Travelnet Tecnologías

Web S.A de C.V.

Comité de Estandarización

de Procesos de Desarrollo

Categoría: Norma de Trabajo

**Proyecciones XML**

**Versión 1.0**

|  |  |  |  |
| --- | --- | --- | --- |
| SISTEMA: | **TRAVELNET** | 17/04/17 | MARCO REF. |
| AUTOR: R. BERMÚDEZ | PROYECCIONES XML | | REVISÓ: S. MONTIEL |
|  | **C. NDX1** |

**Prólogo**

En este documento se explican los fundamentos básicos de los operadores de manipulación y análisis de estructuras XML de la herramienta Proyecciones XML (PXML). No se profundiza en los conceptos bajo los cuales se diseñó, sino el objetivo principal es exponer a los desarrolladores el cómo utilizar esta herramienta correctamente.

Proyecciones XML es un conjunto de funciones diseñadas sobre PHP, que permiten codificar de forma sencilla las operaciones para proyectar los datos de un XML en una estructura de datos distinta y hacer evidente la relación de los datos de entrada con los datos de salida esperados. A lo largo de su implementación se ha demostrado la sustancial reducción de líneas de código necesarias para el análisis de los datos a procesar y la sencillez con que se usan los operadores sugeridos.

Para hacer mejor uso de PXML por los programadores se recomienda tengan un dominio básico de los siguientes temas: Estructuras de árbol, XML Path Lenguaje (XPath), una noción básica del uso de funciones recursivas, PHP y el uso de arreglos asociativos. Se espera que a futuro esta herramienta sea de utilidad a los programadores en sus actividades laborales dentro de Travelnet, permitiéndoles desarrollarse profesionalmente y proporcionando mayor calidad a sus trabajos.

**Índice de contenido**

**Prólogo.......................................................2**

1. **Introducción.............................................4**
2. **Proyecciones.............................................4**

2.1 Nociones básicas**.....................................5**

2.2 Lenguaje de direccionamiento XML: XPath**................9**

2.3 Propiedades de XML y XPath**...........................10**

2.4 Descriptor de la Proyección**..........................12**

2.4.1 **.**Trayectoria de la proyección**.....................12**

2.4.2 Propiedades de la trayectoria de la proyección...**. 12**

2.4.3 Mapa de la proyección**........................... 13**

2.4.4 Propiedades del mapa de la proyección**............. 13**

1. **La clase XMLProjection**

3.1 XMLProjection:: construct() (Inicialización).**....... 16**

3.1.1 Descripción**................................. 16**

3.1.2 Parámetros**.................................. 16**

3.1.3 Ejemplos.............**........................ 16**

3.2 XMLProjection::runProjection()**........................ 17**

3.2.1 Descripción**................................. . 17**

3.2.2 Parámetros**................................. . . 17**

3.2.3 Valores resultantes**......................... . . 17**

3.3 Sintaxis y semántica del DP**......................... . 18**

3.4.1 >\_MAP y >\_FROOT**............................. . 21**

3.4.2 >\_CSET**.................................... . . 24**

3.4.3 >\_CONS**.................................... . . 27**

3.5 Descriptores de proyección anidados**................. . . 28**

3.6 Recomendaciones durante la codificación**............. . . 35**

3.6.1 Variables**................................ . . . . .. 35**

3.6.2 Arreglos**................................. . . . . . .36**

**Índice de Imágenes**

**Índice de códigos fuente**

**Bibliografía**

**Capítulo I. Introducción**

Durante el trabajo con una estructura XML, acceder y tomar los datos que contiene puede ser una tarea bastante trivial cuando la estructura de datos que representa es simple, es apenas necesario utilizar una expresión XPath sin filtros y acceder a los datos con algunos ciclos de repetición. Sin embargo, al incrementar el grado y altura del árbol de datos contenido en determinado XML, la codificación necesaria para su análisis se torna cada vez más compleja y se hace necesario encontrar formas sistemáticas de resolver su procesamiento0.

Distintas instituciones como el World Wide Web Consortium (W3C), IETF, IEEE, etc. se han preocupado por este problema1 desarrollando algunas herramientas de procesamientoy creando formatos de datos cada vez más sencillos de utilizar que en consecuencia faciliten la integración de distintos sistemas.

En este contexto, se ha desarrollado la herramienta de análisis de datos llamada Proyecciones XML (PXML) diseñada en PHP y apoyada en el funcionamiento de las clases DOMXPath y DOMDocument; ambas, implementaciones de los protocolos XML Path Lenguaje (XPath) y Document Object Model (DOM) sobre PHP. Al basarse en ellos (tanto conceptual como funcionalmente), PXML hereda la robustez necesaria para facilitar el procesamiento y análisis de datos contenidos en un XML. Con PXML no se pretende sustituir el uso de las clases DOMDocument o DOMXPath, sino apoyarse en ellas para potenciarlas en el uso particular que se les da al proyectar los datos de una estructura a otra.

El principal enfoque de PXML es mejorar la calidad de codificación de las operaciones de transformación de estructuras de datos arbóreas, que permitan concentrar los esfuerzos del programador en el análisis e intelección de los datos con que trabaja, hacer explicitas las relaciones entre los datos de distintos servicios/operaciones y ofrecer una interfaz que centralice el manejo de errores durante el procesamiento de las proyecciones.

Se resalta que para la implementación de ésta clase es necesario utilizar el núcleo del motor Zend 5.6 de PHP (PHP v5.6.x) que implementa la versión 1.0 de XML Path Language ([XPath)](http://php.net/manual/es/class.domxpath.php) y el núcleo nivel 3 de Document Object Model([DOM).](http://php.net/manual/es/book.dom.php)

**Capítulo II. Proyecciones**

A continuación se presentaran los conceptos básicos necesarios para utilizar correctamente la norma de trabajo Proyecciones XML y que en caso de ser necesario abren el camino para profundizar en los temas tratados.

0  No es de sorprender que en muchas ocasiones, los programadores que se enfrentan a enormes estructuras XML durante su análisis, se preocupen más por las estructuras de control que procesarán los datos, que por entender la información y forma que estos tienen.

1  La necesidad de intercomunicación de distintos sistemas que trabajan con distintas estructuras de datos ha dado pie a buscar mejorar los métodos de transformación de estructuras de árbol. Vale la pena referenciar a *XPath, XSLT y XQuery*; tres protocolos de procesamiento de datos creados por el World Wide Web Consortium en pos de este objetivo.

**2.1 Nociones básicas**

Al trabajar con estructuras de datos codificadas en XML se está trabajando de forma abstracta con estructuras de árbol [[2](https://www.w3.org/TR/2004/REC-DOM-Level-3-Core-20040407/introduction.html#ID-E7C30821),[3](https://www.w3schools.com/xml/xml_tree.asp)], es decir, con árboles de datos. Si el trabajo o requerimiento es tomar datos de un XML de entrada para presentarlos en una estructura de salida diferente, y si esa estructura de salida es un árbol de datos, entonces el problema es *la transformación de un árbol en otro*. A esto se ha llamado *proyección* y se define de la siguiente forma:

Definición 1.- Una **proyección** es el conjunto de operaciones aplicadas para presentar los datos necesitados de una estructura de árbol inicial en una estructura de árbol final.

Formalizamos la definición de la función de proyección P con cualquier conjunto de operaciones de la siguiente manera:

E1.

Dónde:

* *AI* *(Árbol inicial)* Es la estructura a la que se aplicará la transformación/proyección según las operaciones contenidas en DP.
* *DP (Descriptor de la Proyección)* es la estructura que describe las operaciones necesarias para proyectar los datos de *AI* en *AF*.
* *AF (Árbol final)* es la estructura resultante de la operación de proyección aplicada sobre AI.

Al tratar con un problema de transformación de estructuras de árboles es fundamental distinguir los tres elementos mencionados anteriormente. Con un ejemplo se aclarará esto.

Ejemplo 1. A través de un sistema web, un restaurante recibe diariamente una lista de productos de su proveedor de insumos, de estos solo requiere conocer los apartados de frutas y verduras, pues es un restaurante vegetariano. En la lista final de productos, al gerente del restaurante no le interesa clasificar los insumos como frutas o verduras, sino en “Aprobados para la compra” o “Por revisar”. Los aprobados para la compra son aquellos cuyo precio por unidad es menor a 20 pesos y los clasificados como “por revisar” aquellos cuyo precio es mayor o igual a 20 pesos. De cada producto solo le interesa conocer los siguientes datos: nombre, precio, no. de unidades disponibles y unidad de medida. Propóngase una estructura de salida que represente los datos necesitados y muéstrese cuál es el resultado de utilizarla para representar la información requerida.

Un ejemplo de uno de los XML que recibe a diario el restaurante en su sistema es el siguiente:

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

<Alimentos>

<Frutas>

<Producto Unidad="Pz">

<Nombre>Sandía</Nombre>

<CostoUnidad>10.00</CostoUnidad>

<Disponibles>10</Disponibles>

<Calorías>1500</Calorías>

</Producto>

</Frutas>

<Verduras>

<Producto Unidad="Kg">

<Nombre>Jitomate</Nombre>

<CostoUnidad>40.00</CostoUnidad>

<Disponibles>3</Disponibles>

<Calorías>300</Calorías>

</Producto>

</Verduras>

<Carnes>

<Producto Unidad="Kg">

<Nombre>Res</Nombre>

<CostoUnidad>160.00</CostoUnidad>

<Disponibles>1</Disponibles>

<Calorías>500</Calorías>

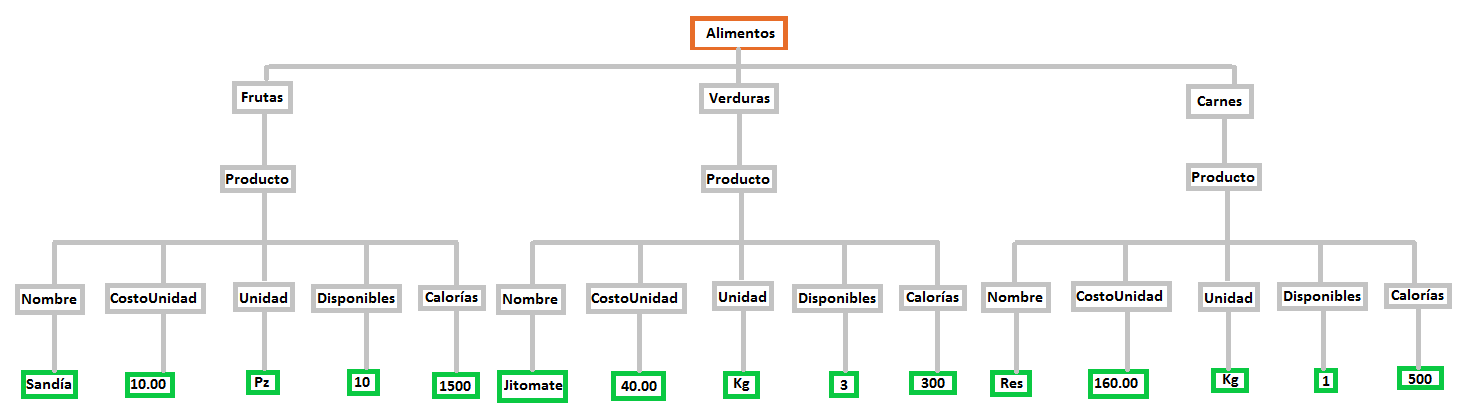
</Producto>

</Carnes>

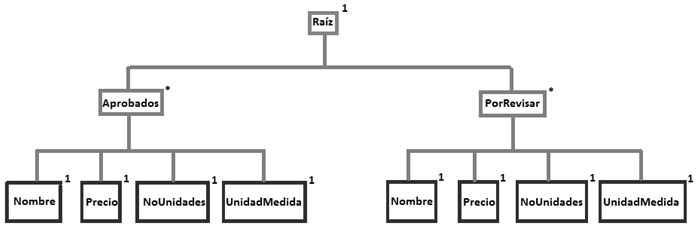
</Alimentos>

X2.1 XML de productos.

Y de manera abstracta, la estructura de datos que representa (Árbol inicial) es:

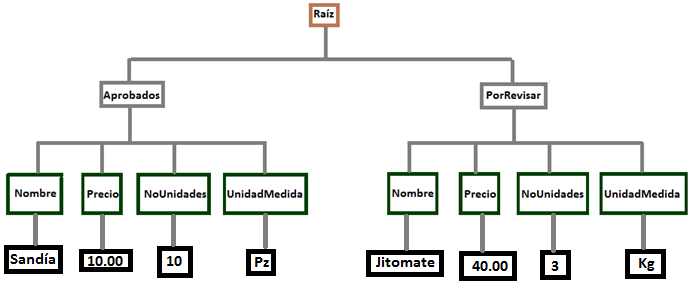
 Los datos requeridos de nombre, precio, no. de unidades disponibles y unidad de medida se encuentran contenidos en los nodos <Nombre>, <CostoUnidad>, <Disponibles> y @Unidad4 respectivamente, todos ellos definidos dentro de los nodos <producto>**.** El dato de <Calorías> y el contenido de <Carnes> u otros posibles nodos del mismo nivel se ignorarán. Para este caso, se definió una estructura abstracta sobre de la cual se proyectaran los datos que necesita el gerente del restaurante (Árbol Final):

A2.1 Árbol asociado al XML de productos X2.1.



A2.2. Estructura de árbol final.

Revisando someramente la relación de los datos y la descripción del problema que funge como descriptor de la proyección, se entiende que por cada nodo <producto> dentro de los nodos <Frutas> o <Verduras> se generará un nodo <Aprobados> o <PorRevisar> teniendo en cuenta el umbral de la clasificación sobre del precio de 20 pesos. Los datos contenidos en los nodos <Nombre>, <CostoUnidad>, <Disponibles> y @Unidad se mapearan sobre de los nodos <Nombre>, <Precio>, <NoUnidades> y <UnidadMedida> respectivamente, estos se encontrarán contenidos en los nodos <Aprobados> o <PorRevisar>, según sea el caso. Así, el árbol resultante de proyectar los datos recibidos en el XML inicial sobre la estructura final propuesta en la imagen A2.2 es el siguiente:

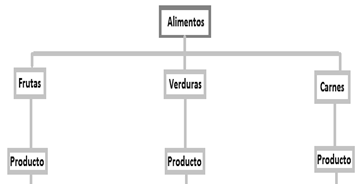


A2.3. Proyección de datos sobre de la estructura de árbol final A2.2.

4  El símbolo “@” detrás de un nombre simboliza que el nodo mencionado se trata de un [atributo](https://www.w3schools.com/xml/xpath_syntax.asp).

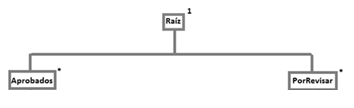
En la esquina superior derecha de cada nodo del árbol final propuesto (Imagen A2.2), se escribió un signo que representa la cardinalidad de los nodos del mismo tipo, se usa “\*” cuando el nodo de determinado tipo puede aparecer 0 o N veces y “1” cuando el como solo puede aparecer solo una vez.

De manera general e independientemente del número de productos que se tengan, se puede definir la relación de los nodos de los arboles inicial y final con las siguientes flechas:



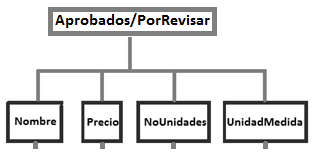
**->Productos con precio mayor o igual a 20 pesos**

**->Productos con precio menor a 20 pesos**



AR2.1 Relación estructural.

 Y para los nodos contenidos en cada elemento <producto> el mapeo es el siguiente:



AR2.2 Mapeo de los datos

Como se verá más adelante, PXML proporciona un lenguaje para expresar de forma sencilla estás relaciones resaltadas gráficamente por las flechas coloreadas en los diagramas anteriores.

**2.2 Lenguaje de direccionamiento XML: XPath.**

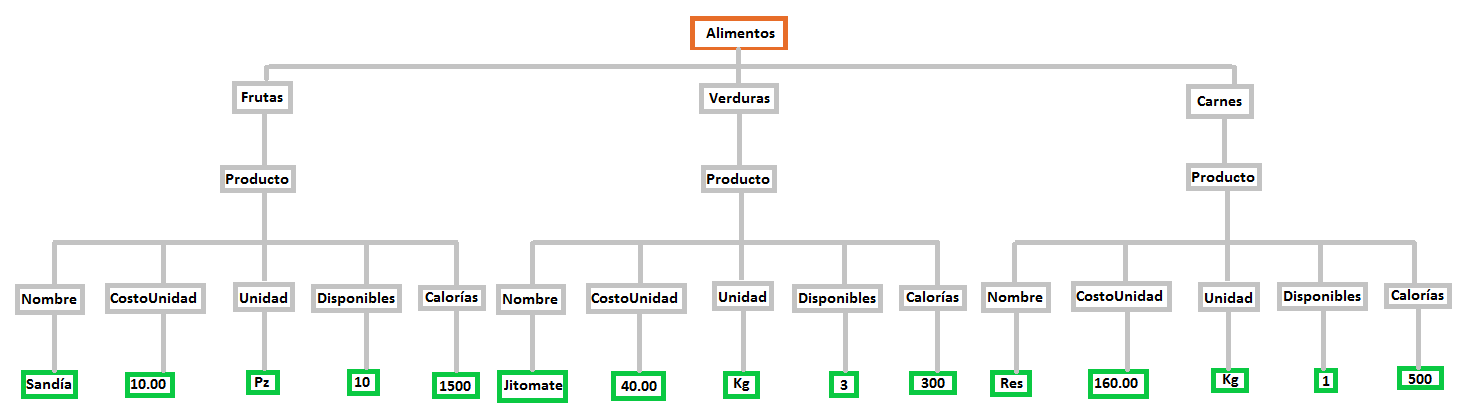
Uno de los problemas surgidos durante la proyección es conocer ***la posición y estructura*** de los datos requeridos en el árbol inicial. Esta es la función de XPath, proporcionar un lenguaje de direccionamiento o selección de estructuras dentro de un XML. Es vital comprender la importancia que tiene su uso, pues el utilizar una expresión/instrucción XPath adecuada para la selección de los datos necesitados de cierto XML, puede ahorrar al programador y al programa bastantes rutinas de procesamiento de datos, donde su mantenimiento se puede complicar en función de la altura y grado de la estructura a procesar.

Por el limitado espacio del que se dispone para este texto es imposible desenvolver los conceptos que acompañan a XPath, se recomienda ampliamente leer el [tutorial](https://www.w3schools.com/xml/xpath_intro.asp) proporcionado por el W3C, donde se explican los fundamentos para usar este lenguaje en beneficio propio y de los proyectos que desarrolle.

Baste mostrar algunos ejemplos de cómo una expresión XPath puede codificar un enunciado completo. Retomando el Ejemplo 1 de la anterior sección, en los requerimientos se describe que solo se necesitan los datos de los elementos <Frutas> y <Verduras> e ignorar las demás secciones; utilizando los operadores sobre los [ejes](https://www.w3schools.com/xml/xpath_axes.asp) de estos elementos, la expresión XPath que los selecciona es la siguiente:

**//Alimentos/\*[self::Frutas|self::Verduras] (XQ1)**

Gráficamente se expresa de la siguiente forma:



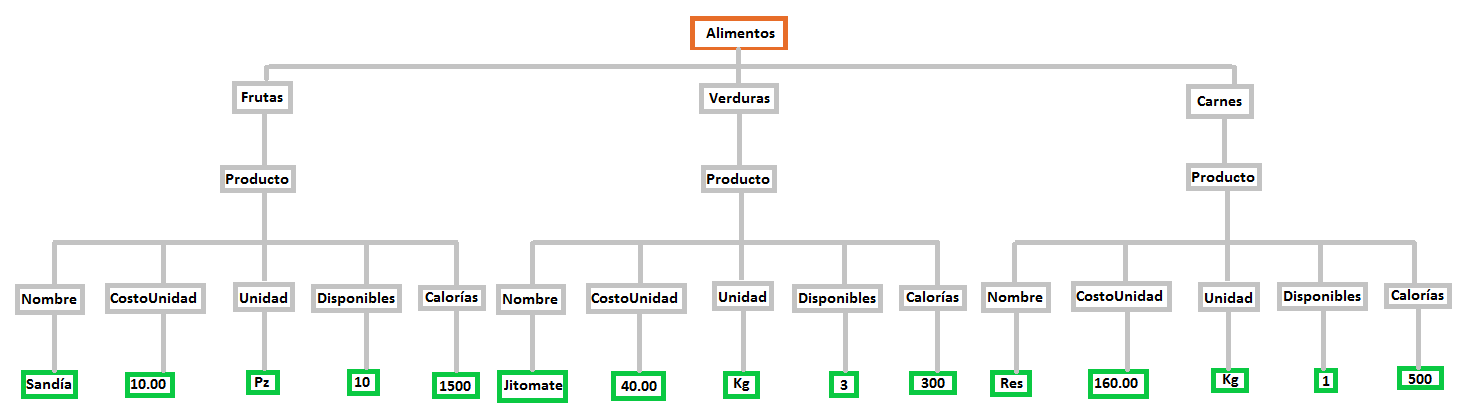
AQ2.1. Selección de frutas y verduras

Ahora, si el enunciado fuera: seleccionar todos los productos de frutas y verduras cuyo precio es menor a veinte pesos y el número de unidades disponibles sea mayor a cero, la expresión que direcciona a la estructura que contiene estos datos podría ser:

**(XQ2)**

**//Alimentos/\*[self::Frutas|self::Verduras]/Producto[Disponibles>0 and CostoUnidad<20]**

Gráficamente la selección se representa de la siguiente manera:



AQ2.2 Productos en frutas y verduras con precio menor a 20 pesos

Si bien con una expresión XPath se puede determinar la posición y estructura de los datos en el árbol inicial, falta determinar cuál será su destino o posición en el árbol final.

**2.3 Propiedades de XML y XPath**

Vale señalar algunas de las nociones formales de los elementos pertenecientes a un XML, para después definir las propiedades de los elementos del descriptor de la proyección. Siempre que se trabaje con un XML debe tomarse en cuenta lo siguiente:

* + - 1. **Propiedades de una estructura XML.** Considerando un XML como estructura de árbol:
         1. En un XML solo puede existir un único nodo raíz que es de tipo elemento*.*
         2. Todo nodo rama es de tipo elemento o atributo.
         3. Todo nodo hoja es de tipo texto o elemento vacío.
      2. **Propiedades de las estructuras internas de un XML.** Considerando los tipos estructuras (nodos) dentro de un XML; documento, elementos, atributos y textos:
         1. Todo nodo de tipo elemento puede tener un grado mayor o igual a 0.
         2. Un nodo de tipo elemento en caso de tener grado mayor a 0, sus hijos pueden ser nodos de tipo atributo, etiqueta o texto.
         3. Todo nodo de tipo atributo tiene un grado igual a 1.
         4. Un nodo de tipo atributo tiene como hijo un nodo de tipo texto.
         5. Todo nodo de tipo texto tiene un grado igual a 0.
         6. Nodos elemento con igual nombre de etiqueta y nivel de profundidad (nodos hermanos) pueden tener distintos grados.

A continuación se ejemplifican las reglas mencionadas anteriormente con el siguiente XML:

<Texto>

<Palabra orden="1">

Hola

</Palabra>

<Palabra>

Mundo

</Palabra>

<Palabra/>

</Texto>

X2.2 XML de Palabras

Palabra

(N: Hoja, T: Texto, G: 0)

(N: Hoja, T: Texto, G: 0 0)

(N: Rama, T: Atributo, G: 1)

Palabra

(N: Rama, T: Elemento, G: 1)

(N: Hoja, T: Elemento, G: 0)

1

Hola

Palabra

(N: Rama, T: Elemento, G: 2)

Orden

Texto

(N: Raíz, T: Elemento, G: 3)

(N: Hoja, T: Texto, G: 0)

Mundo

-N: Nodo.

-T: Tipo de nodo.

-G: Grado.

A2.4 Estructura de Árbol asociada al documento XML X2.2.

* + - 1. **Propiedades de consulta de las expresiones XPath.** Téngase en cuenta lo siguiente:

1. Una consulta de una expresión XPath ejecutada sobre un nodo atributo o texto tiene como resultado un conjunto vacío de nodos.
2. De la premisa anterior se deduce que no se puede direccionar a través de una expresión XPath un nodo texto contenido en un nodo atributo.
3. Todo nodo dentro de un XML (exceptuando los nodos tipo texto dentro de los nodos atributo) se pueden seleccionar con una expresión XPath que tenga como componente primario la raíz del árbol XML.

**2.4 Descriptor de la proyección**

**2.4.1 Trayectoria de la proyección**

Debe notarse que entre el árbol final y el árbol inicial en una operación de proyección existe una *relación principal*, que no relaciona directamente los nodos hoja del árbol inicial con la etiqueta y posición que tendrán en el árbol final, sino relaciona los nodos rama de árbol inicial que de manera indirecta contienen los datos requeridos con la etiqueta de estructura sobre de la cual se proyectarán esos datos. Por ejemplo, en el Ejemplo 1 planteado en la sección 2.1, no tendría caso que se generaran nodos <Aprobados> o <PorRevisar> dentro de la lista final si no existieran elementos <Frutas> y <Verduras> dentro de <Alimentos> (Véase la imagen AR2.1). Esta relación puede señalarse a partir de una expresión XPath que se ha denominado ***Trayectoria de la proyección (TP).***

**2.4.2 Propiedades de la trayectoria de la proyección**

Una expresión XPath puede o bien direccionar nodos hoja (Terminales) o nodos rama (subárboles) en un XML determinado, debido a esto debe tenerse en cuenta lo siguiente:

1. La expresión XPath que represente a la TP DEBE direccionar nodos de tipo elemento y NUNCA a nodos atributo. (Por las propiedades 2.A, 3.B y 3.C)

En este contexto la TP debe direccionar a nodos rama, esto “debe” ser cuando menos de manera abstracta, ya que en ocasiones los nodos elemento que se consideraron como rama en determinado XML pueden cambiar a ser hoja o nulos y con esto el árbol final será un árbol vacío (Véase la propiedad 2.A).

1. La trayectoria de la proyección se debe hacer explícita *la relación entre las raíces de los árboles inicial y final* (Véase la propiedad 1.A).
2. La metaetiqueta (en el descriptor de la proyección) a la que se asocie la trayectoria de la proyección, tendrá el mismo grado en su estructura resultante (árbol final) que el número de nodos en el árbol inicial que emparejaron en la consulta de la TP.

**2.4.3 Mapa de la proyección**

Una vez definida la relación principal por la TP, bastará tomar los datos necesitados de los nodos que emparejaron con la trayectoria de la proyección. Entonces, se define el ***Mapa de la proyección (MP)*** como el conjunto de asociaciones de direcciones XPath (perteneciente al árbol final) con un elemento nominal o abstracto (perteneciente al árbol inicial), donde ese elemento nominal determina la etiqueta y posición de los datos en el árbol final y se ha definido como ***metaetiqueta,*** mientras la expresión XPath indica la posición y el origen de los datos en cada nodo rama o subárbol que emparejo en la TP. Cada asociación se tiene la siguiente estructura:

**(R1)**

**{Metaetiqueta del árbol final <- Expresión XPath}**

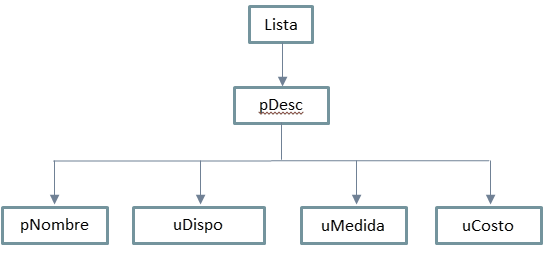
**2.4.4 Propiedades del mapa de la proyección**

Téngase en cuenta los siguientes puntos al determinar el mapa de la proyección:

1. El conjunto de asociaciones del mapa de la proyección no puede ser vacío.
2. Las expresiones XPath asociadas a una metaetiqueta en el mapa de la proyección DEBEN direccionar a nodos hoja o a nodos cuyos hijos sean directamente nodos terminales.
3. La cardinalidad de cada una de las expresiones XPath que participan en el MP puede ser mayor o igual a cero, siempre y cuando cada nodo perteneciente a ese conjunto esté en acuerdo con el punto anterior.

Así, el descriptor de la proyección será la estructura cuyos elementos sean el mapa y la trayectoria de la proyección, que son los elementos necesarios para ejecutar la operación de transformación.

Ejemplo 2. Para ejemplificar el uso de los conceptos de MP y TP, supóngase que otro restaurante recibió la misma lista de insumos que el restaurante vegetariano del Ejemplo 1 de la sección 2.1, de la cual solo le interesa conocer los datos <Nombre>, <CostoUnidad>, @Unidad y <Disponibles> de los productos cuya disponibilidad es mayor a 0 sin importar si se trata de carnes, frutas o verduras. Una estructura de árbol que represente esos datos podría ser la siguiente:



A2.5 Árbol Final

La dirección XPath que selecciona los nodos <Producto> dentro <Carnes>, <Frutas> y <Verduras> con <Disponibles> mayor a cero (Véase en el Cuadro C2.1 de la sección 2.1) es la siguiente:

**TP:**

**(XQ3)**

**//Alimentos/\*/Producto[Disponibles>0]**

Y por cada nodo <Producto> que empareje con la consulta XQ3, debe generarse un nodo <pDesc> en el árbol final, con la siguiente estructura:

**pNombre <- Nombre (Q5)**

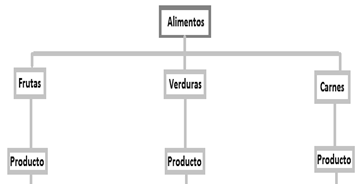
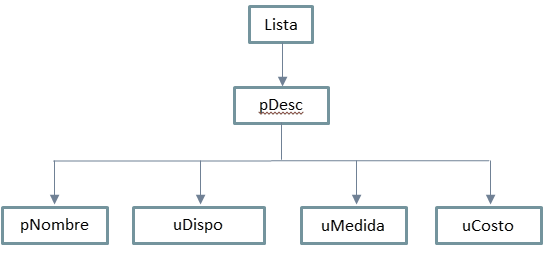
**uDispo <- Disponibles (Q6)**

**uMedida <- @Unidad (Q7)**

**uCosto <- CostoUnidad (Q8)**

**MP:**

Gráficamente se expresa de la siguiente manera:



**Q3**

**Q5**

**Q6**

**Q7**

**Q8**

**TP:**

**MP:**

**Relación implícita entre raíces**



AR2.3 Relación de proyección.

A manera de pseudocódigo el descriptor de la proyección quedaría ensamblado de la siguiente forma:

**//Lista/**pDesc/pDesc**:{MP: {pNombre <- Nombre**

**uDispo <- Disponibles**

**uMedida <- @Unidad**

**uCosto <- CostoUnidad},**

**TP: //Alimentos/\*/Producto[Disponibles>0]}**

PS2.1. Pseudocódigo del descriptor de la proyección.

Como conclusión de lo anterior se tiene que con un sencillo lenguaje de asociaciones se puede expresar como debe ser la proyección de una estructura de árbol XML sobre otra. En el siguiente capítulo se definirá la formalmente el lenguaje utilizado para el descriptor de la proyección y la clase XMLProjection, que en conjunto son capaces de expresar complejas operaciones de transformación con apenas unas cuantas líneas de código, produciendo mejoras sustanciales a la calidad y mantenimiento de los sistemas que dependen de estructuras de datos provenientes de agentes externos con estructuras distintas a las necesarias.

**Capítulo III. La clase XMLProjection**

Los conceptos mostrados en el anterior capítulo fueron llevados a la programación mediante un conjunto de funciones agrupadas en una clase llamada XMLProjection, parte sustancial del motor de las proyecciones, cuya estructura es la siguiente:



C3.1 Clase XMLProjection.

A continuación se describirán los métodos de propósito general \_\_construct() y projection(), mientras que los métodos \_CONS(), \_CSET() y \_setSubArray() NO DEBERÍAN intentar utilizarse, dado que estas funciones son piezas de código especializadas para ser utilizadas por el motor de las proyecciones, que está representado directamente por el método IAPI\_XMLProjection::runProjection().

**3.1 IAPI\_XMLProjection::\_\_construct()**

**3.1.1 Descripción**

IAPI\_XMLProjection::\_\_construct([array $xmlns])

Crea instancias de la clase IAPI\_XMLProjection. Si el/los XML a procesar por el motor de las proyecciones poseen un espacio de nombres, estos DEBEN ser indicados en el argumento del constructor, en caso contrario se puede inicializar un objeto XMLProjection sin pasar argumentos al constructor.

**3.1.2 Parámetros**

**xmlns**

El argumento xmlns es un arreglo asociativo, donde cada pareja clave => valor significan:

[Prefijo del espacio de nombres => URI del espacio de nombres]

**3.1.3 Ejemplos**

Si se fuera a procesar con un objeto de la clase IAPI\_XMLProjection un XML como el que se presenta a continuación:

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

<raiz xmlns:tag="http://www.example.org" xmlns:snd="https://www.ietf.org/rfc/rfc2119.txt">

<tag:level1>

<snd:level2>

¡Hola mundo!

</snd:level2>

</tag:level1>

</raiz>

X3.1 Hola mundo codificado en XML.

La clase debería ser inicializada de la siguiente forma:

<?php

include\_once 'IAPI/XMLProjection';

/\*\*Declaración del espacio de nombres del XML Ejemplo 1 \*/

$xmlns = array (

'tag' => 'http://www.example.org',

'snd' => 'https://www.ietf.org/rfc/rfc2119.txt'

);

$pxmlEngine = new IAPI\_XMLProjection($xmlns);

?>

PX3.1 Inicialización de una instancia de la clase IAPI\_XMLProjection.

**3.2 IAPI\_XMLProjection::projection()**

**3.2.1 Descripción**

IAPI\_XMLProjection::projection(array $projectionDescriptor, string $xmlStruct): array

Esta función ejecuta la proyección o transformación de una estructura de árbol XML a otra de acuerdo al descriptor de la proyección $projectionDescriptor y el XML de entrada $xmlStruct (Para más información véase la sección 2.1 de este documento titulada <<Nociones básicas>>).

**3.2.2 Parámetros**

**xmlStruct**

Es la cadena de texto que contiene la estructura XML a procesar.

**projectionDescriptor**

Es un arreglo asociativo que define el descriptor de la proyección.

**3.2.3 Valores resultantes**

Un arreglo que contiene la estructura resultante de aplicar la proyección al XML.

**3.3 Sintaxis y semántica del descriptor de la proyección (DP)**

En general la sintaxis de un DP se muestra a continuación:

<?php

$dpMain = [

'>\_MAP' => [

],

'>\_FROOT' => ''

];

?>

PX3.2 Estructura general del descriptor de la proyección.

El descriptor de la proyección se compone de *operaciones* y *asociaciones.* La sintaxis de una operación PXML es:

**(D2)**

**Operación PXML**:

Dónde:

* . Es una clave de tipo string, se utiliza para definir las *palabras reservadas* que indican la operación a ejecutar.
* . Es un valor donde tipo y formato de dato aceptado depende de la operación a ejecutar. Define los argumentos de la operación.

Dentro del mapa de la proyección (>\_MAP), cada pareja [clave => valor], tiene el siguiente significado:

**(D3)**

**Asociación:**

Donde:

* . Es una clave de tipo string o entero. Cada clave describe la estructura que tendrá el árbol final, indicando el nombre y posición que ocupa el elemento referenciado en un determinado nivel del árbol final.
* Es un valor de tipo string o array. Cuando este valor es de tipo string, se interpretará como una **expresión XPath** que direcciona a determinado dato o datos contenidos en $xmlStruct. En este caso, estás expresiones XPath DEBEN cumplir con las propiedades señaladas la sección 2.4.4 <<Propiedades del mapa de la proyección>>**.** Mientras que si se define un array se interpretará como una **Operación XML.**

Las palabras reservadas utilizadas en el descriptor de la proyección se encuentran en el siguiente cuadro:

|  |  |  |
| --- | --- | --- |
| Instrucción | Significado | Estructura |
| >\_FROOT | (Del inglés ***fr****om r****oot –****Desde la raíz****-***) Con esta instrucción se señala la dirección XPath que representa la ***trayectoria de la proyección.*** | Donde:   * Es una cadena de texto (tipo string) con la expresión XPath que direcciona el conjunto de nodos a procesar como subárboles del XML definido en $xmlStruct. Esta dirección XPath DEBE cumplir con las propiedades señaladas en la sección 2.4.2 <<Propiedades de la trayectoria de la proyección >>.   Se puede ver esta instrucción como sinónimo de un *foreach* que recorrerá el conjunto de nodos que direccione , y con cada iteracion que corresponde con un nodo de ese conjunto, crear una estructura como la señalada en >\_MAP a con datos provenientes del XML inicial. |
| >\_MAP | (Del inglés ***map****ping* ***–****trazado de mapas o mapeo****-***) Con esta instrucción se señala el arreglo que contiene el conjunto de asociaciones (D3) que forman parte del ***mapa de la proyección.*** (Para más información véase las secciones 2.4.3 y 2.4.4) | Donde:   * Representa una metaetiqueta. * Representa una metaestructura.   Contiene la estructura que tendrá el arreglo final con la descripción de las consultas que obtendrán los datos solicitados del XML inicial. |
| >\_CONS | (Del inglés ***cons****tant* ***–****constante****-***) Esta instrucción se utiliza para asociar a una metaetiqueta con un valor constante. | Donde:   * Representa una metaetiqueta. * Representa una metaestructura. * Es un dato constante que puede ser de cualquier tipo permitido en PHP. |
| >\_CSET | (Del inglés ***c****ountable* ***set*** *–conjunto contable-*) Esta instrucción sirve para agrupar y asociar los datos resultantes de una consulta a una metaetiqueta. Es sumamente útil cuando la cardinalidad del conjunto de nodos resultante de la consulta varía en el intervalo siendo un valor entero mayor 0.  Agrupa los valores de un conjunto de datos en una cadena de texto separándolos por un símbolo (token) cuando este es definido en o en un arreglo donde cada elemento del arreglo es un dato del conjunto resultante de la consulta cuando no es definido. | Donde:   * Representa una metaetiqueta. * Representa una metaestructura. * Representa una dirección XPath que DEBE cumplir con las propiedades señaladas en la sección 2.4.2. * Representa un símbolo separador (en inglés token). |

**3.4 Ejecución de una proyección**

A continuación se mostraran algunos ejemplos de cómo definir y ejecutar una proyección de datos.

**3.4.1 >\_MAP y >\_FROOT**

Ejemplo 3. Del siguiente menú en formato XML, obtenga un listado de las comidas disponibles (aquellas donde el elemento <Disponibilidad> es igual a “DISPONIBLE”), solo interesa recuperar el nombre de la comida (<Nombre>), su precio (atributo @neto en <Precio>) y moneda de cotización (atributo @moneda en <Precio>).

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

<Menu>

<Comida>

<Nombre calorias="300">Sandwitch</Nombre>

<Precio iva="16%" neto="30.00" moneda="MXN"/>

<Tipo>Desayuno</Tipo>

<Disponibilidad>DISPONIBLE</Disponibilidad>

</Comida>

<Comida>

<Nombre calorias="1300">Chilaquiles</Nombre>

<Precio iva="16%" neto="70.00" moneda="MXN"/>

<Tipo>Desayuno</Tipo>

<Disponibilidad>DISPONIBLE</Disponibilidad>

<Extra>Incluye jugo de naranja</Extra>

</Comida>

<Comida>

<Nombre calorias="2500">Corte Americano Rib Eye</Nombre>

<Precio iva="16%" neto="500.00" moneda="MXN"/>

<Tipo>Comida</Tipo>

<Disponibilidad>DISPONIBLE</Disponibilidad>

<Extra>Incluye copa de vino tinto libre elección</Extra>

<Extra>Incluye sopa azteca o caldo tlalpeño de entrada</Extra>

</Comida>

<Comida>

<Nombre calorias="2500">Brochetas de Camarón</Nombre>

<Precio iva="16%" neto="100.00" moneda="MXN"/>

<Tipo>Antojo</Tipo>

<Disponibilidad>AGOTADO</Disponibilidad>

</Comida>

</Menu>

X3.2 XML del menú de un restaurante.

Se buscará formar un array con la siguiente estructura:

$afComidas = array(

Indice1 => array(

'nombre' => ...,

'precio' => ...,

'moneda' => ...

),

Indice2 => array(

'nombre' => ...,

'precio' => ...,

'moneda' => ...

)

.

.

IndiceN => array(

'nombre' => ...,

'precio' => ...,

'moneda' => ...

)

)

PX3.3 Meta estructura del árbol final del Ejemplo 3.

De esta forma el descriptor de la proyección quedaría como en la variable $dpComidas del siguiente programa, Dando como resultado el arreglo de la figura PE3.1.

<?php

include\_once '../Ejemplo/Proyeccion.php';

/\*\*Se carga el XML en una variable tipo string.\*/

$xmlMenu = file\_get\_contents('Menu.xml');

/\*\*Inicializa el motor el intérprete de las proyecciones.\*/

$pxmlEngine = new IAPI\_XMLProjection();

/\*\*Se define el descriptor de la proyección.\*/

$dpComidas = [

'>\_MAP' => [

'nombre' => 'Nombre',

'precio' => 'Precio/@neto',

'moneda' => 'Precio/@moneda',

],

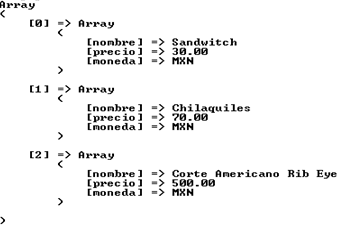
'>\_FROOT' => '//Menu/Comida[Disponibilidad="DISPONIBLE"]'

];

$afComidas = $pxmlEngine->runProjection($dpComidas, $xmlMenu);

echo print\_r($afComidas, true);

PX3.4 Solución del Ejemplo 3.



PE3.1 Resultado de la ejecución del programa PX3.4.

En resumen, tres nodos emparejaron con la traza de la proyección definida en >\_FROOT, dado que el cuarto nodo <Comida> en <Disponibilidad> contiene el valor de “AGOTADO”, por otra parte nótese como a través del símbolo “@” se pueden direccionar los atributos contenidos en un elemento.

Ejemplo 4. Suponga ahora que se requiere extraer el valor(es) del elemento(s) <Extra> en el XML, baste agregar la asociación de una metaetiqueta con la expresión XPath de direccione los elementos <Extra> en <Comida> dentro del arreglo del mapa de la proyección.

<?php

include\_once '../Ejemplo/Proyeccion.php';

/\*\*Se carga el XML en una variable tipo string.\*/

$xmlMenu = file\_get\_contents('Menu.xml');

/\*\*Inicializa el motor el intérprete de las proyecciones.\*/

$pxmlEngine = new IAPI\_XMLProjection();

/\*\*Se define el descriptor de la proyección.\*/

$dpComidas = [

'>\_MAP' => [

'nombre' => 'Nombre',

'precio' => 'Precio/@neto',

'moneda' => 'Precio/@moneda',

'extras' => 'Extra',

],

'>\_FROOT' => '//Menu/Comida[Disponibilidad="DISPONIBLE"]'

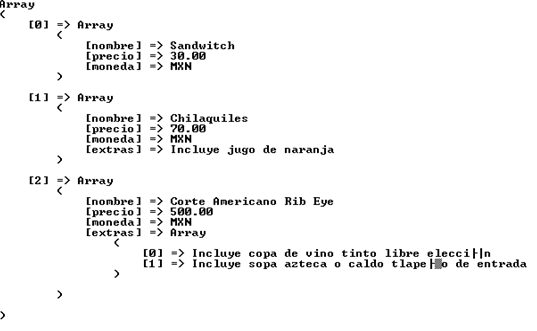
];

$afComidas = $pxmlEngine->runProjection($dpComidas, $xmlMenu);

echo print\_r($afComidas, true);

PX3.5 Solución del Ejemplo 4.

Sin embargo, el primer elemento <Comida> no contiene elementos <Extra>, mientras que el segundo y tercer elementos <Comida> contienen uno y dos elementos <Extra> respectivamente. Por lo que al ejecutar la proyección correspondiente, el resultado es:



PE3.2 Resultado de la ejecución del programa PX3.5.

En el arreglo resultante el primer índice no contiene la clave “extras”, mientras que el segundo se escribe un valor de tipo string y en el tercero un array con dos elementos. En general, el resultado de una asociación de una metaetiqueta con una expresión XPath se comporta de la siguiente forma:

1. Cuando la expresión XPath asociada a determinada metaetiqueta, no coincide con algún nodo durante la ejecución de la proyección, provoca que en el arreglo resultante no se cree la clave de esa metaetiqueta.
2. Cuando la expresión XPath asociada a determinada metaetiqueta, coincide con uno y solo un nodo durante la ejecución de la proyección, entonces se crea la clave de la metaetiqueta en el arreglo resultante con el valor del nodo en formato string.
3. Cuando la expresión XPath asociada a determinada metaetiqueta, coincide con más de un nodo durante la ejecución de la proyección, entonces se crea la clave de la metaetiqueta en el arreglo resultante con el valor de la clave en formato de arreglo; donde cada índice de este arreglo es el valor en formato de string de cada uno de los nodos que coincidió con la expresión XPath.

**3.4.2 >\_CSET**

Ejemplo 5. Si se quisiera forzar a inicializar la clave en el arreglo final aunque no emparejaran nodos durante la ejecución de la proyección con la expresión XPath asociada a su metaetiqueta, se puede utilizar la instrucción >\_CSET de la siguiente forma:

<?php

include\_once '../Ejemplo/Proyeccion.php';

/\*\*Se carga el XML en una variable tipo string.\*/

$xmlMenu = file\_get\_contents('Menu.xml');

/\*\*Inicializa el motor el intérprete de las proyecciones.\*/

$pxmlEngine = new IAPI\_XMLProjection();

/\*\*Se define el descriptor de la proyección.\*/

$dpComidas = [

'>\_MAP' => [

'nombre' => 'Nombre',

'precio' => 'Precio/@neto',

'moneda' => 'Precio/@moneda',

'extras' => ['>\_CSET' => ['Extra']],

],

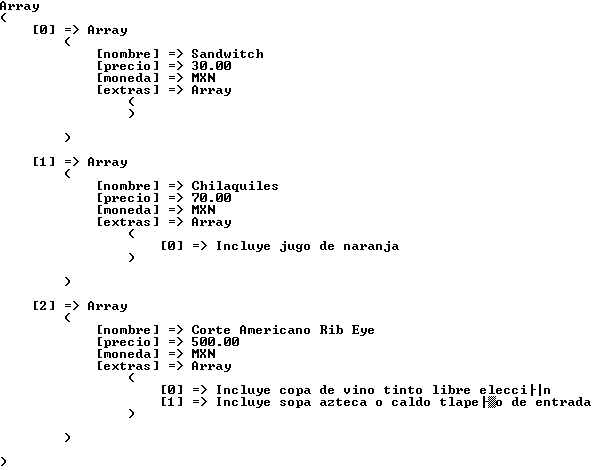
'>\_FROOT' => '//Menu/Comida[Disponibilidad="DISPONIBLE"]'

];

$afComidas = $pxmlEngine->runProjection($dpComidas, $xmlMenu);

echo print\_r($afComidas, true);

PX3.6 Solución del Ejemplo 5.



PE3.3 Resultado de la ejecución del programa PX3.6.

>\_CSET es una instrucción utilizada para ayudar a evitar condicionales de validación de tipos de datos o existencia de una clave dentro del arreglo resultante de la proyección, en caso de requerirse un análisis de los datos posterior a la ejecución de la proyección.

La instrucción >\_CSET permite formar un conjunto de valores en formato string a partir de un símbolo separador o identificador (token) como se muestra en el programa PX3.7. Dando como resultado el arreglo de la siguiente imagen, donde cada clave extra contiene un string, vacío en el caso de que ningún nodo empareje con la expresión XPath de la clave asociada o un string donde cada dato está separado por el identificador “”.

<?php

include\_once '../Ejemplo/Proyeccion.php';

/\*\*Se carga el XML en una variable tipo string.\*/

$xmlMenu = file\_get\_contents('Menu.xml');

/\*\*Inicializa el motor el intérprete de las proyecciones.\*/

$pxmlEngine = new IAPI\_XMLProjection();

$separador = '<->';

/\*\*Se define el descriptor de la proyección.\*/

$dpComidas = [

'>\_MAP' => [

'nombre' => 'Nombre',

'precio' => 'Precio/@neto',

'moneda' => 'Precio/@moneda',

'extras' => ['>\_CSET' => ['Extra', $separador]],

],

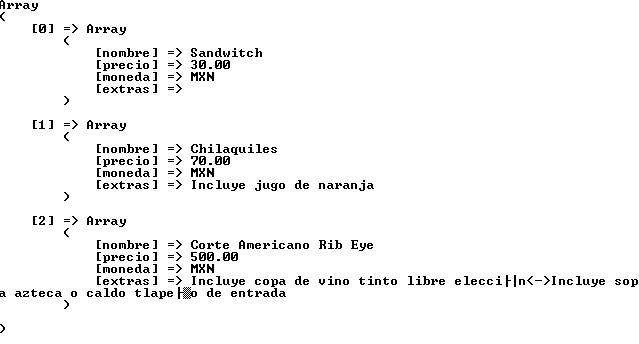
'>\_FROOT' => '//Menu/Comida[Disponibilidad="DISPONIBLE"]'

];

$afComidas = $pxmlEngine->runProjection($dpComidas, $xmlMenu);

echo print\_r($afComidas,true);

PX3.7 Utilización de >\_CSET con un símbolo separador.

****

PE3.4 Resultado de la ejecución del programa PX3.7.

**3.4.3 >\_CONS**

Ejemplo 6. Si se tuviera la necesidad de inicializar una clave con un valor por defecto que no proviene del XML, como por ejemplo colocar a cada índice en el arreglo resultante de las comidas la fecha en que se realizó la transformación, se puede utilizar la instrucción >\_CONS de la siguiente forma:

<?php

include\_once '../Ejemplo/Proyeccion.php';

/\*\*Se carga el XML en una variable tipo string.\*/

$xmlMenu = file\_get\_contents('Menu.xml');

/\*\*Inicializa el motor el intérprete de las proyecciones.\*/

$pxmlEngine = new IAPI\_XMLProjection();

/\*\*Valor constante.\*/

$fecha = date('d-m-y');

/\*\*Se define el descriptor de la proyección.\*/

$dpComidas = [

'>\_MAP' => [

'fecha' => ['>\_CONS' => $fecha],

'nombre' => 'Nombre',

'precio' => 'Precio/@neto',

'moneda' => 'Precio/@moneda',

'extras' => ['>\_CSET' => ['Extra']],

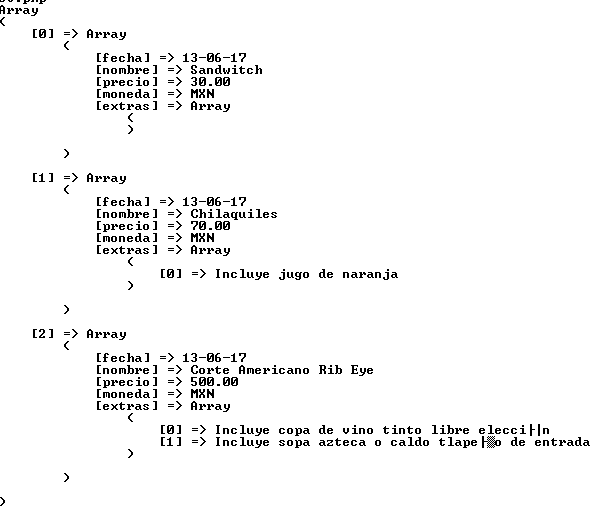
],

'>\_FROOT' => '//Menu/Comida[Disponibilidad="DISPONIBLE"]'

];

echo print\_r($pxmlEngine->runProjection($dpComidas, $xmlMenu), true);

PX3.8 Solución del Ejemplo 6.

Dando como resultado:

PE3.5 Resultado de la ejecución del programa PX3.8.

**3.5 Descriptores de proyección anidados**

Se han mostrado la sintaxis y semántica de las cuatro instrucciones que componen este pseudolenguaje llamado Proyecciones XML, y aún está por hacerse notar su funcionalidad de mayor importancia. Los ejemplos hasta ahora propuestos no rebasan los dos niveles de altura en el árbol final. ¿Qué sucedería en el caso de que la estructura del árbol final que se necesita requiriera de 3, 4, 5 o más niveles de profundidad? A continuación se propone un ejemplo donde se verá cómo se soluciona este problema.

Ejemplo 7. En un XML se almacenan los datos geográficos y socioeconómicos de los municipios de México, en los siguientes recuadros se muestra el XML que contiene los datos requeridos. Los municipios son agrupados en regiones económicas de los Estados y los Estados en regiones federales. Se necesita recuperar los datos de los conteos poblacionales hechos en el Estado de Aguascalientes entre 1990 y 2010 con una estructura como la mostrada en la figura PX3.9. Proponga una solución a este problema.

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

<RegionesFederales>

<Region nombre="Centro Norte">

<EntidadFederativa nombre="Aguascalientes">

<Capital>Aguascalientes</Capital>

<Propiedades>

<Propiedad tipo="ISO 3166-2">MX-AGU</Propiedad>

</Propiedades>

<RegionSocioeconomica nombre="El Bajío">

<Municipios>

<Municipio>

<Nombre>Aguascalientes</Nombre>

<Coordenadas LA="21.8833" LO="-102.3"/>

<DatosSocioEconomicos>

<ConteoPoblacion año="1995">

<conteo variable="Hombres" total="283536"/>

<conteo variable="Mujeres" total="299291"/>

</ConteoPoblacion>

<ConteoPoblacion año="2005">

<conteo variable="Hombres" total="349203"/>

<conteo variable="Mujeres" total="373840"/>

</ConteoPoblacion>

</DatosSocioEconomicos>

</Municipio>

<Municipio>

<Nombre>Asientos</Nombre>

<Coordenadas LA="22.2333" LO="-102.0833"/>

<DatosSocioEconomicos>

<ConteoPoblacion año="1995">

<conteo variable="Hombres" total="17954"/>

<conteo variable="Mujeres" total="17808"/>

</ConteoPoblacion>

<ConteoPoblacion año="2005">

<conteo variable="Hombres" total="19853"/>

<conteo variable="Mujeres" total="20694"/>

</ConteoPoblacion>

</DatosSocioEconomicos>

</Municipio>

<Municipio>

<Nombre>Calvillo</Nombre>

<Coordenadas LA="21" LO="-102"/>

<DatosSocioEconomicos>

<ConteoPoblacion año="1995">

<conteo variable="Hombres" total="25627"/>

<conteo variable="Mujeres" total="26031"/>

</ConteoPoblacion>

<ConteoPoblacion año="2005">

<conteo variable="Hombres" total="24018"/>

<conteo variable="Mujeres" total="26165"/>

</ConteoPoblacion>

</DatosSocioEconomicos>

</Municipio>

<Municipio>

<Nombre>Cosío</Nombre>

<Coordenadas LA="22.3677" LO="-102.2997"/>

<DatosSocioEconomicos>

<ConteoPoblacion año="1995">

<conteo variable="Hombres" total="5994"/>

<conteo variable="Mujeres" total="6142"/>

</ConteoPoblacion>

<ConteoPoblacion año="2005">

<conteo variable="Hombres" total="6540"/>

<conteo variable="Mujeres" total="7147"/>

</ConteoPoblacion>

</DatosSocioEconomicos>

</Municipio>

<Municipio>

<Nombre>Jesús María</Nombre>

<Coordenadas LA="21.9611" LO="-102.3433"/>

<DatosSocioEconomicos>

<ConteoPoblacion año="1995">

<conteo variable="Hombres" total="26909"/>

<conteo variable="Mujeres" total="27567"/>

</ConteoPoblacion>

<ConteoPoblacion año="2005">

<conteo variable="Hombres" total="40556"/>

<conteo variable="Mujeres" total="42067"/>

</ConteoPoblacion>

</DatosSocioEconomicos>

</Municipio>

<Municipio>

<Nombre>Pabellón de Arteaga</Nombre>

<Coordenadas LA="22" LO="-102"/>

<DatosSocioEconomicos>

<ConteoPoblacion año="1995">

<conteo variable="Hombres" total="15477"/>

<conteo variable="Mujeres" total="16173"/>

</ConteoPoblacion>

<ConteoPoblacion año="2005">

<conteo variable="Hombres" total="18799"/>

<conteo variable="Mujeres" total="20113"/>

</ConteoPoblacion>

</DatosSocioEconomicos>

</Municipio>

<Municipio>

<Nombre>Rincón de Romos</Nombre>

<Coordenadas LA="22.2166" LO="-102.3166"/>

<DatosSocioEconomicos>

<ConteoPoblacion año="1995">

<conteo variable="Hombres" total="19198"/>

<conteo variable="Mujeres" total="19554"/>

</ConteoPoblacion>

<ConteoPoblacion año="2005">

<conteo variable="Hombres" total="21918"/>

<conteo variable="Mujeres" total="23553"/>

</ConteoPoblacion>

</DatosSocioEconomicos>

</Municipio>

<Municipio>

<Nombre>Rincón de Romos</Nombre>

<Coordenadas LA="22.2166" LO="-102.3166"/>

<DatosSocioEconomicos>

<ConteoPoblacion año="1995">

<conteo variable="Hombres" total="19198"/>

<conteo variable="Mujeres" total="19554"/>

</ConteoPoblacion>

<ConteoPoblacion año="2005">

<conteo variable="Hombres" total="21918"/>

<conteo variable="Mujeres" total="23553"/>

</ConteoPoblacion>

</DatosSocioEconomicos>

</Municipio>

<Municipio>

<Nombre>San José de Gracia</Nombre>

<Coordenadas LA="22.15" LO="-102.4166"/>

<DatosSocioEconomicos>

<ConteoPoblacion año="1995">

<conteo variable="Hombres" total="3492"/>

<conteo variable="Mujeres" total="3678"/>

</ConteoPoblacion>

<ConteoPoblacion año="2005">

<conteo variable="Hombres" total="3600"/>

<conteo variable="Mujeres" total="4031"/>

</ConteoPoblacion>

</DatosSocioEconomicos>

</Municipio>

<Municipio>

<Nombre>Tepezalá</Nombre>

<Coordenadas LA="22.2166" LO="-102.16"/>

<DatosSocioEconomicos>

<ConteoPoblacion año="1995">

<conteo variable="Hombres" total="8100"/>

<conteo variable="Mujeres" total="8075"/>

</ConteoPoblacion>

<ConteoPoblacion año="2005">

<conteo variable="Hombres" total="8318"/>

<conteo variable="Mujeres" total="9054"/>

</ConteoPoblacion>

</DatosSocioEconomicos>

</Municipio>

<Municipio>

<Nombre>El Llano</Nombre>

<Coordenadas LA="21.9166" LO="-101.9666"/>

<DatosSocioEconomicos>

<ConteoPoblacion año="1995">

<conteo variable="Hombres" total="7268"/>

<conteo variable="Mujeres" total="7010"/>

</ConteoPoblacion>

<ConteoPoblacion año="2005">

<conteo variable="Hombres" total="8552"/>

<conteo variable="Mujeres" total="8563"/>

</ConteoPoblacion>

</DatosSocioEconomicos>

</Municipio>

<Municipio>

<Nombre>San Francisco de los Romo</Nombre>

<Coordenadas LA="22" LO="-102"/>

<DatosSocioEconomicos>

<ConteoPoblacion año="1995">

<conteo variable="Hombres" total="8769"/>

<conteo variable="Mujeres" total="9067"/>

</ConteoPoblacion>

<ConteoPoblacion año="2005">

<conteo variable="Hombres" total="14007"/>

<conteo variable="Mujeres" total="14825"/>

</ConteoPoblacion>

</DatosSocioEconomicos>

</Municipio>

</Municipios>

</RegionSocioeconomica>

</EntidadFederativa>

</Region>

</RegionesFederales>

X3.3 XML con datos socioeconómicos del INEGI.\*

$censosAguascalientes = array(

Indice1 => array(

'nombreMunicipio' => ..., /\*\*Nombre del municipio\*/

'noConteosPoblacionales' => ...,

'conteosPoblacionales' => array(

Indice1 => array(

'añoDelCenso' => ..., /\*\*Año del censo\*/

'poblacionHombres' => ..., /\*\*Población de Hombres\*/

'poblacionMujeres' => ..., /\*\*Población de Mujeres\*/

'poblacionTotal' => ..., /\*\*Población Total\*/

)

.

.

.

)

),

.

.

.

)

PX3.9 Meta estructura del árbol final del Ejemplo 75.

De esta manera el programa que resuelve la estructura necesitada de acuerdo a la figura PX3.9 se muestra continuación:

5  Los datos mostrados en el XML X3.3 fueron tomados del registro de conteo poblacional <<[Conteo de Población y Vivienda 1995](http://www.inegi.org.mx/lib/Olap/consulta/general_ver4/MDXQueryDatos.asp?#Regreso&c=11884)>> y <<[Conteo de Población y Vivienda 2005](http://www.inegi.org.mx/lib/Olap/consulta/general_ver4/MDXQueryDatos.asp?#Regreso&c=17352)>> en la página del Instituto Nacional de Estádistica y Geografía INEGI en www.inegi.org.mx.

<?php

/\*\*Carga a la ejecución el script que contiene la clase Projection.\*/

include\_once '../Ejemplo/Proyeccion.php';

/\*\*Se carga el XML en una variable tipo string.\*/

$xmlCensosInegi = file\_get\_contents('ConteosInegi.xml');

/\*\*Inicializa el motor o intérprete de las proyecciones.\*/

$pxmlEngine = new IAPI\_XMLProjection();

/\*\*Definición del descriptor de la proyección\*/

$dpCensosMxAgu = [

'>\_MAP' => [

'nombreMunicipio' => 'Nombre',

'noConteosPoblacionales' => 'count(DatosSocioEconomicos/ConteoPoblacion[@año>1990 and @año<2010])',

'censosPoblacionales' => [

'>\_MAP' => [

'añoDelCenso' => '@año',

'poblacionHombres' => 'conteo[@variable="Hombres"]/@total',

'poblacionMujeres' => 'conteo[@variable="Mujeres"]/@total',

'poblacionTotal' => 'sum(conteo/@total)',

],

'>\_FROOT' => 'DatosSocioEconomicos/ConteoPoblacion'

. '[@año>1990 and @año<2010]'

]

],

'>\_FROOT' => '//RegionesFederales/Region'

. '/EntidadFederativa[@nombre="Aguascalientes"]'

. '/RegionSocioeconomica/Municipios'

. '/Municipio'

];

/\*\*Ejecución de la proyección e impresión en pantalla.\*/

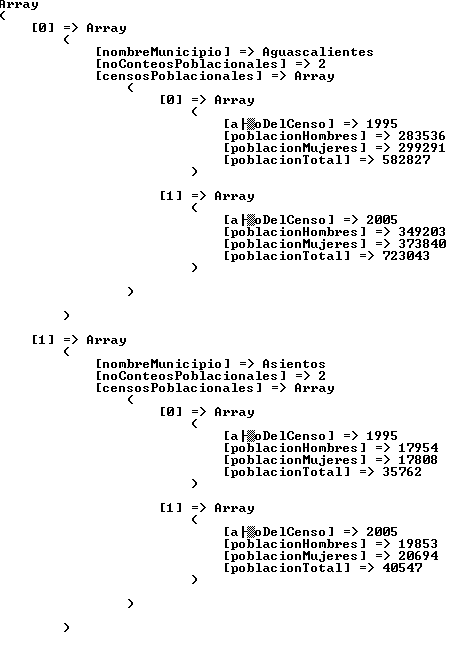
echo print\_r($pxmlEngine->runProjection($dpCensosMxAgu, $xmlCensosInegi), true);

PX3.9 Solución del Ejemplo 7.

Se muestra parte de la salida de este programa en la imagen PE3.6 donde lo importante es notar como la estructura del arreglo de salida se ajusta a lo solicitado en el descriptor de la proyección.

Una operación de proyección es una operación que necesita de dos instrucciones para ser definida, >\_FROOT y >\_MAP, que representan la traza y el mapa de la proyección respectivamente. De esta manera cuando a una metaclave se asocia a un arreglo con la estructura de un descriptor de proyección (Véase la figura PX3.2), se le está indicando al procesador de las proyecciones que parte de los datos requeridos no pueden ser tomados sin ignorar el contexto del nodo rama al que pertenecen, por lo que estos datos necesitan permanecer agrupados tal como en el subárbol en que se encuentran contenidos.

\*  Las funciones *count* y *sum* se encuentran especificadas en el capítulo 4to <<[Number functions](https://www.w3.org/TR/xpath/#section-Number-Functions) >> de la norma de procesamiento <<[XML Path Language Version 1.0](https://www.w3.org/TR/xpath/)>> publicada por el World Wide Web Cosortium en www.w3c.org. . Recomiendo leer la versión traducida por Juan Gómez Duaso <<[Lenguaje de direccionamiento XML](http://www.sidar.org/recur/desdi/traduc/es/xml/xpath.html#section-Function-Calls)>> en www.sidar.org.



PE3.6 Parte del resultado de la ejecución del programa PX3.9.

No hay límites para definir un DP dentro de otro DP, pues el motor de las proyecciones procesa de forma recursiva las definiciones internas de subproyecciones. Téngase en cuenta que cuando la TP (>\_FROOT) de un subdescriptor de proyección no coincide con algún nodo, la metaclave a la que se asocia se inicializa como un arreglo vacío durante la ejecución de la proyección.

**3.6 Recomendaciones durante la codificación**

Con la finalidad de mejorar la calidad de codificación y mantener un código limpio, legible y en consecuencia entendible, se hace una serie de RECOMENDACIONES respecto a la forma en que se codifican los descriptores de proyección en base a los documentos <<[Guia de entorno de usuario para nombres](http://php.net/manual/es/userlandnaming.rules.php)>> de PHP, <<[Coding style](https://framework.zend.com/manual/1.11/en/coding-standard.coding-style.html)>> del Estandar de Codificación de Zend Framework y <<[Convencion de denominaciones de Java](https://www.ibm.com/developerworks/ssa/library/ws-tip-namingconv/index.html)>> de IBM.

Se recomienda leer cada uno de los documentos mencionados anteriormente, pero en general se resaltan las siguientes reglas de codificación:

1. Se RECOMIENDA que una línea de código no supere los 80 caracteres, y en caso de ser requerido como máximo una línea DEBE contener 120 caracteres.
2. Se recomienda que un nombre de variable no supere los 15 caracteres.
3. Para nombrar las variables utilícese la variante de nombramiento de “Letras capitales intermedias” (en inglés *camelCase*).

**3.6.1 Variables**

Además de utilizar las reglas anteriormente mencionadas, se propone utilizar la siguiente lista de prefijos para nombrar las variables que participan en el programa que codifica la proyección.

|  |  |  |
| --- | --- | --- |
| **Prefijo** | **Significado** | **Ejemplos** |
| ***dp*** | Utilícese con variables que almacenen estructuras equivalentes a un descriptor de proyección. | $dpMenu, $dpPlataforma, etc. |
| ***tp*** | Utilícese con variables que contengan expresiones XPath destinadas a la traza de la proyección. | $tpPrincipal, $tpHoteles, etc. |
| ***map*** | Utilícese con variables que contengan estructuras equivalentes a un mapa de proyección. | $mapCoordenadas, $mapSeleccion, etc. |
| ***xml*** | Utilícese con variables que contengan el documento XML a procesar por el motor de las proyecciones. | $xmlUbicaciones, $xmlCartografia, $xmlResponse, etc. |
| ***af*** | Utilícese para las variables que contenga el resultado de la proyección. | $afResultado, $afConteos, etc. |
| ***xp*** | Utilicese para variables que contengan una expresión XPath y no se ajuste con alguno de los criterios anteriores. | $xpPrincipal, $xpSeleccion, $xpCiudades,etc |

**3.6.2 Arreglos**

Respecto a la definición de los arreglos que formarán parte del descriptor de la proyección, estos se DEBEN *codificar con la sintaxis de corchetes* de PHP, pues su definición con los paréntesis y la palabra reservada *array* hace de su codificación una carga sintáctica más compleja y difícil de seguir. Resalto la siguiente traducción realizada por Benjamín González de respecto del capítulo <<[arrays](https://framework.zend.com/manual/1.11/en/coding-standard.coding-style.html)>> de la norma de Codificación de Zend para tenerse en cuenta al momento de escribir un DP: “Alternativamente, el elemento inicial del array puede comenzar en la siguiente línea. Si es así, debe ser alineado en un nivel de sangría superior a la línea que contiene la declaración del array, y todas las sucesivas líneas deben tener [el mismo nivel de sangría], el paréntesis de cierre debe ser en una nueva línea al mismo nivel [de sangría] que la línea que contiene la declaración del array: Para mejor legibilidad, los diversos operadores de asignación "=>" deben ser rellenados con espacios en blanco hasta que se alineen.” 6. Un ejemplo de lo anterior se presenta a continuación:

$censosAguascalientes = [

'elementoUno' => 'elementoUno',

'elementoDos' => 'elementoDos',

'elementoTres' => 'elementoTres',

];

Tomando en cuenta las reglas anteriores, se muestra abajo un ejemplo de codificación del mismo descriptor de la proyección utilizado en el programa PX3.9 bajo esta norma:

$tpConteos = 'DatosSocioEconomicos/ConteoPoblacion[@año>1990 and @año<2010]';

$dpConteos = [

'>\_MAP' => [

'añoDelCenso' => '@año',

'poblacionHombres' => 'conteo[@variable="Hombres"]/@total',

'poblacionMujeres' => 'conteo[@variable="Mujeres"]/@total',

'poblacionTotal' => 'sum(conteo/@total)',

],

'>\_FROOT' => $tpConteos

];

$dpCensosMxAgu = [

'>\_MAP' => [

'nombreMunicipio' => 'Nombre',

'noConteosPoblacionales' => "count({$tpConteos })",

'censosPoblacionales' => $dpConteos

],

'>\_FROOT' => '//RegionesFederales/Region'

. '/EntidadFederativa[@nombre="Aguascalientes"]'

. '/RegionSocioeconomica/Municipios'

. '/Municipio'

];

PX3.10 Mejora en la codificación del descriptor de la proyección del programa PX3.9.

**Índice de Imágenes**

Índice de gráficos de estructuras de árbol

A2.1- Árbol asociado al XML de productos X2.1, pág. 6.

A2.2- Estructura de árbol final, pág. 7.

A2.3- Proyección de datos sobre de la estructura de árbol final, pág. 7.

A2.4- Estructura de árbol asociada al documento XML X2.2, pág. 11.

A2.5- Árbol final, pág. 13.

Indice de graficos de relaciones entre árboles

AR2.1- Relación estructural, pág. 8.

AR2.2- Relación de los datos, pág. 8.

AR2.3- Relación de proyección, pág. 14.

Índice de gráficos de consultas

AQ2.1- Selección de frutas y verduras, pág. 9.

AQ2.2- Productos en frutas y verduras con precio menor a 20 pesos, pág. 10

**Índice de códigos fuente**

Índice de programas

PS2.1- Pseudocódigo del descriptor de la proyección, pág. 14.

PX3.1- Instancia de la clase IAPI\_XMLProjection, pág. 17.

PX3.2- Estructura general del descriptor de la proyección, pág. 18.

PX3.3- Metaestructura del árbol final del Ejemplo 3, pág. 21.

PX3.4- Solución del Ejemplo 3, pág. 22.

PX3.5- Solución del Ejemplo 4, pág. 23.

PX3.6- Solución del Ejemplo 5, pág. 25.

PX3.7- Utilización de >\_CSET con un símbolo separador, pág. 26.

PX3.8- Solución del Ejemplo 6, pág. 27.

PX3.9- Metaestructura del árbol final del Ejemplo 7, pág. 32.

PX3.10- Solución del Ejemplo 7, pág. 33.

PX3.11- Mejora en la codificación del DP del programa PX3.9, pág. 36.

Pruebas de escritorio

PE3.1- Resultado de la ejecución del programa PX3.4, pág. 22.

PE3.2- Resultado de la ejecución del programa PX3.5, pág. 24.

PE3.3- Resultado de la ejecución del programa PX3.6, pág. 25.

PE3.4- Resultado de la ejecución del programa PX3.7, pág. 26.

PE3.5- Resultado de la ejecución del programa PX3.8, pág. 28.

PE3.6- Parte del resultado de la ejecución del programa PX3.9, pág. 34.

**Fuentes bibliográficas**

**Referencias normativas**

[RFC2119] Bradner, S., “[Key words for use in RFCs to Indicate Requirement](https://tools.ietf.org/html/rfc2119) [Levels”](https://tools.ietf.org/html/rfc2119), BPC 14, RFC 2119, Marzo 1997.

[1] World Wide Web Consortium (W3C), “[XML Path Language (XPath) Version](https://www.w3.org/TR/xpath/) [1.0”,](https://www.w3.org/TR/xpath/) Noviembre 1999.

[2] World Wide Web Consortium (W3C). “[XML Path Language (XPath) Versión](https://www.w3.org/TR/xpath-31/) [3.1”,](https://www.w3.org/TR/xpath-31/) Marzo 2017.

[3] World Wide Web Consortium (W3C). “[Extensible Markup Language (XML)](https://www.w3.org/TR/1998/REC-xml-19980210) [1.0”,](https://www.w3.org/TR/1998/REC-xml-19980210) Febrero 1998.

[4] World Wide Web Consortium (W3C). “[XSL Transformations (XSLT) Version](https://www.w3.org/TR/xslt) [1.0”,](https://www.w3.org/TR/xslt) Noviembre 1999.

[5] Zend Framework, “[Zend Framework Coding Standard for PHP”,](https://framework.zend.com/manual/1.11/en/coding-standard.html) s.d.

[6] World Wide Web Consortium (W3C). “[Document Object Model Core”,](https://www.w3.org/TR/2004/REC-DOM-Level-3-Core-20040407/core.html#ID-1312295772) Abril 2004.

**Referencias informativas**

[8] Scott W. Ambler, International Bussiness Machines (IBM), “[Convencion](https://www.ibm.com/developerworks/ssa/library/ws-tip-namingconv/index.html) [de denominaciones de Java”,](https://www.ibm.com/developerworks/ssa/library/ws-tip-namingconv/index.html) Diciembre 2012.

[9] Real Academia Española “[Normas de escritura de prefijos”,](http://www.rae.es/consultas/normas-de-escritura-de-los-prefijos-exmarido-ex-primer-ministro) Consultado [en junio del 2017 en http://www.rae.es.](http://www.rae.es)

[10] Oracle Communications, “[Using XPath Functions”,](https://docs.oracle.com/cd/E35413_01/doc.722/e35419/dev_xpath_functions.htm) Consultado en junio [del 2017 em https://docs.oracle.com.](https://docs.oracle.com/)

[11] PHP Working Group, “[The DOMXpath class”,](http://php.net/manual/en/class.domxpath.php) Consultado en junio del 2017 en <http://www.php.net/manual/en>.

[12] SIDAR, “[Lenguaje de Caminos XML (XPath) Versión 1.0”,](http://www.sidar.org/recur/desdi/traduc/es/xml/xpath.html#path-abbrev) Traducción del documento “[XML Path Language (XPath) Version 1.0”](https://www.w3.org/TR/xpath/) del W3C realizada por Juan Gómez Duaso el 19 de octubre del 2001.

**Datos de contacto del autor**

Ricardo Bermudez Bermudez

Teléfono de casa: 01 222 2206336

Teléfono celular: 044 5548879549

Teléfono interno: 01 55 52650730 Ext. 3060

Correo electrónico personal[: ricardob.sistemas@gmail.com.mx](mailto:ricardob.sistemas@gmail.com.mx)

[Correo electrónico interno: rbermudez@mig.com.mx](mailto:rbermudez@mig.com.mx)