa para identificar y gestionar los datos. JSON nació como una alternativa a XML, el fácil uso en javascript ha generado un gran número de seguidores de esta alternativa. Una de las mayores ventajas que tiene el uso de JSON es que puede ser leído por cualquier lenguaje de programación. Por lo tanto, puede ser usado para el intercambio de información entre distintas tecnologías.

## Ejercicio 7

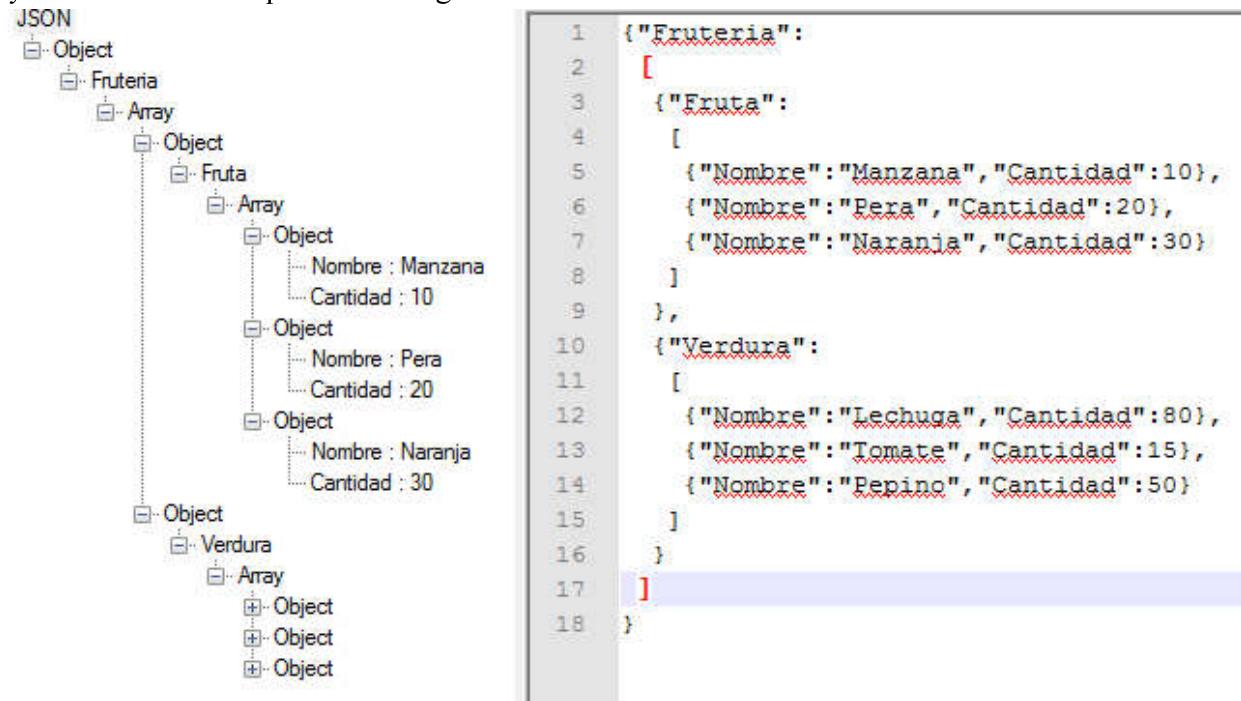
Copiar el siguiente ejemplo en JSON de una frutería.

```
{"Fruteria": [
  [
    {"Fruta": [
      [
        {"Nombre": "Manzana", "Cantidad": 10},
        {"Nombre": "Pera", "Cantidad": 20},
        {"Nombre": "Naranja", "Cantidad": 30}
      ]
    ],
    {"Verdura": [
      [
        {"Nombre": "Lechuga", "Cantidad": 80},
        {"Nombre": "Tomate", "Cantidad": 15},
        {"Nombre": "Pepino", "Cantidad": 50}
      ]
    ]
  ]
]
```

## 15. Visualización de una cadena JSON

Instalar el complemento JSON Viewer para notepad++ de la siguiente dirección:  
<http://sourceforge.net/projects/nppjsonviewer/>

Seleccionar el string en notepad++ y dar click en Plugins -> JSON Viewer -> Show JSON Viewer y se mostrará una representación gráfica de la cadena como se muestra a continuación:



The screenshot shows the Notepad++ interface with a JSON file open. On the left, there is a tree view showing the hierarchical structure of the JSON object. On the right, the actual JSON code is displayed with line numbers. The JSON structure represents a fruit and vegetable store inventory.

```
1  {"Fruteria":  
2  [  
3      {"Fruta":  
4          [  
5              {"Nombre": "Manzana", "Cantidad": 10},  
6              {"Nombre": "Pera", "Cantidad": 20},  
7              {"Nombre": "Naranja", "Cantidad": 30}  
8          ]  
9      },  
10     {"Verdura":  
11         [  
12             {"Nombre": "Lechuga", "Cantidad": 80},  
13             {"Nombre": "Tomate", "Cantidad": 15},  
14             {"Nombre": "Pepino", "Cantidad": 50}  
15         ]  
16     }  
17 }  
18 }
```

También se pueden usar herramientas online como <http://jsonviewer.stack.hu/>

## 16. Reglas de sintaxis de JSON

- Los datos son en pares nombre/valor
  - Los datos son separados por comas
  - Los objetos van entre llaves
  - Los array van entre corchetes.
- 
- **JSON Data:** Un par nombre/valor consiste en un nombre (en comillas dobles), seguido por doble punto y el valor:  
`"firstName": "John"`
  - **JSON Values:** Los valores pueden ser: Un número (entero o flotante), un string (en comillas doble), un boolean (true o false), un array (en corchetes), un objeto (en llaves) o null.
  - **JSON Objects:** Escritos en llaves  
`{"firstName": "John", "lastName": "Doe"}`
  - **JSON Arrays:** Escritos en corchetes  
`"employees": [  
 {"firstName": "John", "lastName": "Doe"},  
 {"firstName": "Anna", "lastName": "Smith"},  
 {"firstName": "Peter", "lastName": "Jones"}]`

]

**Nota:** La extensión para archivos JSON es .json

**Ejercicio 8**

Representar la misma información del ejercicio 6 en un archivo JSON.

**17. JSON schema**

Similarmente a XML schema, JSON permite describir el formato de los datos. Ver ejemplo:  
<http://json-schema.org/example1.html>

**Ejercicio 9**

Realizar el JSON schema para el problema del ejercicio 5.

Todos los archivos deben ser colocados en una carpeta y subida a la plataforma en formato comprimido.