VNIVERSIDAD NACIONAL AVPNOMA DE MEXICO

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

POSGRADO EN CIENCIA E INGENIERÍA DE LA COMPUTACIÓN

DISEÑO DE UNA BASE DE DATOS XML, PARA EL ALMACENAMIENTO Y RECUPERACIÓN DE REPORTES DE INVESTIGACIÓN

T E S I S

QUE PARA OBTENER EL GRADO DE:

MAESTRO EN INGENIERÍA (COMPUTACIÓN)

P R E S E N T A:

MIGUEL ANGEL MEJÍA RAMÍREZ

DIRECTORA DE LA TESIS: DRA. AMPARO LÓPEZ GAONA

MÉXICO, D.F. **2009.**

Agradecimientos

A Dios, por concederme la fuerza y el entusiasmo para realización de las metas fijadas.

A mis queridos Padres, por el cariño y apoyo que siempre obtuve de su parte.

A mi hermana Alejandra, por las palabras de aliento y el interés mostrado.

A mis hermanas Xochitl y Susana y mi hermano Mauricio, por el apoyo que me dieron.

A la Dra. Amparo y los sinodales, por el tiempo y los conocimientos aportados.

INDICE

Introducción

Descripción del problema

Actualmente la publicación de información no sólo se realiza en papel, en estos días existen innumerables documentos en los medios electrónicos, los cuales cada vez tienen mayor relevancia tanto por la forma en que dicha información es presentada, como por su contenido; y la facilidad con que ésta pueda ser localizada y compartida.

Para la creación y publicación de documentos electrónicos, se utilizan distintas herramientas y lenguajes informáticos, entre ellos destaca el HyperText Markup Language, más conocido como lenguaje **HTML**, éste ayuda a la creación de páginas web y documentos electrónicos con relativa facilidad, al mismo tiempo proporciona elementos que permiten dar una buena presentación a la información para que ésta sea vista a través de cualquier navegador o de programas específicos.

En Internet, existe una complicada colección de información, la cual se encuentra disponible pero que desgraciadamente, no facilita la recuperación llamémosla "inteligente" de un conjunto de documentos que contengan las datos requeridos. Estos datos se pueden presentar en muy variadas formas, las cuales van desde un simple texto a una imagen específica de un tema en particular pasando hasta por un idioma preciso.

Aunque HTML es un buen lenguaje, tiene sus limitaciones y una de ellas es que no cuenta con herramientas necesarias para realizar un análisis del contenido de los documentos presentados, motivo por el cual es necesario apoyarse en otros lenguajes que ayuden en la solución para el problema de la publicar y compartir la información por medio de archivos electrónicos.

HTML puede trabajar de manera conjunta con el eXtensible Markup Language, o lenguaje **XML**, dicho lenguaje sí proporciona los elementos o marcas para realizar un análisis del

contenido de los documentos generados, así como la generación de etiquetas personalizadas, de acuerdo a las necesidades particulares en cada tema.

Hoy en día para realizar un análisis de contendido, se requiere de metadatos, estos se podrían definir como datos que proporcionan información o realizan una descripción sobre otros datos. Para el manejo de estos metadatos, se cuenta con el ya mencionado lenguaje **XML**. Y sus múltiples elementos, entre los cuales destacan las DTD y los Esquemas.

En el caso de los reportes de investigación, éstos requieren de un formato específico para su creación y difusión, en el cual se puede incluir información del autor o autores, bibliografía utilizada, palabras clave, párrafos específicos, etc. Este formato permitiría la estandarización al momento de generar los reportes. Asimismo se puede compartir la información de una manera segura y con la confianza de que aquel que la reciba podrá entenderla y explotarla, lo que aseguraría el intercambio y almacenamiento de los documentos en una base de datos, con las consecuentes ventajas que esto brinda (fácil recuperación, modificaciones de manera sencilla y búsquedas avanzadas).

Como todo en la vida tiene un precio, el generar dicho documento requiere de conocer **XML,** y disponer de una herramienta que facilite la captura de la información y la transforme en un documento correcto de XML.

Afortunadamente, la mayoría de personas que trabajan en la creación de estos documentos, posee ya ciertos conocimientos informáticos, y utilizan algunos programas, llámense procesadores de palabras o editores de texto, uno de los programas más populares es Word, así que es posible aprovechar los conocimientos que mucha gente posee sobre este programa, para facilitar la captura de información y con ello la generación de reportes con un formato en específico.

Actualmente existen algunas normas que se han adoptado, con la finalidad de facilitar el manejo de la información, esto es, que al implementarse en una base de datos o conjunto de

reportes, estos pueden organizarse de una mejor manera y con ello facilitar el acceso y el análisis de su contenido.

Definición del Objetivo y Alcance

El objetivo principal de este trabajo es generar una base de datos que facilite la manipulación de documentos XML, y la creación de herramientas que permitan la elaboración de dichos documentos, para ello es necesario realizar un análisis de los Esquemas XML existentes, en cuanto a publicación de documentos electrónicos se refiere, para en su caso definir o proponer uno, que se adapte a las necesidades de los reportes de investigación realizados en las dependencias universitarias.

De igual manera se realizará un análisis sobre sistemas manejadores de bases de datos (SMBD) para seleccionar el que proporcione las mejores herramientas, para realizar el diseño de la Base de Datos para documentos XML, que permita el almacenamiento y recuperación de reportes de investigación definidos bajo el esquema propuesto.

Asimismo, se desarrollarán las herramientas necesarias para la creación de documentos XML, mediante el uso de Microsoft Word y Adobe Acrobat, aprovechando el conocimiento que poseen los usuarios sobre estos programas, de tal manera que su elaboración sea de manera fácil y sencilla para cualquier persona, sin que ello signifique que tenga que aprender el lenguaje de marcado XML y el diseño de esquemas.

Descripción de capítulos

El presente trabajo se divide en cuatro capítulos, en el primero se presenta el lenguaje XML, como una opción para el marcado de documentos científicos, y el uso de las estructuras llamadas esquemas, así como un análisis de diferentes esquemas existentes, y como resultado de dicho análisis la propuesta de un esquema para el diseño de los reportes que se almacenarán en una base de datos.

En el capítulo 2 se identifican los conceptos de “bases de datos nativas y habilitadas” para el uso de documentos XML, reconociendo las ventajas y desventajas de uno y otro modelo, asimismo se describen los sistemas manejadores de bases de datos eXist y Postgres.

El capítulo 3 trata sobre la metodología utilizada para desarrollar la solución propuesta, describiendo los requerimientos, las especificaciones y el análisis empleado.

En el capítulo 4 se presenta la construcción de la solución propuesta, indicando los programas que se utilizaron para su elaboración.

Finalmente se mencionan las conclusiones y los resultados obtenidos durante el desarrollo de la presente tesis y la aplicación de las herramientas elaboradas, también se identifican las áreas de futuro desarrollo.

CAPÍTULO 1

XML para marcado de documentos científicos

1.1 El lenguaje XML

Actualmente existen más de 800 millones de páginas Web basadas en el lenguaje HTML, que si bien es un lenguaje popular y exitoso, es utilizado regularmente para facilitar la presentación de documentos, pero no fue diseñado para definir el contenido de los documentos, por lo que la información que éstos almacenan no se puede explotar de manera ideal. Para superar las limitaciones de definición de contenido es de mucha utilidad el lenguaje XML (eXtensible Markup Language), éste es un lenguaje de marcado, que fue diseñado por el W3C (Consorcio World Wide Web).

De manera muy rápida XML ha cobrado gran importancia y actualmente existen diversas áreas donde es útil, por ejemplo:

- Mantenimiento de sitios Web grandes
- Intercambio de información entre organizaciones
- Descargas y cargas de bases de datos
- Aplicaciones de comercio electrónico
- Aplicaciones científicas con nuevos lenguajes de marcado para fórmulas matemáticas y químicas
- Libros electrónicos
- Dispositivos de mano y teléfonos inteligentes

El marcado del documento en XML, se entiende básicamente como el insertar información por medio de etiquetas (metadatos) que ayudan a definir una semántica específica para dichos documentos.

Las etiquetas son cada vez más especializadas. Por ejemplo los investigadores de un instituto "A" y los de un instituto "B" necesitan etiquetas especiales para sus fórmulas, pero estas etiquetas no son las mismas para los dos. Por lo cual es necesario definir cada vez un mayor número de etiquetas de acuerdo a una necesidad en específico.

XML a diferencia de HTML, establece dos cambios fundamentales que mejoran substancialmente su funcionamiento, estos son que "no predefine etiquetas" y "es estricto".

Al no haber etiquetas predefinidas, el autor tiene la libertad de crear las etiquetas que le sean necesarias. Por ejemplo, si se necesita una etiqueta para definir a un coautor, bastaría con lo siguiente:

```
<coautor>Angel Mejía</coautor>
```

Al utilizar HTML se tiene una sintaxis demasiado bondadosa, lo que origina que en ocasiones se tengan muchos errores dentro de las páginas Web, XML "es estricto" en su sintaxis, lo que da como resultado una disminución considerable de los errores y con ello exploradores más pequeños, rápidos y ligeros.

Las aplicaciones de XML se clasifican principalmente en dos tipos:

- Aplicaciones de documento que manejan información que va dirigida principalmente para usuarios que realizarán consultas de manera manual.
- Aplicaciones de datos que manejan información que va dirigida principalmente para aplicaciones de software que explotarán la información de manera automática.

La diferencia principal entre estos dos tipos de aplicaciones es cualitativa. Es el mismo estándar XML, se usan las mismas herramientas, pero sirve a distintos propósitos. [1]

En las aplicaciones a documentos, la principal ventaja de XML es que se concentra en la estructura del documento y esto hace que sea independiente del formato de entrega (ya sea

HTML, XHTML, PostScript, PDF, etc.). De esta manera, el usuario puede acceder a la información de la manera en que mejor le convenga.

Por lo tanto es posible editar y mantener documentos XML y publicarlos **automáticamente** en internet o utilizarlos en alguna aplicación en específico (Microsoft Word o Adobe Acrobat).

En las aplicaciones de datos, XML facilita el acceso a los documentos mediante varias herramientas de software, mediante procesos automáticos, y con ello también se puede compartir información con sistemas de bases de datos.

Con XML, se ofrece a los datos una clase de distribución. Esto genera el concepto de "la aplicación como documento" donde, finalmente no hay diferencia entre documentos y aplicaciones. [1]

Un documento XML puede servir para ingresar información a una base de datos (colocando cada elemento del documento como un campo de datos), de igual forma una base de datos puede servir para extraer información y con ella formar un documento XML (colocando la información de cada campo de datos como un elemento del documento XML).

En este contexto, XML se puede utilizar para intercambiar información entre organizaciones. La información contenida en la Web que es generada con XML se puede ver como una enorme base datos que puede ser aprovechada por diversas aplicaciones.

Al publicar información en la Web utilizando XML, se pueden utilizar herramientas para visitar de manera automática la página en cuestión, extraer información específica y procesarla de acuerdo a una tarea delimitada con anterioridad.

Cuando se utiliza XML, se pueden aprovechar varios estándares desarrollados por la W3C, estos se conocen con el nombre de "estándares acompañantes de XML", algunos de ellos son los siguientes:

- XML Namespace.- asocia un dueño con cada uno de los elementos, lo cual previene los conflictos de nombres.
- DOM y SAX.- son API´s para acceder a documentos XML, las cuales permiten a las aplicaciones leer documentos XML sin tener que preocuparse por la sintaxis. DOM (Modelo de Objetos de Documento) es la representación estándar de un documento en memoria, en la cual se representa un documento como un árbol de nodos, SAX (API Simple para XML) en lugar de crear un árbol de un documento XML, va leyendo el archivo y ejecuta unos oyentes registrados que perciben cuándo ocurren ciertos eventos del análisis.
- Hojas de estilo.- especifican cómo deben presentarse en pantalla, en papel o en un editor los documentos XML. Es soportado por dos lenguajes XSL (Lenguaje de Hojas de Estilo Extensible) y CSS (Hojas de Estilo en Cascada).

La estructura general de un documento XML está dividida en encabezado y cuerpo, similar a la siguiente:

```
<?xml versión=”1.0”>
        <elementoraiz atributos (xmlns,…)>
                Resto del documento …
                ……
        </elementoraiz>
```

Estructura de un documento XML [2]

En el encabezado, se tiene la línea **<?xml versión=”1.0”>** en donde se indica que este documento es un documento XML, que cumple con la versión 1.0 del estándar establecido por la W3C.

El cuerpo del documento está formado por un conjunto de elementos que serán definidos de acuerdo a las necesidades del autor, y se encuentran dentro de un **<elementoraiz>**.

Para poder explotar de manera adecuada los documentos XML, es recomendable que éstos se encuentren bien formados y sean válidos.

Se dice que un documento bien formado es aquel que obedece un orden jerárquico para todos y cada uno de sus elementos.

Para que un documento sea válido, primero tendrá que estar bien formado y después se deberá verificar que cada uno de sus elementos, obedecen una cierta secuencia lógica.

Para la definición de la estructura de documentos XML y las etiquetas a emplear en su elaboración, se pueden utilizar archivos llamados DTD (Definición de Tipos de Documentos) y Schema (Esquema XML).

Un archivo DTD permite definir y asignar nombre a los elementos que se pueden utilizar en el documento, el orden en el cual podrán aparecer dichos elementos y los atributos de los elementos que podrán utilizarse.

Los elementos de la DTD, indican cuales serán las etiquetas que se podrán usar dentro del documento XML y en qué orden. El formato general de la definición de un elemento es el siguiente:

```
<!ELEMENT nombre contenido>
```

En donde **ELEMENT** es una parte obligatoria que permitirá establecer el nombre del elemento; **nombre** indica el nombre del elemento, y serán las etiquetas que podrán ser utilizadas en el documento; **contenido** define el tipo de datos que van a contener las etiquetas.

```
<!ELEMENT Documento ( #PCDATA )>
<!ELEMENT TEI2 ( teiHeader, text ) >
<!ELEMENT back ( div1* ) >
```

Los atributos también son definidos dentro de la DTD, y el formato general es el siguiente:

```
<!ATTLIST elemento atributo tipo utilización>
```

En donde **ATTLIST** es una parte obligatoria que permitirá establecer el atributo, en **elemento** se establece el elemento al cual estará asociado el atributo, en **atributo** se indica el nombre del atributo, en **tipo** se indica el tipo de datos que puede contener el atributo que se está definiendo (ejemplo CDATA y PCDATA), en **utilización** se indica si el atributo deberá aparecer obligatoriamente (#REQUIRED) o si es opcional (#IMPLIED).

```
<!ATTLIST div1 n NMTOKEN #IMPLIED >
<!ATTLIST div2 name CDATA #IMPLIED >
<!ATTLIST head rend NMTOKEN #IMPLIED >
```

Un archivo Schema, permite escribir esquemas detallados para los documentos XML, utilizando una sintaxis estándar XML. Es una alternativa que permite la creación de nuevos tipos de datos y con ello limitar los errores que pudieran presentarse al momento de generar un documento XML.

La estructura general de un Schema es la siguiente:

```
<?xml versión=”1.0” standalone=”yes” ?>
    <schema xmlns=ubicaciondelschema versión=”1.0”>
        Cuerpo del documento XML Schema ……
    </schema>
```

En la parte del cuerpo del programa se indican los elementos, atributos y tipos de datos.

Los elementos del Schema, al igual que en los DTD, indican cuales serán las etiquetas que se podrán usar dentro del documento XML, el formato general de la definición de un elemento es la siguiente:

```
<element nombre=”valornombre” tipo=”valortipo” />
```

En donde **element** es una parte obligatoria que permitirá establecer el nombre del elemento; **nombre** indica el nombre del elemento mediante **valornombre**, y serán las etiquetas que podrán ser utilizadas en el documento; **tipo** define el tipo de datos que van a contener las etiquetas mediante **valortipo**.

```
<xs:element ref="teiHeader"/>
<xs:element ref="div1" minOccurs="0" maxOccurs="unbounded"/>
<xs:element name="fileDesc">
```

Los atributos también son definidos de una manera similar a la manejada en los archivos DTD, y el formato general es el siguiente:

```
<attribute name=”nombreatri” type=”tipodevalor” />
```

En donde **attibute** es una parte obligatoria que permitirá establecer el atributo, en **name** se establece el nombre del atributo mediante **nombreatri**, en **type** se indica el tipo de datos que puede contener el atributo que se esta definiendo mediante **tipodevalor** (ejemplo string, integer, decimal).

```
<xs:attribute name="corresp" type="xs:IDREFS"/>
<xs:attribute name="n" type="xs:anySimpleType"/>
<xs:attribute name="rend" type="xs:NMTOKEN" use="required"/>
```

Actualmente existen varias herramientas que facilitan la generación de documentos XML, las cuales van realizando una validación de acuerdo con las definiciones del lenguaje, así como el cierre automático de etiquetas y la indentación necesaria para una fácil lectura.

Asimismo, y aprovechando las ventajas que exhibe XML, ya se cuenta con colecciones de marcas diseñadas de manera específica para ciertas aplicaciones como son el uso de

fórmulas químicas (CML), equipos inalámbricos (WML), fórmulas matemáticas (MathML), etc.

Como XML permite el uso de etiquetas personalizadas y diseñadas de manera particular, se podría presentar el hecho de que los nombres de dos o más etiquetas se repitieran, aunque éstas pudieran tener un significado totalmente distinto.

Para solucionar este conflicto existen los namespaces (espacios de nombres), que permiten agrupar las etiquetas, en subconjuntos o particiones, estableciendo un nombre específico para cada uno de ellos y dentro de éstos el nombre de todas y cada una de las etiquetas que podrán utilizarse (mediante un DTD o Schema), de tal manera que se puede definir una o más etiquetas con el mismo nombre pero en diferente subconjunto.

Para hacer uso de ellas, se deberá indicar primeramente el espacio de nombres y después el nombre de la etiqueta, separados por dos puntos ":", esto es:

```
<espaciodenombres : etiqueta>
```

El espacio de nombres no es esencial, ni de uso obligatorio, pero si resulta muy útil cuando se usan etiquetas que deben ser procesadas de forma distinta o bien provenientes de diferentes sitios.

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

Actualmente existen muchas aplicaciones disponibles que permiten manipular documentos XML, como son Exploradores (Internet Explorer, Mozilla), Editores (Adobe Framemarker, XML Pro) y Analizadores (XML for Java).

1.2 Lenguajes para descripción de metadatos basados en XML

Los esquemas para la definición de documentos XML, pueden ser definidos de acuerdo a las necesidades especificas de una compañía o usuario, sin embargo, existen algunos esquemas que procuran definir un formato estándar, descriptivo y a la vez sencillo, que facilite la interoperabilidad dentro de Internet, para que éste pueda ser empleado en el diseño de documentos XML, y de esta manera puedan ser intercambiados y examinados por un número mayor de usuarios.

A continuación se toman en consideración a algunos de ellos, con la finalidad de observar las ventajas que pudieran brindar para la generación de reportes de investigación.

Una aplicación que se encuentra promovida por la W3C, es el esquema RDF (Resource Description Framework), el cual cuenta con un modelo de datos, una sintaxis y esquema predeterminado, y tiene una gran utilización en el ámbito de las bibliotecas ya que proporciona un marco de trabajo que facilita el intercambio de información de cualquier tipo, incluyendo recursos XML y no-XML.

Derivado de lo anterior, el RDF se utiliza en las bibliotecas para la recuperación de información mediante motores de búsqueda, el intercambio de información en forma de conocimiento generado mediante software inteligente, en la representación de documentos lógicos, en la elaboración de catálogos para descripción de contenidos, para la definición de los derechos de autor de las páginas Web, etc.

El RDF define un modelo de datos simple para diseñar las relaciones entre recursos, así las propiedades RDF pueden concebirse como atributos de un recurso y de esta manera se relacionan como atributo-valor. Este modelo RDF asemeja un diagrama entidad-relación, sin embargo, no suministra todos los mecanismos para declaración de propiedades, ni para las relaciones entre éstas y otros recursos.

Los atributos y su tipo de datos se definen, en el contexto del RDF como un esquema, el cual define las propiedades de un recurso (autor o autores, tema, titulo, etc.) y de igual manera define los tipos de recurso (tesis, proyectos de investigación, libros, etc.).

En resumen, el esquema RDF suministra un sistema tipo que precisa recursos y propiedades que se usan en la definición de esquemas de aplicación para un uso determinado.

El lenguaje de especificación del esquema RDF permite hacer una representación del conocimiento empezando por una lógica de predicados y puede llegar hasta una red semántica, de igual manera se puede utilizar en la descripción de esquemas para bases de datos y otros modelos de datos.

```
<?xml version="1.0" encoding="utf-8"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:dc="http://purl.org/dc/elements/1.1/">
  <rdf:Description rdf:about="http://www.w3.org/TR/rdf-syntax-grammar">
    <dc:title>RDF/XML Syntax Specification (Revised)</dc:title>
    <dc:title xml:lang="en">RDF/XML Syntax Specification (Revised)</dc:title>
    <dc:title xml:lang="en-US">RDF/XML Syntax Specification (Revised)</dc:title>
  </rdf:Description>

  <rdf:Description rdf:about="http://example.org/buecher/baum" xml:lang="de">
    <dc:title>Der Baum</dc:title>
    <dc:description>Das Buch ist außergewöhnlich</dc:description>
    <dc:title xml:lang="en">The Tree</dc:title>
  </rdf:Description>
</rdf:RDF>
```

Ejemplo de esquema RDF

Principalmente el esquema RDF se ha utilizado para la recopilación de información de Bibliotecas digitales y la definición de los metadatos que puedan definirlos. Mediante RDF se han definido una serie de vocabularios que permiten un mejor procesamiento, estos vocabularios pueden utilizarse de manera independiente.

Los esquemas RDF proporcionan información sobre la manera en que se interpretarán el grupo de sentencias determinadas en un modelo específico, eso los diferencia de las DTD y Esquemas XML convencionales, que sólo limitan la estructura de un documento XML.

Al igual que con XML, en RDF también se puede utilizar esquemas (schemas), XML es un modelo para etiquetar datos con una interpretación sintáctica, mientras que RDF sirve para etiquetar metadatos y la interpretación de este lenguaje es semántica.

Otro esquema muy conocido es el Dublin Core (DC), el cual trata de facilitar la indización para preparar la interoperabilidad semántica en la web, principalmente se tienen tres grupos, uno para definir el contenido, otro para la propiedad intelectual y uno más para la temporalidad, formato del documento e identificación.

Todo esto mediante quince elementos básicos, los cuales no son obligatorios y en ocasiones pueden ser repetidos.

Contenido	Propiedad intelectual	Temporalidad, formato e identificación
1. Title	8. creator	12. data
2. subject	9. publisher	13. type
3. description	10. contributor	14. format
4. source	11. rights	15. identifier
5. language		
6. relation		
7. coverage		

Elementos básicos de Dublin Core (DC)

Es importante mencionar que este modelo de esquema se utiliza de manera conjunta con el esquema RDF antes mencionado.

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:dc= "http://purl.org/dc/elements/1.1/">
 <rdf:Description rdf:about="http://dublincore.org/documents/2003/06/02/dces/">
   <dc:title>Dublin Core Metadata Element Set, Version 1.1: Reference Description</dc:title>
   <dc:description>The reference description, version 1.1 of the Dublin Core Metadata
Element Set.</dc:description>
   <dc:date>2004-12-20</dc:date>
   <dc:format>text/html</dc:format>
   <dc:language>en</dc:language>
   <dc:publisher>Dublin Core Metadata Initiative</dc:publisher>
 </rdf:Description>
</rdf:RDF>
```

Ejemplo de esquema Dublin Core

Otro esquemas analizado es el TEI (Text Encoding Initiative) el cual surge como un proyecto para el área de humanidades, pero después se utilizó para la codificación de textos principalmente literarios, al igual que RDF y DC, su objetivo es estandarizar el formato de marcado para el intercambio de datos y gracias a su estructura, sirve para todo tipo de datos.

Uno de los principales componentes del TEI es la cabecera TEIH, la cual posibilita la definición bibliográfica pormenorizada de cada documento desde su creación.

El TEI, acoge una estructura especial para distintas materias, así como diversas iniciativas, entre ellas la CIMI (Computer Interchange of Museum Information) que se utiliza para la información de museos digitales, o la FGDC (Federal Geographic Data Committee) utilizada para la información digital geoespacial, también se encuentra la EAD (Encoded Archival Description) para la información archivística.

Algunas de las normas de descripción formal, como Harvard, ISO y Chicago, han incorporado elementos para ser utilizados en páginas web, documentos electrónicos, libros y mensajes de correo electrónicos, etc. De igual manera otras instituciones llevan a cabo contribuciones creando nuevos estándares o ampliando los existentes, con la finalidad de adaptar lo mejor posible los estándares clásicos de acuerdo a los requerimientos existentes.

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:dc= "http://purl.org/dc/elements/1.1/">
 <rdf:Description rdf:about="http://dublincore.org/documents/2003/06/02/dces/">
   <dc:title>Dublin Core Metadata Element Set, Version 1.1: Reference Description</dc:title>
   <dc:description>The reference description, version 1.1 of the Dublin Core Metadata
Element Set.</dc:description>
   <dc:date>2004-12-20</dc:date>
   <dc:format>text/html</dc:format>
   <dc:language>en</dc:language>
   <dc:publisher>Dublin Core Metadata Initiative</dc:publisher>
 </rdf:Description>
</rdf:RDF>
```

Ejemplo de esquema TEI

La Norma ISO 690-2 (Information and documentation - Bibliographic references) establece los elementos para el manejo de documentos del tipo de monografías electrónicas y bases de datos principalmente, e intenta almacenar las fechas de consulta y creación de los documentos, así como, las consultas que se han realizado.

Definición de bibliografías con la Norma ISO 690-2 con los siguientes datos:
1. Autor (es) del capítulo.
2. Título del capítulo.
3. Lugar de publicación.
4. Editorial.
5. Año de publicación.
6. Paginación.

Ejemplo:

GOLDFARB, Charles & PRESCOD, Paul. Manual de XML, México, Prentice Hall, 2005. pp. 100-130

Ejemplo de publicación con Norma ISO 690-2

La APA (American Psychological Association) también ha realizado aportaciones al marcado de documentos para hacer referencia a los textos que publica, ya sea en sitios web, artículos, reportes de investigación o un texto específico dentro de un documento.

Libros

Autor, A. A. (año). Título de la obra. Lugar de publicación: Editor o casa publicadora.

Ejemplo:

Benoit Marchal, (2004). XML con ejemplos, México: Prentice Hall.

Ejemplo de publicación con APA

El método Harvard también es muy utilizado por algunos investigadores, y principalmente contempla el título, autor, edición, lugar de su publicación, disponibilidad, fechas de creación y de consulta, de igual manera, este modelo comprende las categorías de CD, direcciones URL, correo electrónico, etc.

Se puede componer de los siguientes elementos:

1. Los apellidos del autor en minúsculas, excepto la inicial.
2. El año de publicación entre paréntesis.
3. Los títulos del documento pueden ir o subrayadas o en cursiva.
4. El lugar de edición y la editorial,
5. p. (1 página) pp. (más de una página)

Ejemplo:

Young, Michael,(2001). Aprenda XML ya, México, Mc Graw Hill-Microsoft

Ejemplo de publicación con método Harvard

1.3 Esquema propuesto

Actualmente existe un gran número de formatos para realizar los reportes generados en diversas instituciones, analizando los estándares existentes para publicación de documentos en forma electrónica, se encontró que la mayoría intenta definir esquemas de acuerdo a las necesidades propias, derivado de esto y adoptando las mejores prácticas de dichos estándares, se observó que el modelo o esquema que más se adapta a la generación de reportes de investigación y que contiene los elementos suficientes para asistir en la solución necesaria, es el formato TEI (Text Encoding Initiative).

En éste caso en particular, se utilizará el Esquema XML, el cual permitirá aprovechar algunas de las herramientas de software que han incorporado a su manejo el uso del lenguaje XML. En este sentido los usos que se darán al esquema XML son muy variados, como por ejemplo: se usará como formato interno en determinadas herramientas (Microsoft Word y Adobe Acrobat); lo que facilitará la creación de documentos bien formados y válidos basados en el esquema TEI, una vez creados, éstos pueden ser transformados a una versión en formato HTML para ver el archivo mediante un navegador de Internet, de igual manera se ofrecerá una versión PDF, e incluso se permitirá al usuario acceder directamente a la documentación en formato XML para realizar búsquedas sobre datos estructurados; una vez que estos hayan sido almacenados en una base de datos, lo que facilitará el intercambio de información y su utilización con distintas herramientas informáticas.

Algunas de las características más importantes que favorecieron la elección del esquema TEI, es el uso de la cabecera TEIH, así como, la especificación de uno o más autores, desglose para Resumen y/o Abstract, la especificación de palabras clave, la definición de elementos para incluir imágenes o figuras, párrafos específicos y la bibliografía necesaria.

El esquema utilizado, se detalla en el Apéndice F, pero una parte del esquema utilizado se muestra a continuación.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
        <xs:import namespace="http://www.w3.org/XML/1998/namespace"/>
        <xs:element name="TEI2">
                <xs:complexType>
                        <xs:sequence>
                                <xs:element ref="teiHeader"/>
                                <xs:element ref="text"/>
                        </xs:sequence>
                </xs:complexType>
……
        <xs:element name="author">
                <xs:complexType mixed="true"/>
        </xs:element>
……
        <xs:element name="bibl">
                <xs:complexType mixed="true">
                        <xs:choice minOccurs="0" maxOccurs="unbounded">
                                <xs:element ref="author"/>
                                <xs:element ref="biblScope"/>
                                <xs:element ref="date"/>
                                <xs:element ref="pubPlace"/>
```

Elemento principal

Bibliografía

```
                        <xs:element ref="title"/>
                        <xs:element ref="publisher"/>
                        <xs:element ref="url"/>
                    </xs:choice>
                </xs:complexType>
            </xs:element>
……
            <xs:element name="div1">
                <xs:complexType mixed="true">
                    <xs:choice minOccurs="0" maxOccurs="unbounded">
                        <xs:element ref="def"/>
                        <xs:element ref="div2"/>
                        <xs:element ref="head"/>
                        <xs:element ref="listBibl"/>
                        <xs:element ref="listInfo"/>
                        <xs:element ref="note"/>
                        <xs:element ref="p"/>
                        <xs:element ref="table"/>
                        <xs:element ref="term"/>
                        <xs:element ref="list"/>
                        <xs:element ref="formula"/>
                        <xs:element ref="figure"/>
                    </xs:choice>
                    <xs:attribute name="n" type="xs:NMTOKEN"/>
                    <xs:attribute name="name" type="xs:anySimpleType"/>
                    <xs:attribute name="type" type="xs:NMTOKEN" use="required"/>
                </xs:complexType>
            </xs:element>
……
            <xs:element name="figure">
                <xs:complexType>
                    <xs:sequence minOccurs="0">
                        <xs:element ref="head"/>
                        <xs:element ref="figDesc"/>
                    </xs:sequence>
                    <xs:attribute name="entity" type="xs:ENTITY"/>
                    <xs:attribute name="TEIform" type="xs:anySimpleType" default="formula"/>
                </xs:complexType>
            </xs:element>
            <xs:element name="figDesc">
                <xs:complexType mixed="true">
                    <xs:choice minOccurs="0" maxOccurs="unbounded">
                        <xs:element ref="num"/>
                        <xs:element ref="title"/>
                        <xs:element ref="figure"/>
                    </xs:choice>
                    <xs:attribute name="id" type="xs:ID"/>
                    <xs:attribute name="n" type="xs:anySimpleType"/>
                    <xs:attribute name="lang" type="xs:IDREF"/>
                    <xs:attribute name="rend" type="xs:anySimpleType"/>
                    <xs:attribute name="TEIform" type="xs:anySimpleType" default="figDesc"/>
                </xs:complexType>
            </xs:element>
……
            <xs:element name="docAuthor">
                <xs:complexType mixed="true">
```

Elemento para divisiones del documento

Elemento para incluir figuras

Autor del Reporte

```
                    <xs:choice minOccurs="0" maxOccurs="unbounded">
                        <xs:element ref="note"/>
                    </xs:choice>
                </xs:complexType>
            </xs:element>
……
            <xs:element name="head">
                <xs:complexType mixed="true">
                    <xs:choice minOccurs="0" maxOccurs="unbounded">
                        <xs:element ref="p"/>
                    </xs:choice>
                    <xs:attribute name="rend" type="xs:NMTOKEN"/>
                    <xs:attribute name="type" type="xs:NMTOKEN"/>
                </xs:complexType>
            </xs:element>
……
            <xs:element name="keywords">
                <xs:complexType mixed="true"/>
            </xs:element>
            <xs:element name="listBibl">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element ref="bibl" maxOccurs="unbounded"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
……
            <xs:element name="p">
                <xs:complexType mixed="true">
                    <xs:choice minOccurs="0" maxOccurs="unbounded">
                        <xs:element ref="abbr"/>
                        <xs:element ref="date"/>
                        <xs:element ref="num"/>
                        <xs:element ref="note"/>
                    </xs:choice>
                </xs:complexType>
            </xs:element>
……
            <xs:element name="teiHeader">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element ref="fileDesc"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
……
            <xs:element name="title">
                <xs:complexType mixed="true"/>
            </xs:element>
            <xs:element name="titlePage">
                <xs:complexType mixed="true">
                    <xs:choice minOccurs="0" maxOccurs="unbounded">
                        <xs:element ref="docDate"/>
                        <xs:element ref="docTitle"/>
                        <xs:element ref="for"/>
                        <xs:element ref="from"/>
```

Elemento para palabras clave

Elemento para párrafos

Título del documento

```
                    </xs:choice>
                </xs:complexType>
            </xs:element>
……
            <xs:notation name="png" public="-//TEI//NOTATION IETF RFC2083 Portable Network
Graphics//EN"/>
            <xs:notation name="jpeg" public="ISO DIS 10918//NOTATION JPEG Graphics Format//EN"/>
</xs:schema>
```

Aprovechando la definición del TEI, se está en posibilidad de proporcionar una herramienta para la generación de documentos XML, y de esta manera facilitar la captura de información para su incorporación en una base de datos XML, para su explotación posterior.

CAPÍTULO 2

Bases de datos XML

Las bases de datos forman parte esencial en todas las organizaciones, ya que en ellas se almacena toda la información generada, derivada de su operación, mucha de esta información es necesario guardarla en formato XML para facilitar el análisis de su contenido, para ello, se utiliza un modelo especial, llamado Base de Datos XML.

Una Base de Datos XML define un modelo específico de un documento XML, para almacenarlos y administrarlos de una manera eficiente. Se presentan dos tipos: Habilitadas y Nativas.

2.1 Bases de datos XML Habilitadas

Bases de datos habilitadas para XML (XML Enabled Databases) son básicamente bases de datos relacionales que permiten el almacenamiento en forma tabular, esto es, en forma de tablas, registros y campos o columnas, definiendo un tipo de dato especial para guardar documentos XML, permiten el desglose de la información de un documento XML mediante un esquema relacional o de objetos, de igual manera estas bases de datos permiten la realización de consultas que presentan su resultado en formato XML; es por ello que reciben el nombre de "Bases de Datos habilitadas para XML".

Las principales desventajas de las BD habilitadas para XML es que:

- En consultas y extracción de información no se explotan todas las ventajas del lenguaje XML.
- Son dependientes de una tecnología en especial.

- Su poder descriptivo en cuanto a la estructura de documentos XML se encuentra muy limitado.

Algunas ventajas de las bases de datos XML habilitadas son:

- Establecen un tipo de datos XML
- Siguen el modelo relacional
- Se pueden realizar consultas sencillas sobre los documentos XML almacenados

Existen varios sistemas manejadores de bases de datos XML habilitadas, ejemplo de ellas son las siguientes:

Producto	Empresa
PostgreSQL	**www.postgresql.org**
Oracle 11 g	**Oracle**
SQL Server 2005	**Microsoft**
DB2	**IBM**
MySQL	**Sun Microsystems**

2.2 Sistema manejador de bases de datos Postgres

En la Universidad Berkeley de California comenzó un proyecto de bases de datos; que tras un proceso continúo de evolución, en 1986 se crea un sistema de bases de datos llamado Postgres y para 1996 se incrementa su funcionalidad y nace PostgreSQL.

PostgreSQL es considerado por un gran número de personas, como el sistema de bases de datos de open source (código abierto) más completo. En la Fig. 1 se muestra la pantalla de inicio.

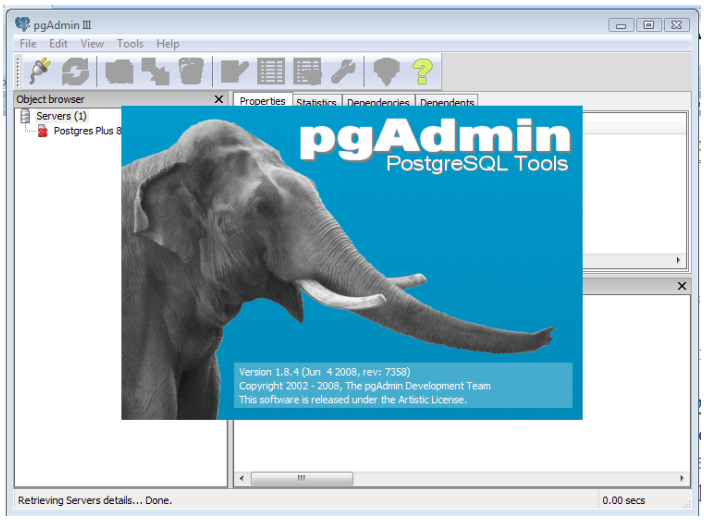


Fig. 1 Pantalla de inicio de PostgreSQL

Al ser un sistema de código abierto, se tienen diversas ventajas, entre ellas usar el programa sin tener que realizar la compra de una licencia de uso, y en un determinado caso, se puede obtener el código fuente, y éste puede ser modificado libremente, de acuerdo a las necesidades de cada usuario.

Postgres es apto para manejar complejas rutinas y reglas, y tiene una alta funcionalidad de consultas SQL declarativas, así como un control de concurrencia multi-versión, a la vez que proporciona un soporte multi-usuario, el uso de transactions, y una optimización de consultas, asimismo, incorpora de buena manera la herencia y el empleo de arrays.

De igual manera incorpora funciones avanzadas como son el uso de joins (uniones), y de otras funciones incluidas en las especificaciones SQL92 y SQL99.

Además de que cuenta con una arquitectura cliente/servidor, soporta operadores, funcionales, métodos de acceso y tipos de datos definidos por el usuario, por lo que se considera altamente extensible, para garantizar que los datos capturados son validos, se apoya en la integridad referencial. En la Fig. 2 se muestran algunos de los tipos de datos que se pueden utilizar en PostgreSQL.

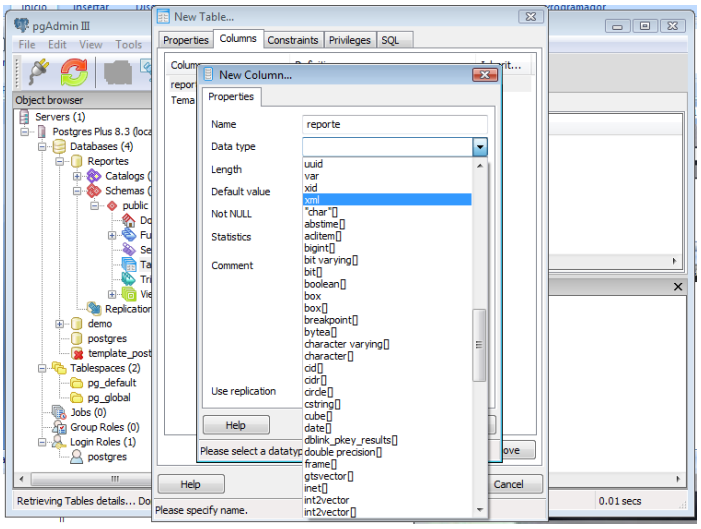


Fig. 2 Ejemplo de tipos de datos en Postgres

La API utilizada proporciona gran flexibilidad, por lo que se puede utilizar PHP, Object Pascal, Java/JDBC, Perl, C/C++, Python, etc. Así como su propio lenguaje llamado PL/pgSQL.

Para impedir bloqueos innecesarios cuenta con un Control de Concurrencia Multi-Versión (Multi-Version Concurrency Control). Lo que permite el manejo de registros sin necesidad de que los usuarios tengan que esperar a que dichos registros dejen de ser utilizados por otros usuarios.

En caso de una caída de la base de datos, se cuenta con un registro de transacciones que permite la restauración de dicha base, lo anterior mediante la característica conocida como WAL (Write Ahead Logging), así los usuarios pueden continuar su trabajo desde el punto donde fue la caída de la base de datos.

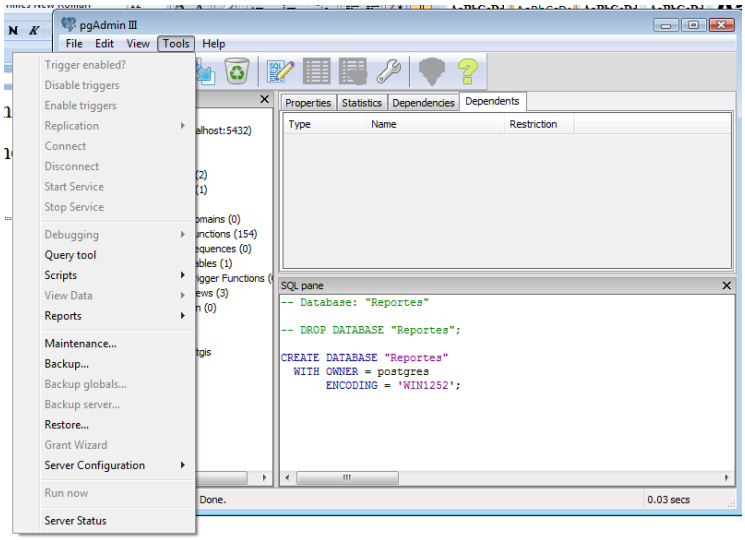


Fig. 3 Herramientas de Postgres

Si bien Postgres es un manejador de bases de datos poderoso, no es un manejador de bases de datos nativas de XML, por lo que las desventajas mencionadas anteriormente para este modelo de bases de datos, son también aplicables a Postgres, aunque presenta ya algunos elementos que permiten la explotación de este tipo de documentos XML, como son un conjunto de transformaciones XSLT y XPath, y la opción de generar reportes XML con el Report Tool (Fig.3 y 4).

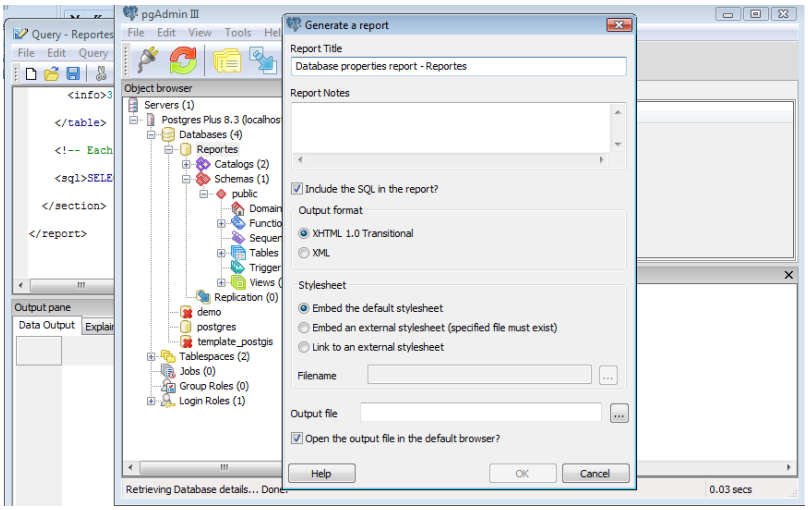


Fig. 4 Generador de reportes de Postgres

2.3 Bases de datos XML Nativas

Las Bases de datos nativas de XML (XML Native Databases), son aquellas que definen un modelo lógico para documentos XML y al momento de realizar su almacenamiento, respetan la estructura de dicho documento, en ellas se pueden hacer consultas sobre dicha estructura y es posible recuperar el documento tal como fue insertado originalmente, el cual casi siempre se almacenan en un formato BLOB (Binary Large Object).

Una base de datos nativa de XML, sólo almacena documentos XML, no cuenta con campos, ni almacena datos atómicos, permite la creación de catálogos a manera de tablas para una mejor organización de la información, pero es importante mencionar que no son tablas, es por ello que estas bases de datos también pueden ser centradas en documentos (data centric databases).

Las principales características de las bases de datos nativas de XML son:

- El almacenamiento de la información es en formato XML, con repositorios de formato tipo XML, como puede ser DOM.
- Al momento de guardar los documentos XML, se generan índices que permiten acelerar las búsquedas, dichos índices se guardan en el mismo repositorio.

Para la realización de búsquedas, se pueden utilizar los lenguaje de consulta Xpath, XPointer y XQuery, mediante los cuales se pueden llevar a cabo búsquedas hasta a nivel de documento y extraer un tipo de información en específico.

En el proceso de actualización y borrado de documentos, muchos sistemas soportan lenguajes como XUpdate, que es un lenguaje enfocado a la actualización y de XLink que permite la definición de relaciones entre documentos e imágenes, archivos, etc.

Algunas de las Bases de Datos Nativas de XML de licencia comercial, son las siguientes:

Producto	Empresa
Birdstep	**Birdstep**
dbXML	**dbXML Group**
DOM-Safe	**Ellipsis**
MindSuite XDB	**Wired Minds**
GoXML DB	**XML Global**
TEXTML Server	**IXIA, Inc.**

Algunas de las Bases de Datos Nativas de XML Open Source, son las siguientes:

Producto	Empresa
4Suite, 4Suite Server	**FourThought**
eXist	**Wolfgang Meier**
Xindice	**Apache Software Foundation**

2.4 Sistema manejador de bases de datos eXist

Como resultado de la investigación sobre las diversas herramientas existentes para el manejo de documentos XML, sobresale la base de datos nativa de XML **eXist**, que aparte de ser un proyecto Open Source (http://exist.sourceforge.net/), cuenta entre sus múltiples ventajas que está completamente escrito en Java, lo cual asegura su portabilidad e independencia de plataforma. En la Fig. 5 se puede observar la pantalla inicial de eXist.

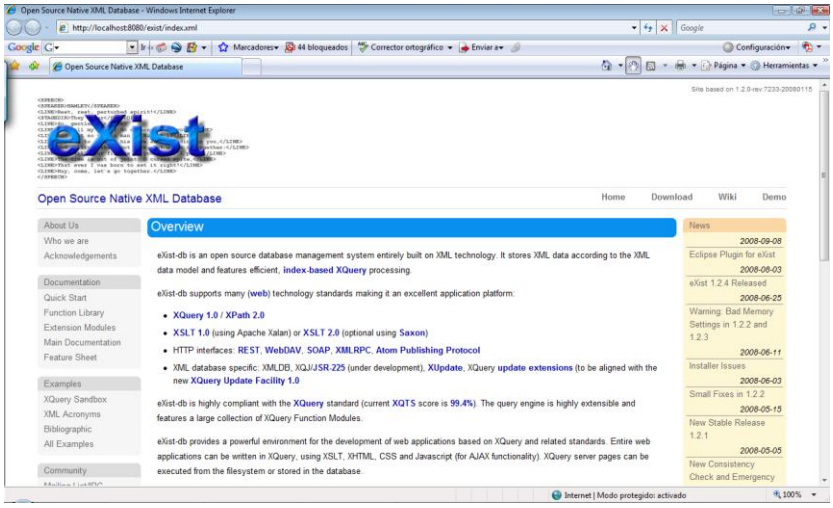


Fig. 5 Pantalla inicial de eXist

Proporciona un potente entorno para el desarrollo de aplicaciones web basadas en XQuery, así mismo, cuenta con su propio motor de búsqueda basado en XQuery (Fig. 6), que puede ser ejecutado desde el sistema de archivos o mediante procedimientos almacenados en la base de datos.

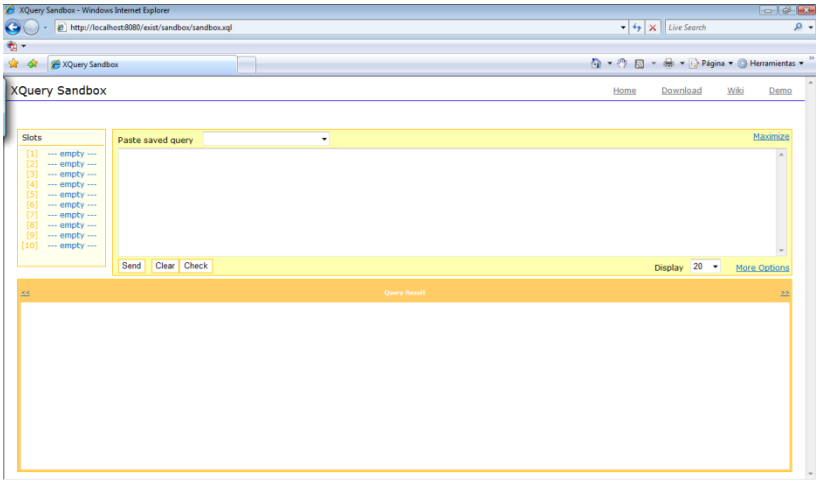


Fig. 6 Modulo de XQuery

Las colecciones de datos se encuentran en un modo jerárquico, similar al sistema de archivos, dichas colecciones juegan el papel de las tablas en las BD relacionales, en ellas, los documentos se suelen agrupar, en función de la información que contienen, en

colecciones que a su vez pueden contener otras colecciones. Como se puede observar en la Fig. 7.

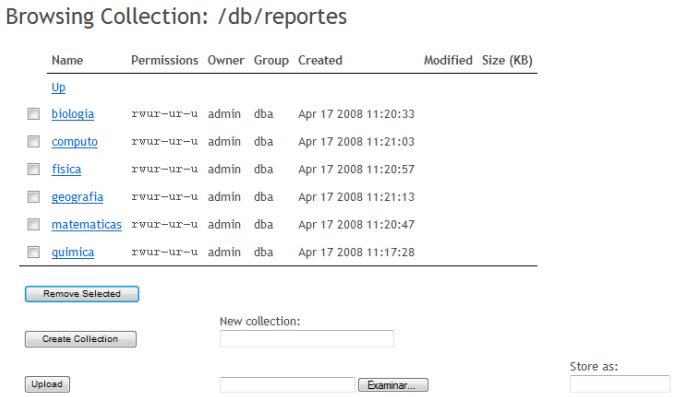


Fig. 7 Ejemplo de colecciones en eXist

Para evitar el uso excesivo de memoria (estructuras de árbol), eXist utiliza una eficiente estructura de los índices B+, basada en un sistema numérico de indexación para la identificación de los nodos XML mediante DOM. A cada documento XML le asocia un identificador único por el que será reconocido dentro del repositorio. También permite la creación de índices que aceleran las consultas realizadas frecuentemente.

Aunque inicialmente no se encuentra vinculado a un esquema específico (puede almacenar cualquier tipo de documento XML), se puede definir un modelo lógico para un documento XML (para el documento, no para los datos) y con ello se asegura que el almacenamiento y recuperación de documentos se llevará a cabo de acuerdo con ese modelo, realizando así una validación previa.

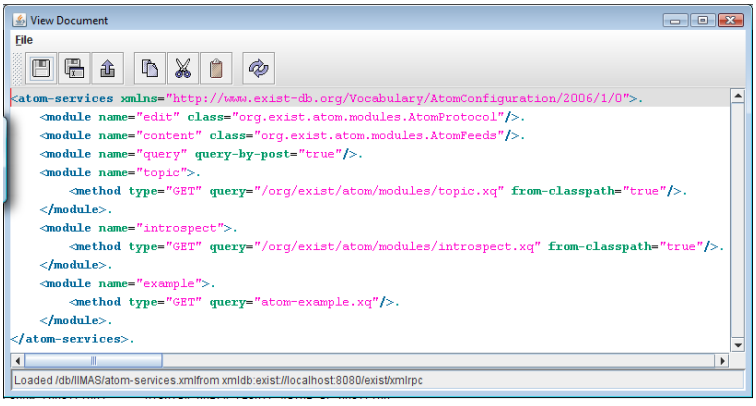


Fig. 8 Fragmento de documento XML, en eXist

Contiene un visor de documentos XML (Fig.8), el número máximo de documentos almacenados es de 2^{31} y el tamaño de cada documento, depende del sistema operativo en el cual se esté trabajando.

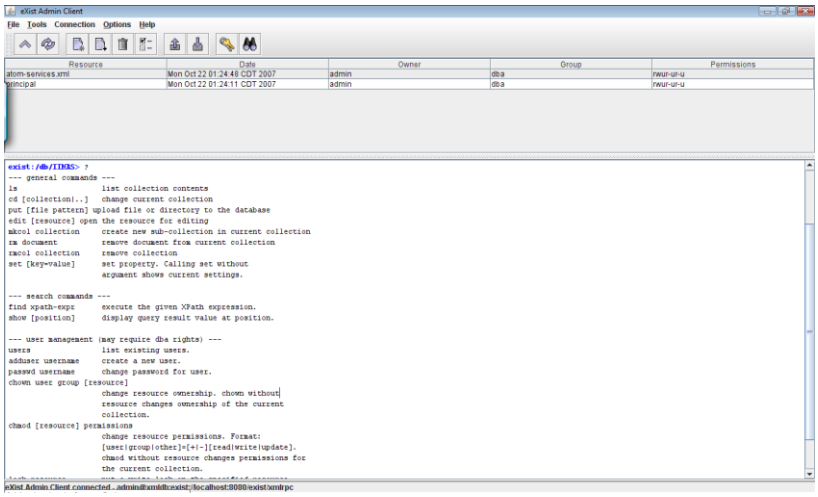


Fig. 9 Pantalla ejemplo de eXist

Para el desarrollo de aplicaciones en eXist se puede utilizar XQuery, XSLT, XHTML, CSS y quizás Javascript (para la funcionalidad de AJAX).

En caso de tener un modelo definido, éste deberá incluir como mínimo, elementos, atributos, manejo de PCDATA y un orden dentro del documento.

El acceso a la base de datos se puede realizar mediante dos tipos de interfaz, una que permite especificar el usuario, el tipo de acceso es remoto o local y a que colección se deberá conectar (Fig. 10). Y otra donde solo se específica el usuario (Fig. 11).

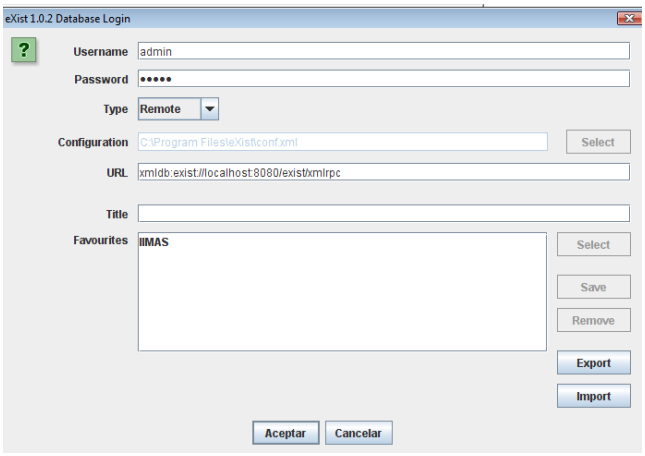


Fig. 10 Modulo de acceso a la base de datos en eXist

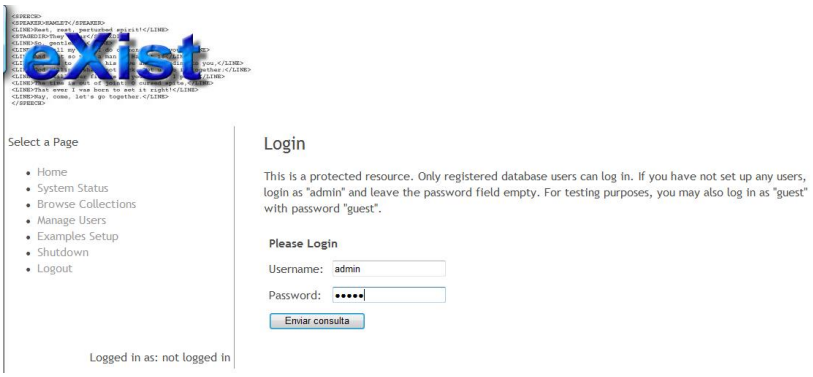


Fig. 11 Pantalla de acceso de eXist

CAPÍTULO 3

Metodología utilizada para la solución propuesta

3.1 Requerimientos

El objetivo es auxiliar en la elaboración de Reportes de Investigación, generados en diversas organizaciones, y a la vez realizar un marcado XML de los mismos, de acuerdo a un esquema estándar que contemple los datos necesarios para la correcta elaboración de los reportes, con la finalidad de poder intercambiar y explotar la información que contienen.

Para incrementar el poder de análisis de la información será necesario el "Diseño de una Base de Datos XML, para el almacenamiento y recuperación de reportes de investigación".

Es importante mencionar que los usuarios finales, no estarán obligados a conocer el lenguaje XML, aunque esto facilitaría el uso y comprensión de la aplicación resultante.

La mayoría de usuarios, posee ciertos conocimientos sobre el uso de procesadores de texto, lo cual pudiera significar una ventaja al momento de utilizar la aplicación resultante y con ello un ahorro significativo en la capacitación necesaria para su operación.

3.2 Especificaciones

El resultado final deberá solventar por lo menos los siguientes puntos:

1. Se contará con un esquema que permita la especificación de documentos XML.
2. La aplicación final tendrá que generar Reportes de Investigación, los cuales a su vez deberán ser documentos XML, bien formados y válidos.

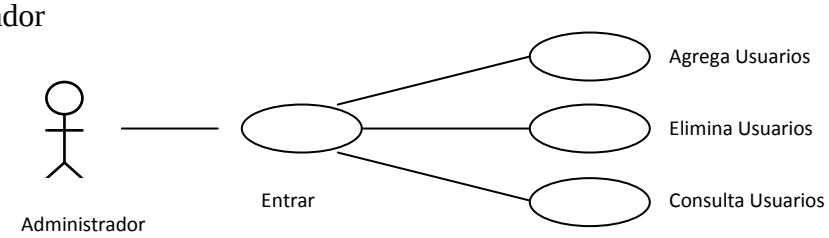
3. Estos documentos XML tendrán que ser almacenados en una base de datos especial, para su posterior explotación.
4. Se diseñará una base de datos XML para el almacenamiento y recuperación de reportes de investigación.
5. La base de datos especial, permitirá compartir la información contenida en los Reportes de investigación.
6. Se contará por lo menos con dos herramientas para el diseño de los reportes de investigación, una de las cuales deberá ser usada mediante el programa Microsoft Word (para aprovechar los conocimientos que los usuarios poseen en el uso de procesadores de texto).
7. La capacitación del sistema deberá ser sencilla y rápida.
8. El sistema protegerá los documentos XML, para que éstos no puedan ser modificados en su estructura, de manera involuntaria o derivados de un error.

3.3 Análisis (algoritmos y diagramas de flujo)

3.3.1 Casos de Uso

Caso de uso: Entrar (al sistema)

Actor: Administrador



Caso de uso Entrar (al Sistema)

Descripción: El Administrador desea entrar al sistema para dar de Alta, Eliminar o Consultar usuarios.

Precondiciones:

- El Administrador debe tener abierto un navegador .
- El Administrador debe teclear la dirección adecuada para acceder a la página de la Base de Datos de Reportes XML.

Flujo:

ACTOR		SISTEMA		
Paso	**Acción**	**Paso**	**Acción**	**Excepción**
1	Teclea en su navegador la dirección de la página del sistema	2	Despliega la interfaz principal para seleccionar entre Entrar, Salir o Acerca de…	E1
3	Selecciona el botón Entrar	4	Despliega la interfaz de Administrador, solicitando los datos para su ingreso	

Excepciones:

Id	Nombre	Acción
E1	Dirección incorrecta	La página no se despliega en el navegador

Postcondiciones:

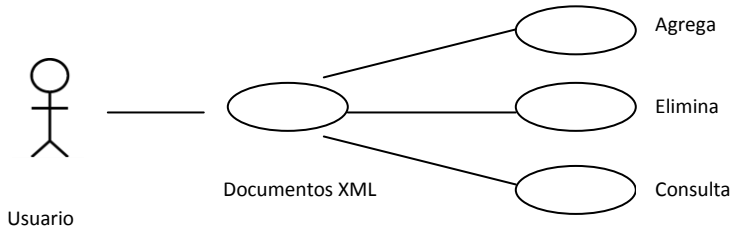
- El sistema ha desplegado la interfaz de Administrador.

Plan de Pruebas del Sistema

Caso de uso	Entrada	Esperado	Obtenido
Entrar	Dirección de la página	Dirección correcta	Despliega interfaz inicial
	Dirección de la página	Dirección incorrecta	El navegador no despliega interfaz inicial

Caso de uso: Administrar Documentos XML

Actor: Usuario



Caso de uso Administrar Documentos XML

Descripción: El Usuario debe elegir una opción del menú Documentos XML, las cuales pueden ser Agrega, Elimina o Consulta.

Precondiciones:

- El sistema despliega interfaz del menú Documentos XML.

Flujo:

ACTOR		SISTEMA		
Paso	**Acción**	**Paso**	**Acción**	**Excepción**
1	Elige opción Agrega	2	Despliega la interfaz Agrega Documento XML	E1
1	Elige opción Elimina	2	Despliega la interfaz Elimina Documento XML	E1
1	Elige opción Consulta	2	Despliega la interfaz Consulta Documento XML	E1
1	Elige Botón Salir	2	Sale del Sistema	E1

Excepciones:

Id	Nombre	Acción
E1	Red desconectada	Mostrar Mensaje de “Red desconectada”

Postcondiciones:

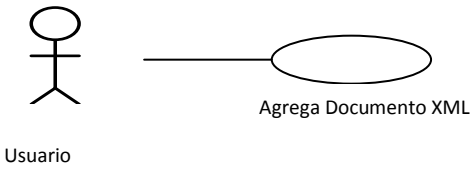
- El sistema presenta la Interfaz de Agrega, Elimina o Consulta Documentos XML.
- El sistema se cierra.

Plan de Pruebas del Sistema

Caso de uso	Entrada	Esperado	Obtenido
Administrar Documentos XML	Agrega Documento XML	Opción Agrega Documento XML	Interfaz Agrega Documento XML
	Elimina Documento XML	Opción Elimina Documento XML	Interfaz Elimina Documento XML
	Consulta Documento XML	Opción Consulta Documento XML	Interfaz Consulta Documento XML
	Salir	Botón Salir	Cierre del Sistema

Caso de uso: Agrega Documento XML

Actor: Usuario



Caso de uso Agrega Documento XML.

Descripción: El Usuario proporcionará los datos necesarios (Nombre o ubicación del Documento XML) para Agregar el Documento XML a la Base de Datos.

Precondiciones:

- El sistema presenta la Interfaz de Agrega Documento XML.

Flujo:

ACTOR		SISTEMA		
Paso	**Acción**	**Paso**	**Acción**	**Excepción**
1	Teclea todos los datos solicitados para Agregar un Documento XML a la Base de Datos			
2	Presiona Botón Aceptar	3	Verifica que todos los datos se hayan capturado de manera correcta y que el Documento XML exista	E1, E2
		4	Verifica que el Documento XML no exista en la BD	E3
		5	Guarda la información y el Documento XML en la Base de Datos	
1	Presiona Botón Cancelar	2	Regresa a la Interfaz del menú Documentos XML	

Excepciones:

Id	Nombre	Acción
E1	Faltan datos	Presenta mensaje de Error “Falta el dato _____, favor de capturarlo”
E2	Documento XML no existe en esa ruta	Presenta mensaje “Documento XML no encontrado”
E3	Documento XML duplicado	Presenta mensaje “Documento duplicado”

Postcondiciones:

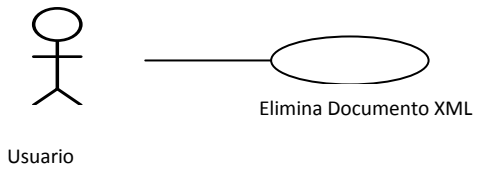
- El sistema guarda la información y el Documento XML en la Base de Datos.
- El sistema regresa a la interfaz de menú Documentos XML.

Plan de Pruebas del Sistema

Caso de uso	Entrada	Esperado	Obtenido
Agrega Documento XML	Botón Aceptar	Datos completos y correctos del Documento XML	Guarda de Información en la Base de Datos
	Botón Aceptar	Datos incompletos y correctos del Documento XML	Mensaje de Error "Falta el dato _____, favor de capturarlo"
	Botón Aceptar	Datos completos e incorrectos del Documento XML	Mensaje de Error "Documento XML no encontrado"
	Botón Aceptar	Documento XML duplicado	Presenta mensaje "Documento duplicado"
	Cancelar	Botón Cancelar	Regresa a la Interfaz del menú Documentos XML

Caso de uso: Elimina Documento XML

Actor: Usuario



Caso de uso Elimina Documento XML.

Descripción: El Usuario proporcionará los datos necesarios (Nombre del Documento XML) para Eliminar el Documento XML de la Base de Datos.

Precondiciones:

- El sistema presenta la Interfaz de Elimina Documento XML.

Flujo:

ACTOR		SISTEMA		
Paso	**Acción**	**Paso**	**Acción**	**Excepción**
1	Teclea todos los datos solicitados para Eliminar el Documento XML			
2	Presiona Botón Aceptar	3	Verifica que exista el Documento XML	E1
		4	Muestra mensaje de Advertencia “Esta usted seguro? S/N”	
5	Elige opción S	6	Elimina la información del Documento XML de la Base de Datos	
5	Elige opción N	6	Cancela opción de Elimina Documento XML	
1	Presiona Botón Cancelar	2	Regresa a la Interfaz del menú Documentos XML	

Excepciones:

Id	Nombre	Acción
E1	Documento XML Inexistente	Presenta mensaje de Error “El Documento XML no existe, corrija por favor”

Postcondiciones:

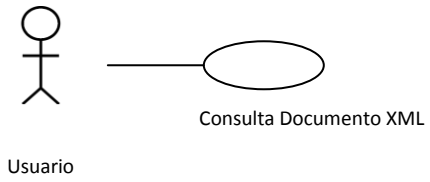
- El sistema elimina la información y el Documento XML de la Base de Datos.
- El sistema regresa a la interfaz de menú Documentos XML.

Plan de Pruebas del Sistema

Caso de uso	Entrada	Esperado	Obtenido
Elimina Documento XML	Botón Aceptar	Datos correctos del Documento XML	Elimina la información y el Documento XML de la Base de Datos
	Botón Aceptar	Datos incorrectos del Documento XML	Mensaje de Error "El Documento XML no existe, corrija por favor"
	Cancelar	Botón Cancelar	Regresa a la Interfaz del menú Documentos XML

Caso de uso: Consulta Documento XML

Actor: Usuario



Caso de uso Consulta Documento XML.

Descripción: El Usuario proporcionará los datos necesarios (Nombre del Documento XML) para Consultar el Documento XML en la Base de Datos.

Precondiciones:

- El sistema presenta la Interfaz de Consulta Documento XML.

Flujo:

ACTOR		SISTEMA		
Paso	**Acción**	**Paso**	**Acción**	**Excepción**
1	Teclea todos los datos solicitados para realizar la Consulta del Documento XML			
2	Presiona Botón Aceptar	3	Verifica que exista el Documento XML	E1
		4	Muestra información del Documento XML	
1	Presiona Botón Cancelar	2	Regresa a la Interfaz del menú Documentos XML	

Excepciones:

Id	Nombre	Acción
E1	Documento XML Inexistente	Presenta mensaje de Error "El Documento XML no existe, corrija por favor"

Postcondiciones:

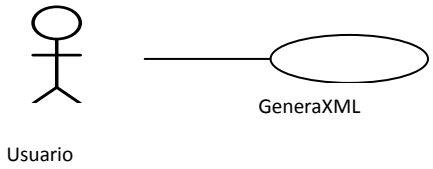
- El sistema presenta la información del Documento XML, almacenado en la Base de Datos.
- El sistema regresa a la interfaz de menú Documentos XML.

Plan de Pruebas del Sistema

Caso de uso	Entrada	Esperado	Obtenido
Consulta de Documento XML	Botón Aceptar	Datos correctos del Documento XML	Muestra información del Documento XML
	Botón Aceptar	Datos incorrectos del Documento XML	Mensaje de Error "El Documento XML no existe, corrija por favor"
	Cancelar	Botón Cancelar	Regresa a la Interfaz del menú Documentos XML

Caso de uso: Generación de Documentos XML bien formados y válidos

Actor: Usuario



Caso de uso Generación de Documentos XML bien formados y válidos

Descripción: El Usuario capturará los datos necesarios del reporte de investigación que este realizando (Titulo, Autor(es), Palabras Clave, Resumen, Abstract, Parrafos del Documento y la bibliografía correspondiente) para poder guardar el archivo como un documento de Microsoft Word o como un documento XML.

Precondiciones:

- El documento base de Microsoft Word se encuentra disponible.

Flujo:

ACTOR		SISTEMA		
Paso	**Acción**	**Paso**	**Acción**	**Excepción**
1	Abre el documento de Microsoft Word			E1
2	Teclea todos los datos necesarios generar el Reporte de investigación	3	Verifica la ortografía de la información capturada	E2
		4	Muestra faltas de ortografía, para ser corregidas	
5	Corrige faltas de ortografía			
6	Guarda documento con formato Microsoft Word	7	Verifica que no exista el documento	E3
		8	Guarda el documento	
6	Guarda documento con formato XML	7	Verifica que no exista el documento	E3
		8	Guarda el documento	

Excepciones:

Id	Nombre	Acción
E1	El Documento Word no existe	Presenta mensaje de Error “El Documento no existe”
E2	Existen faltas de ortografía	Subraya las palabras mal escritas
E3	Ya existe un documento con ese nombre	Solicita nuevo nombre o muestra la opción de reemplazar el archivo

Postcondiciones:

- El documento de Microsoft Word contiene la información de Reporte de investigación.
- El documento de Microsoft Word genera un documento XML.

Plan de Pruebas del Sistema

Caso de uso	Entrada	Esperado	Obtenido
Generación de Documentos XML bien formados y válidos	Abrir archivo	Se abre el archivo y presenta los espacios para captura de información	Presenta los espacios para captura de información
	Guardar el Archivo como documento Word	Solicita nombre del archivo	Guarda el archivo
	Guardar el Archivo como documento Word	Solicita nombre del archivo	Mensaje de Error "El Documento ya existe, corrija por favor"
	Guardar el Archivo como documento XML	Solicita nombre del archivo	Guarda el archivo
	Guardar el Archivo como documento XML	Solicita nombre del archivo	Mensaje de Error "El Documento ya existe, corrija por favor"

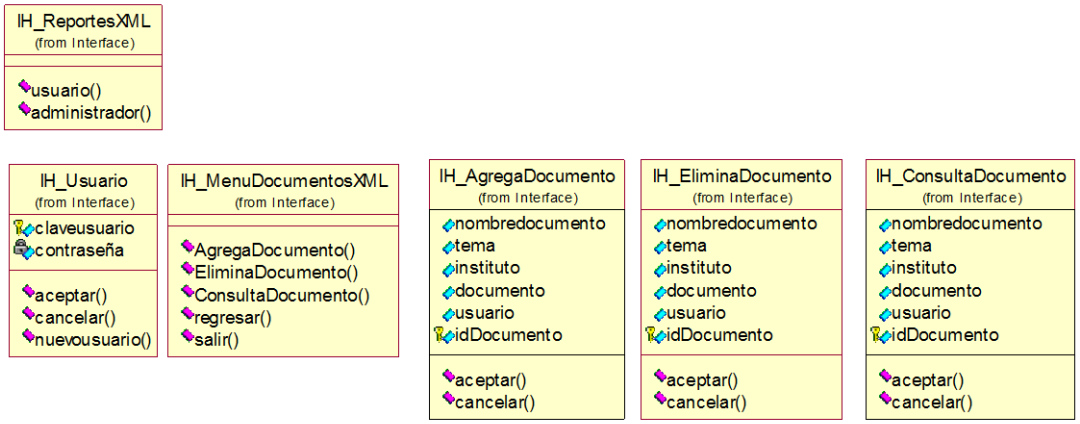
3.3.2 Diagramas de Clases

Para facilitar el análisis de los diagramas de clases, se presentan los tres tipos más comunes que son:

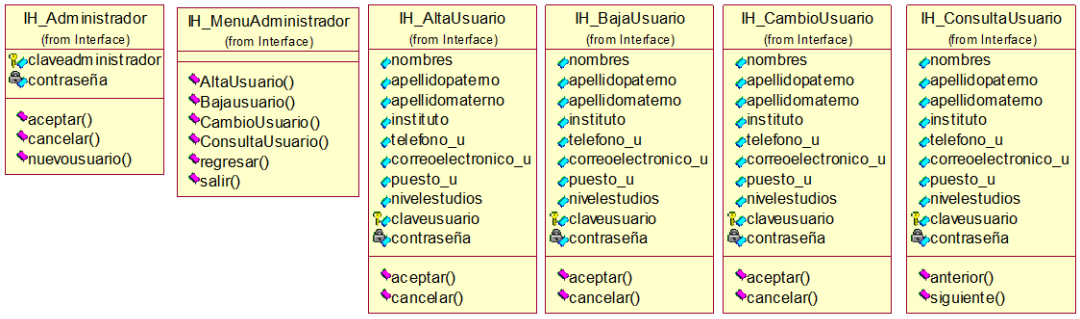
- Interfaz (Límite entre el sistema y su ambiente).
- Control: (Lógica del sistema).
- Entidad: (La información almacenada en el sistema).

Diagramas de Clases de Interfaz

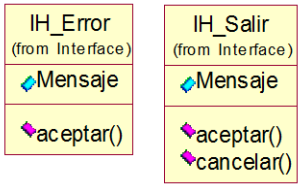
Interfaz principal y Administración de Documentos XML



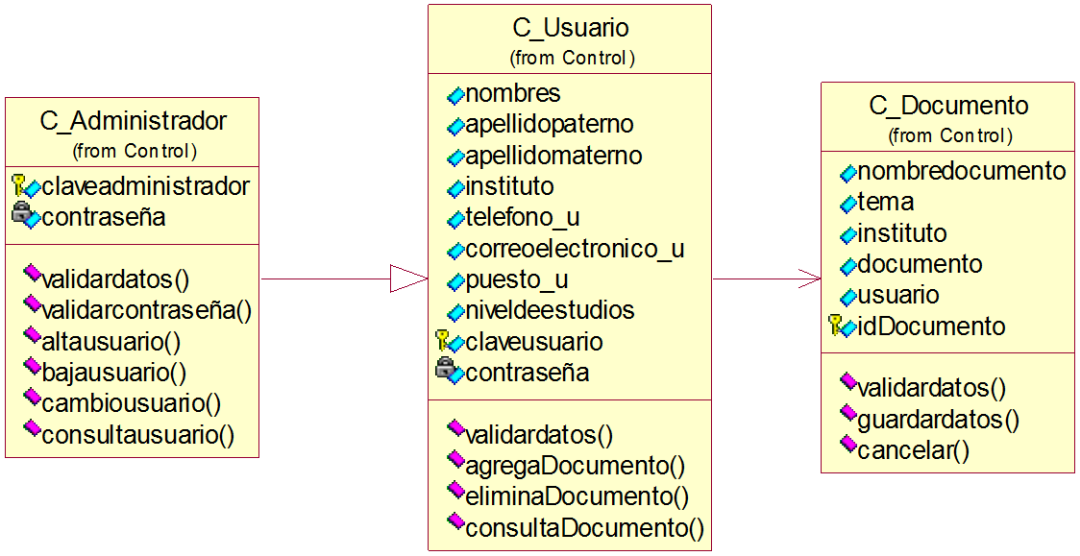
Interfaz de Administrador



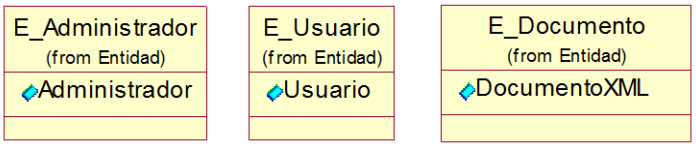
Interfaces de Error y Salir



Diagramas de Clases de Control



Diagramas de Clases de Entidad



3.3.3 Diagramas de Secuencias

Diagrama de secuencia para el caso de uso Entrar al sistema Reportes XML (Usuario).

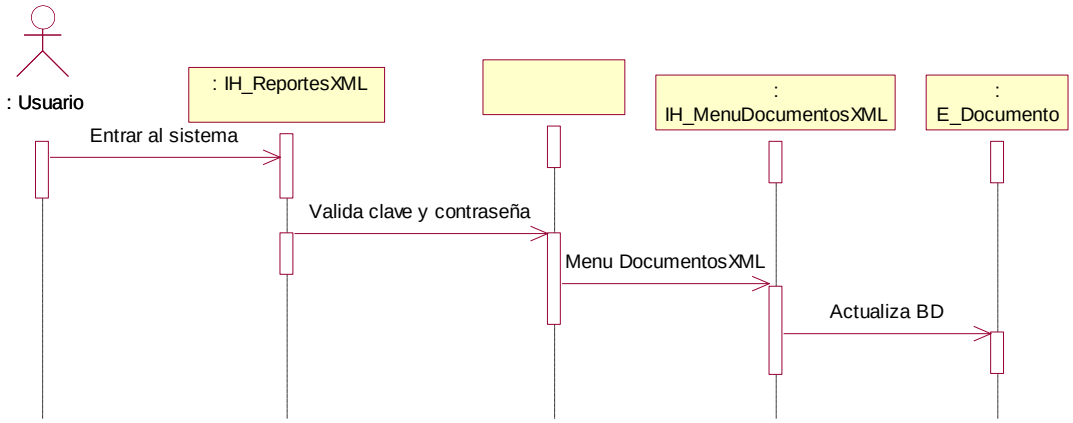


Diagrama de secuencia para el caso de uso Agrega Documento

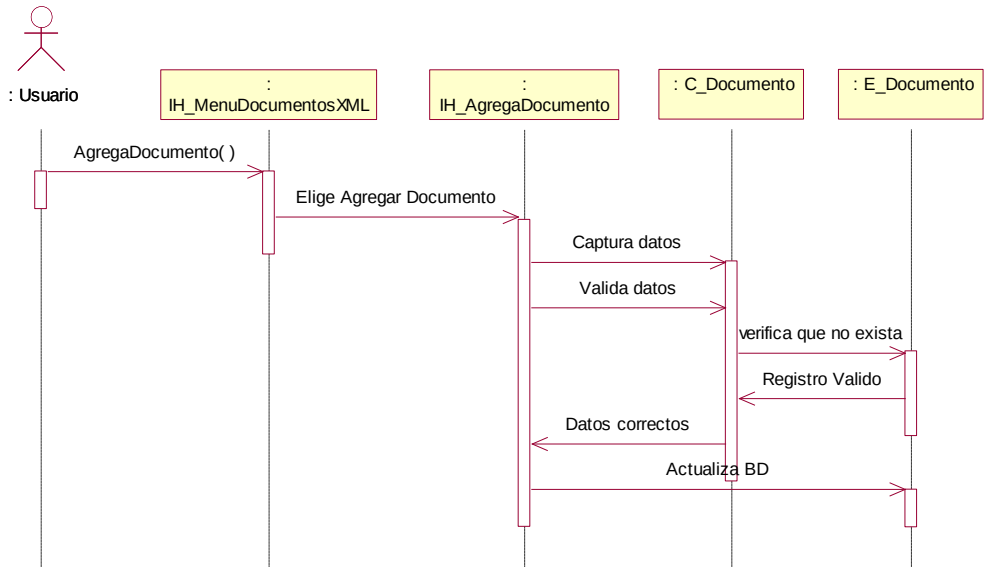


Diagrama de secuencia para el caso de uso Elimina Documento

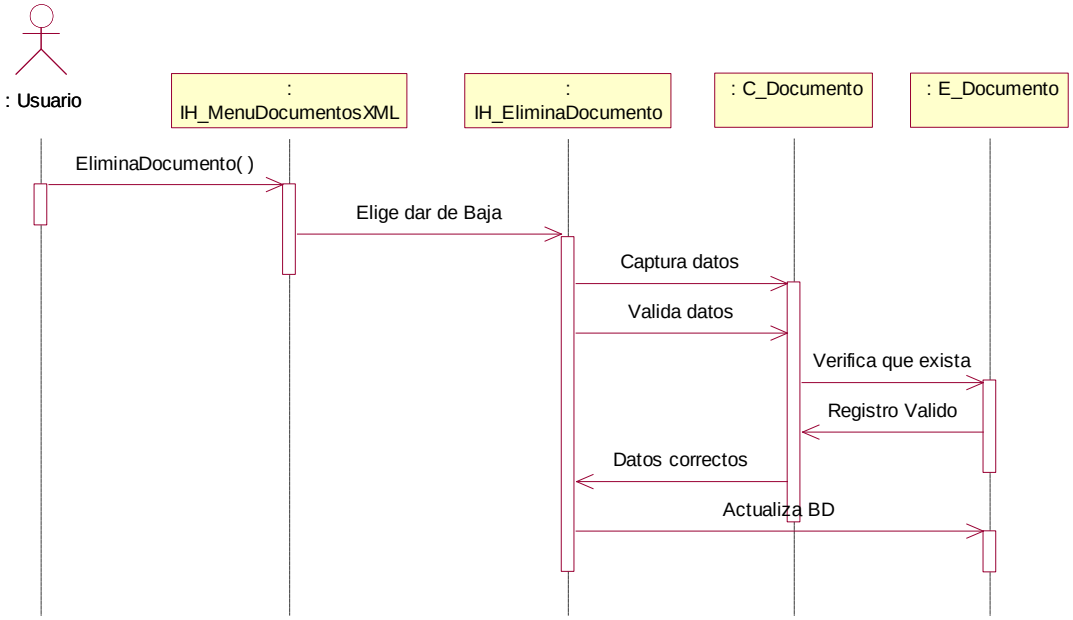


Diagrama de secuencia para el caso de uso Consulta Documento

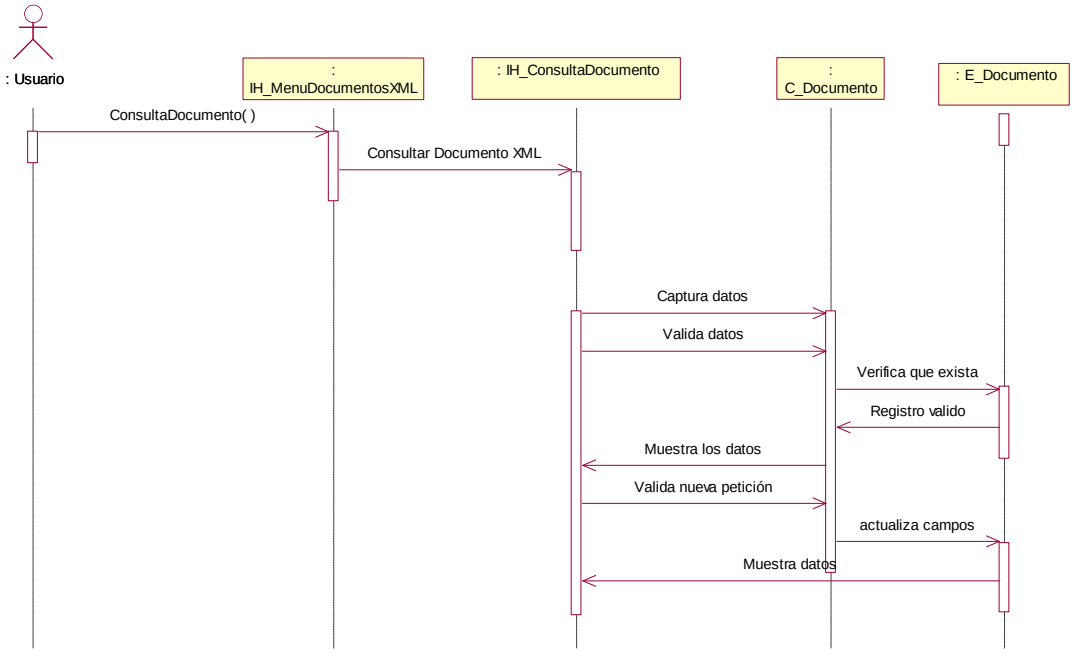


Diagrama de secuencia para el caso de uso Salir del sistema (Usuario)

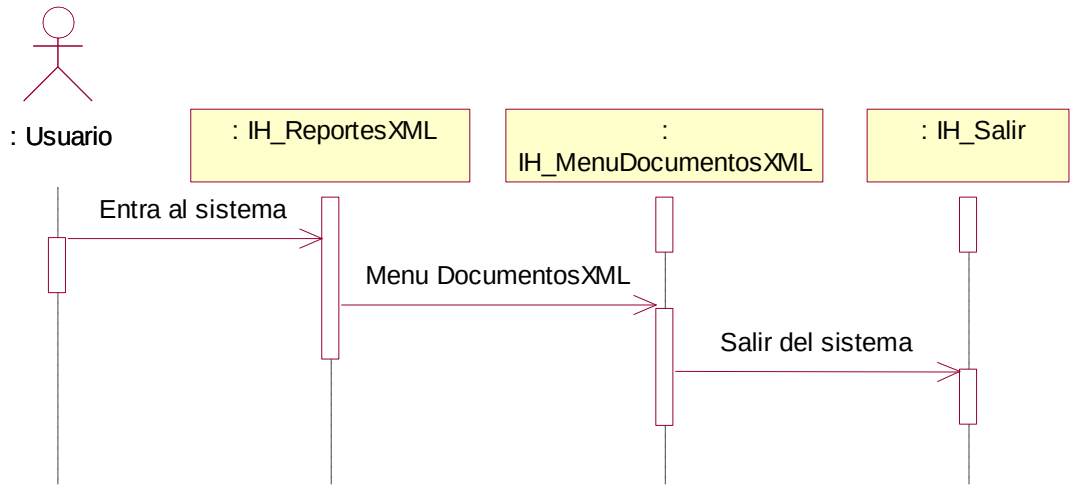


Diagrama de secuencia para el caso de uso Entrar al sistema Reportes XML (Administrador).

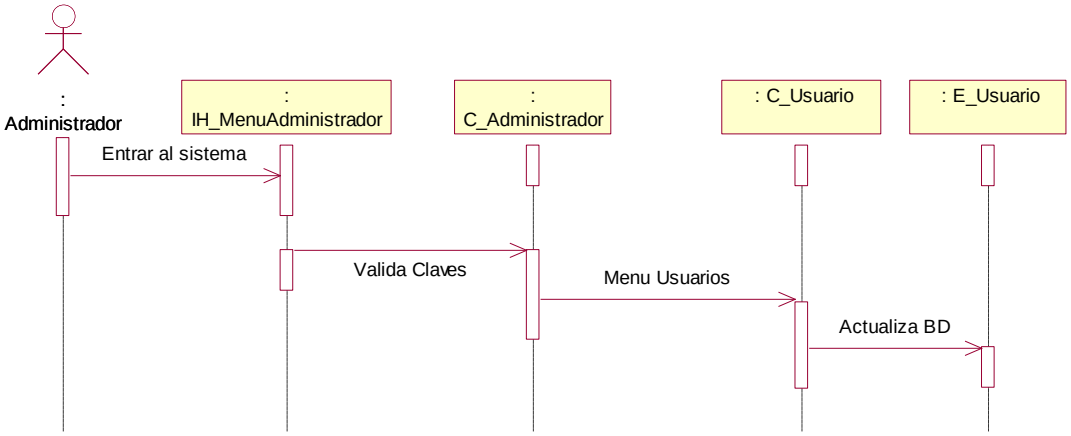


Diagrama de secuencia para el caso de uso Alta Usuario

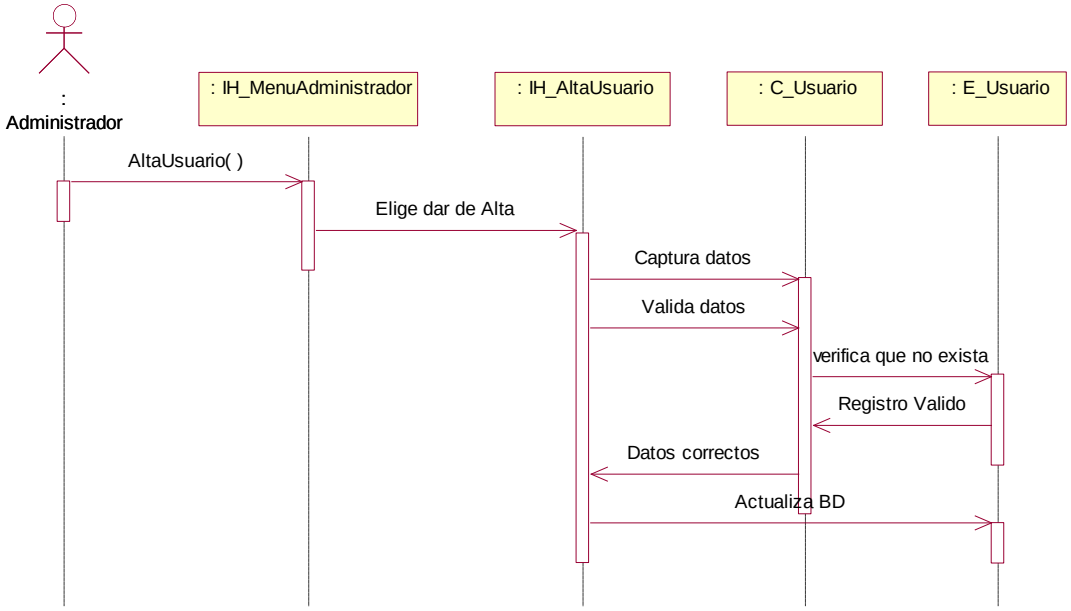


Diagrama de secuencia para el caso de uso Baja Usuario

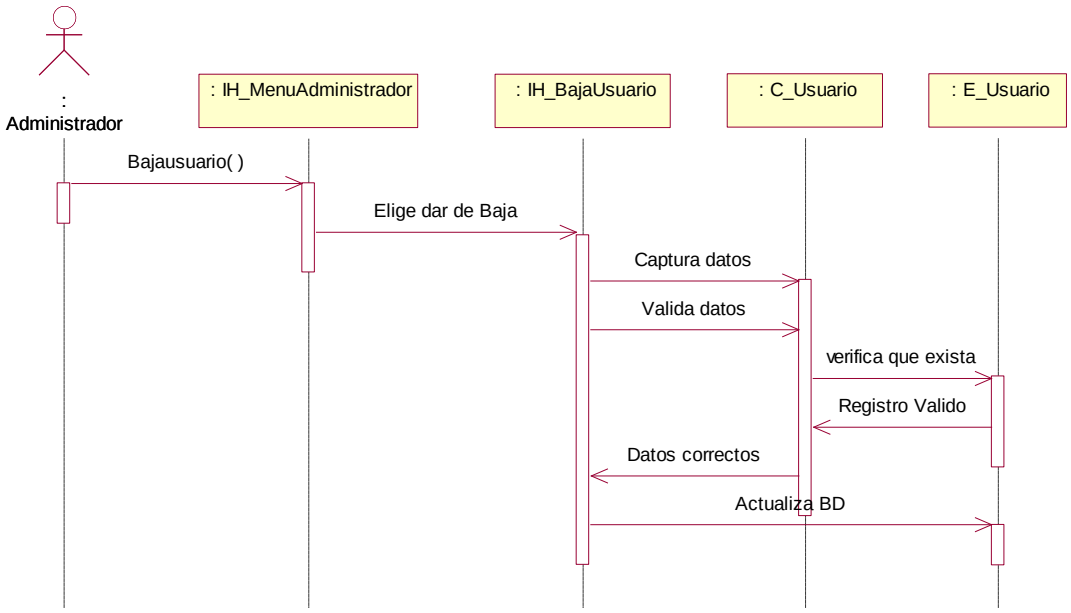


Diagrama de secuencia para el caso de uso Cambio Usuario

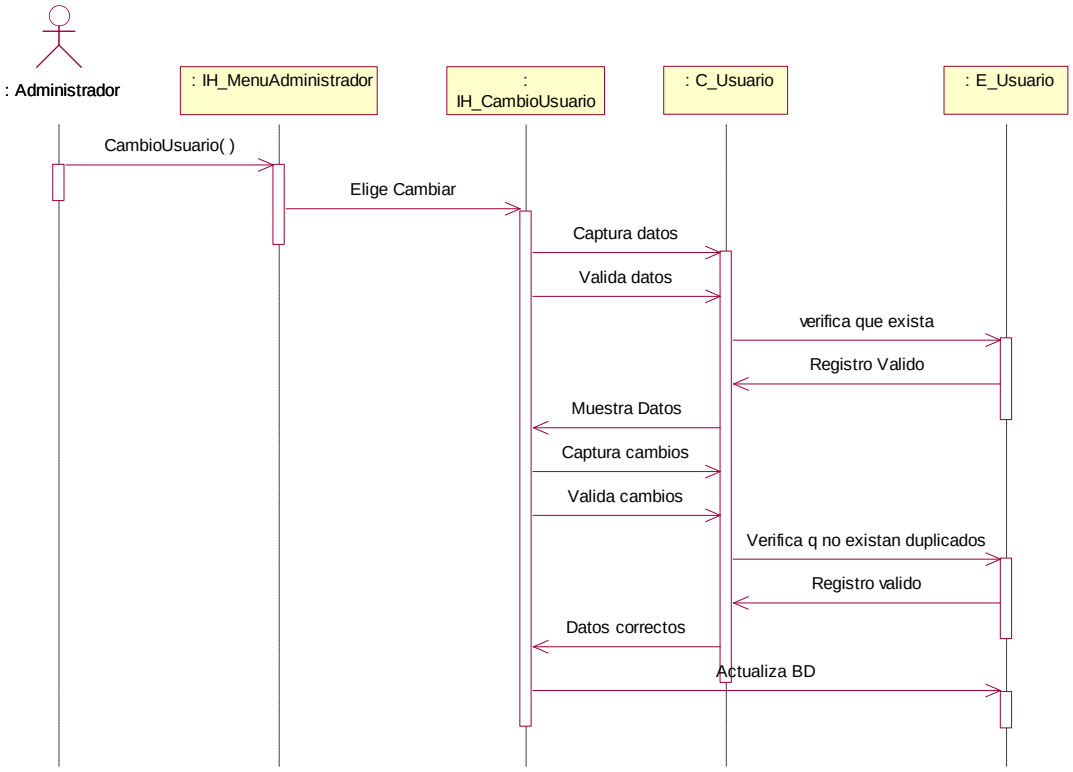


Diagrama de secuencia para el caso de uso Consulta Usuario

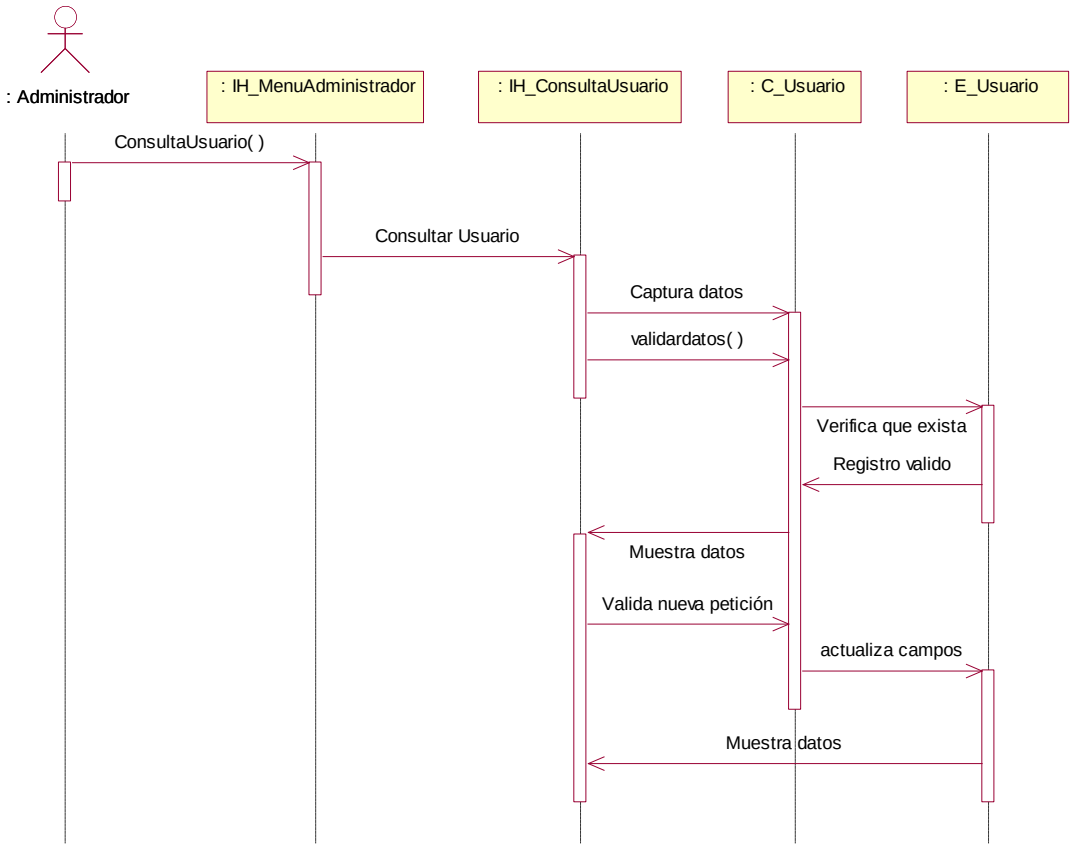
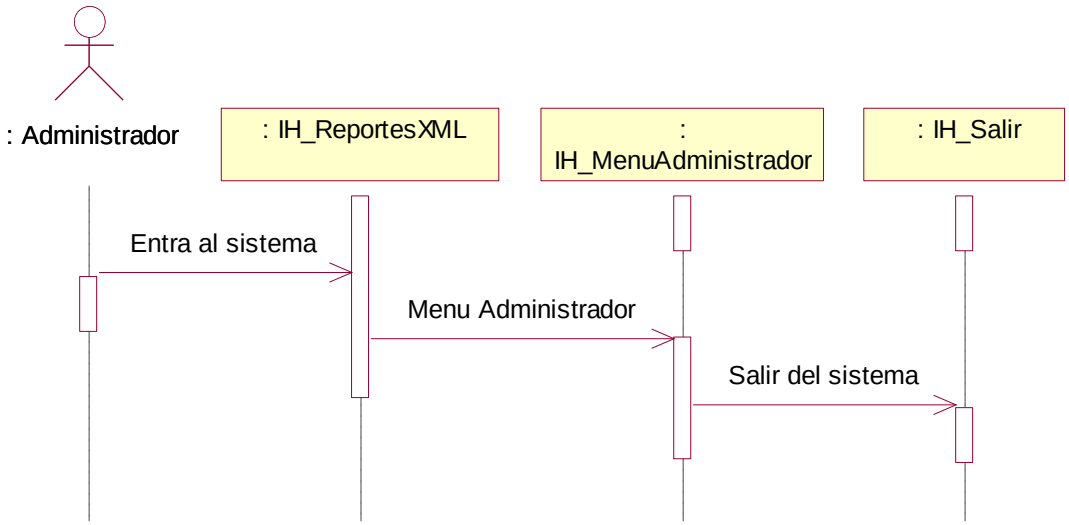
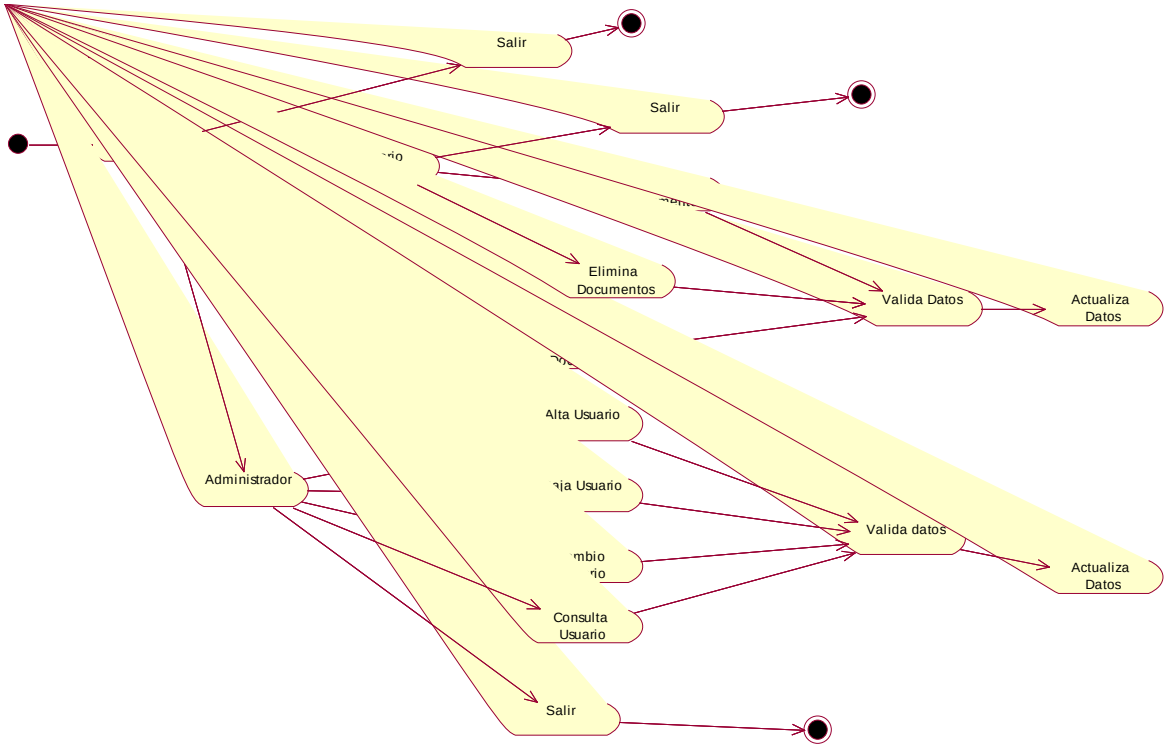


Diagrama de secuencia para el caso de uso Salir del sistema (Administrador)



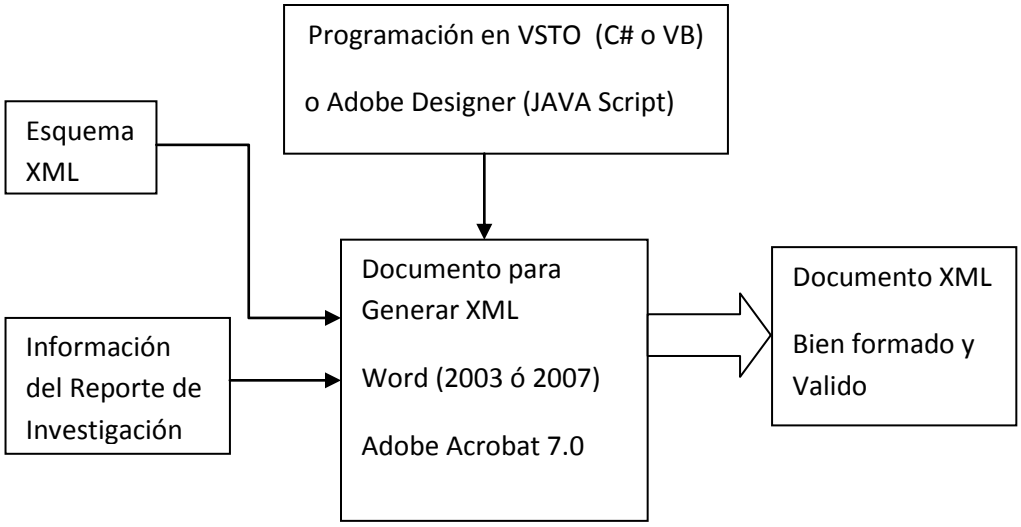
3.3.4 Diagrama de Estados



3.3.5 Diagrama de Bloques

Para la elaboración de los documentos XML, se aprovechará el conocimiento que poseen los usuarios en herramientas de procesamiento de texto, como son Microsoft Word y Adobe Acrobat.

Existen paquetes como Visual Studio Tools for Office y Adobe Designer, que permiten realizar programación a nivel de aplicación, esto es, que se puede modificar un documento ya sea de Microsoft Word o con formato PDF respectivamente, para que realice una tarea en específico, de esta manera se puede incluir en un documento de apariencia normal un esquema XML predeterminado y verificar la información que en el se va capturando, y así generar un Documento XML bien formado y válido.



Generador de Documentos XML

CAPÍTULO 4

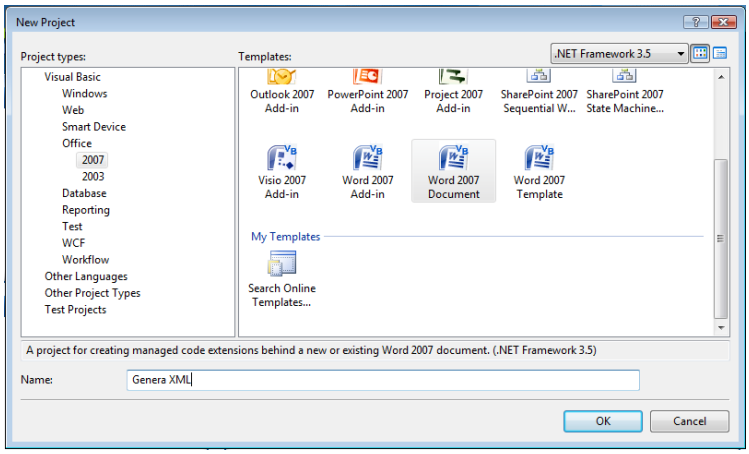
Desarrollo de la solución propuesta

4.1 Desarrollo

Una vez analizado el sistema, y observando los requerimientos por parte del Usuario, se ha elegido el programa Visual Studio (Fig. 4-1.), con su complemento Visual Studio Tools for Office, el cual, como se mencionó anteriormente, permite realizar programación a nivel de aplicaciones para los documentos de Microsoft Office.

Fig. 4-1 Pantalla de Visual Studio 2008 con el complemento VSTO

Visual Studio Tools for Office permite la programación tanto en lenguaje Visual Basic (Fig. 4-2) como en C# (Fig. 4-3), la aplicación o proyecto resultante, depende de la versión del Microsoft Office que se tenga instalada, esto es, que si se desea realizar un documento de Microsoft Word 2007 que contenga una programación específica, se tendrá que tener instalada esa versión de Microsoft Office.



(Fig. 4-2)Pantalla para crear un proyecto de Word 2007 en Visual Basic

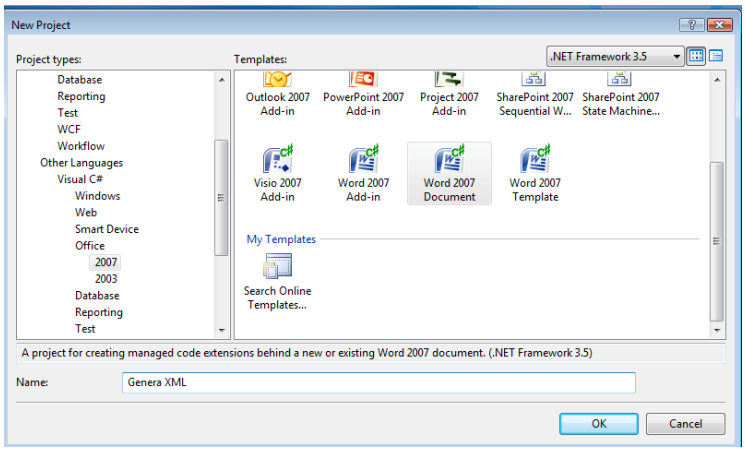


Fig. 4-3 Pantalla para crear un proyecto de Word 2007 en C#

Para proporcionar las mayores facilidades para crear documentos XML, se incluyó un Esquema XML (previamente analizado y propuesto) que permitirá validar la información que es capturada y así generar un Documento XML Bien formado y Válido.

Se implementó una solución para Microsoft Word 2007 (Fig. 4-4), y otra versión para Microsoft Word 2003, para ampliar el rango de uso de los usuarios y no restringirlos a una sola versión.

Los documentos capturados en estas soluciones podrán ser almacenados en una Base de Datos XML, ya sea nativa o habilitada.

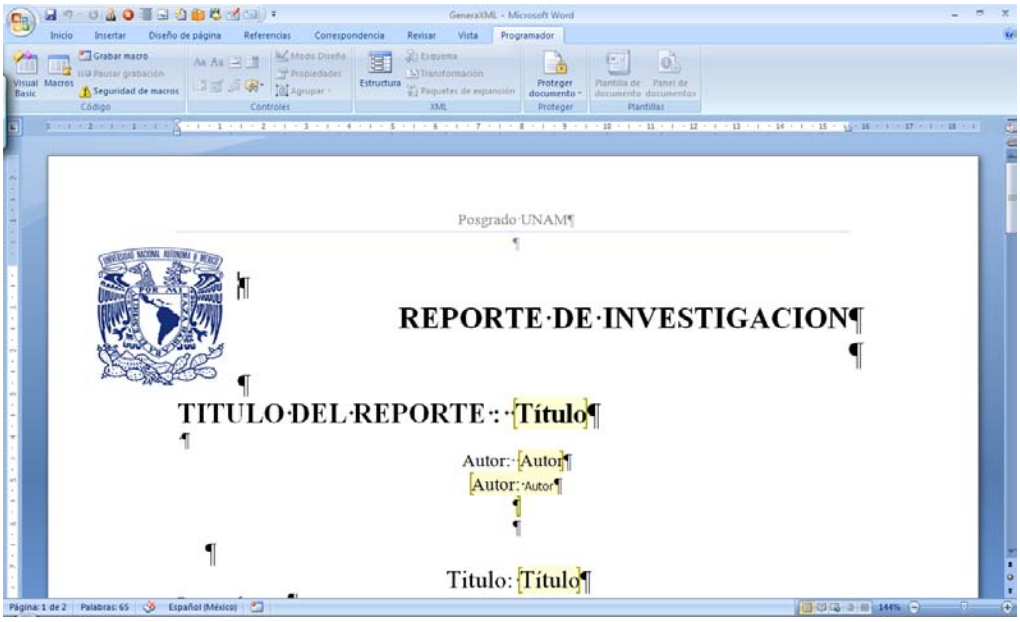


Fig. 4-4 Documento de Word 2007

La solución generada, protege la integridad del Esquema XML, para que no pueda ser modificado de manera involuntaria o por un error del usuario.

Para usuarios expertos, se pueden mostrar las etiquetas del Esquema XML (Fig. 4-5), para que sirvan de guía y facilitar la captura o modificación de la información de los Reportes de Investigación.

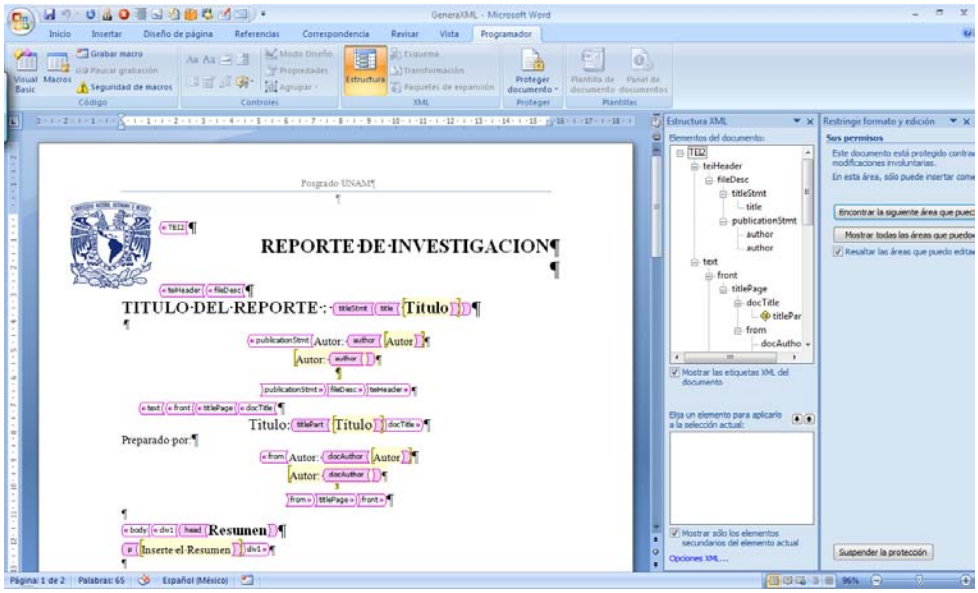


Fig. 4-5 Documento de Word 2007, que muestra etiquetas XML

Una vez que son capturados los datos, estos se pueden guardar como un archivo normal de Word (Fig. 4-6), para su posterior modificación o impresión, o si se desea, guardarlo como un Documento XML, para ello sólo es necesario indicarle a Word que el archivo es de tipo "Documento XML de Word" y asignarle un nombre.

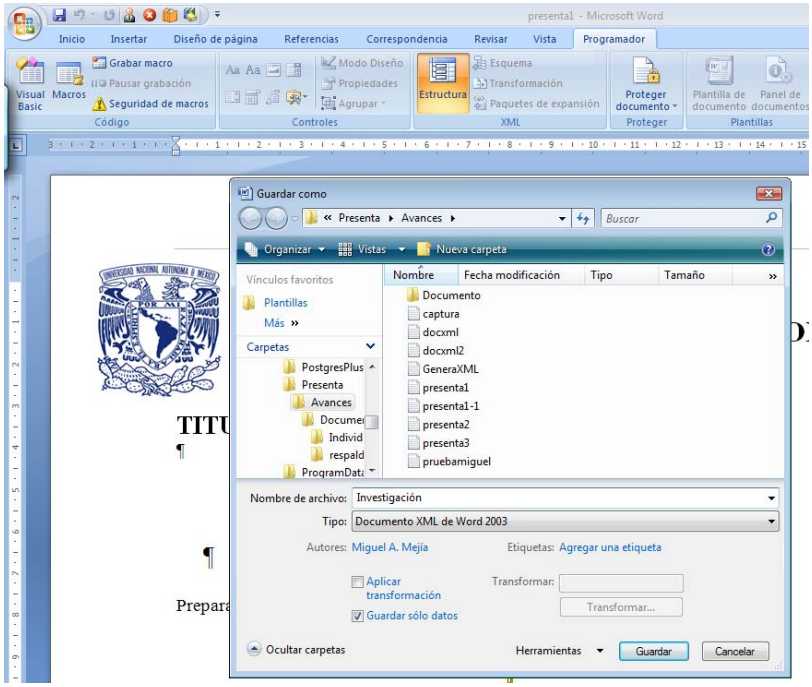


Fig. 4-6 Pantalla de Guardar como Documento XML de Word

Algunos de los problemas que se presentaron para generar la solución en Microsoft Word, incluyeron desde la falta del software necesario (Visual Studio Tools for Office), hasta la adecuación del esquema propuesto.

Sin embargo, uno de los problemas que requirió de mayor trabajo para su solución fue el establecer un método para incorporar elementos XML que no son obligatorios o que pueden ir desde un elemento hasta n, como son los autores o la bibliografía.

El documento Microsoft Word, tiene áreas protegidas que limitan la modificación o eliminación de las etiquetas del esquema XML y sólo permiten insertar nuevos elementos en los casos en que este permitido, para realizar esta función se colocaron botones de

comando en una barra de herramientas. La programación de los botones de comando tiene instrucciones de tipo XMLNodes, (diseñadas específicamente para el manejo de documentos XML), estas instrucciones deben estar ligadas a un esquema XML para validar el tipo y el número de elementos que pueden ser capturados.

Independientemente de las soluciones en Microsoft Word, se realizó otra aplicación, utilizando el sistema Adobe Designer 7.0 (fig. 4-8), para realizar una versión que permita generar documentos en formato PDF.

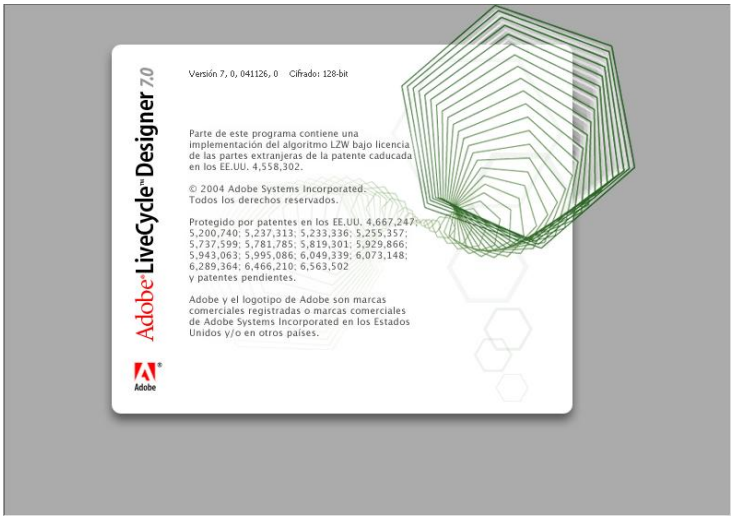


Fig. 4-8 Pantalla de Adobe Designer 7.0

El programa Adobe Designer, al igual que Visual Studio Tools for Office, facilita la inclusión de un esquema XML (fig. 4-9), para que la información sea validada al momento de su captura y de esta manera se puedan generar Documentos XML Bien formados y Válidos.

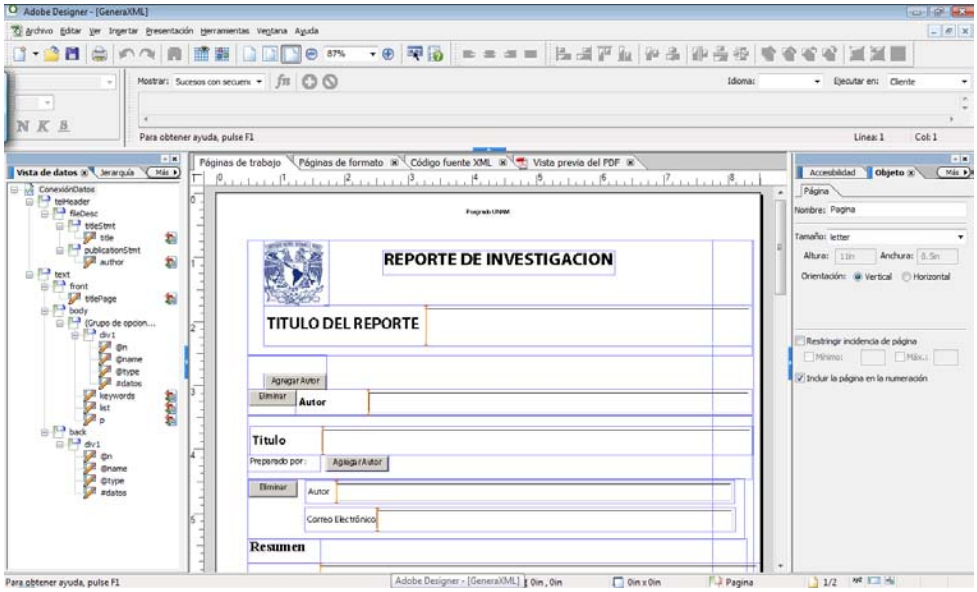


Fig. 4-9 Documento de Adobe Designer en desarrollo

Los documentos realizados en Adobe Designer, tienen el inconveniente de que los botones para inserción de nuevos elementos XML, se encuentran en el cuerpo del Reporte, lo cual no es muy conveniente ya que generan una vista que no es la que tendrán de manera definitiva los Reportes en el momento de su impresión.

Una vez generados los Documentos XML, es necesario su almacenamiento en una Base de Datos que facilite su administración, para ello, se creó una Base de datos en eXist, dicho manejador posee una interfaz de usuario y de administrador que permite manipular los documentos XML de manera fácil y amigable.

El SMBD eXist es una Base de Datos XML Nativa, lo cual provee de mejores herramientas para el manejo y explotación de documentos XML (fig. 4-10).

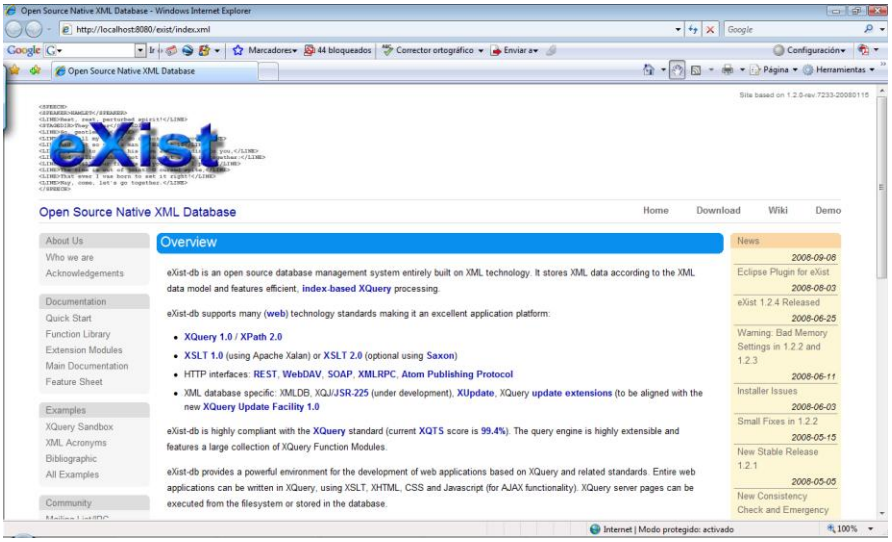


Fig. 4-10 Pantalla de inicio de eXist

La Base de Datos diseñada, se encuentra organizada en colecciones por rama o materia de investigación, lo que facilita su administración (fig. 4-11).

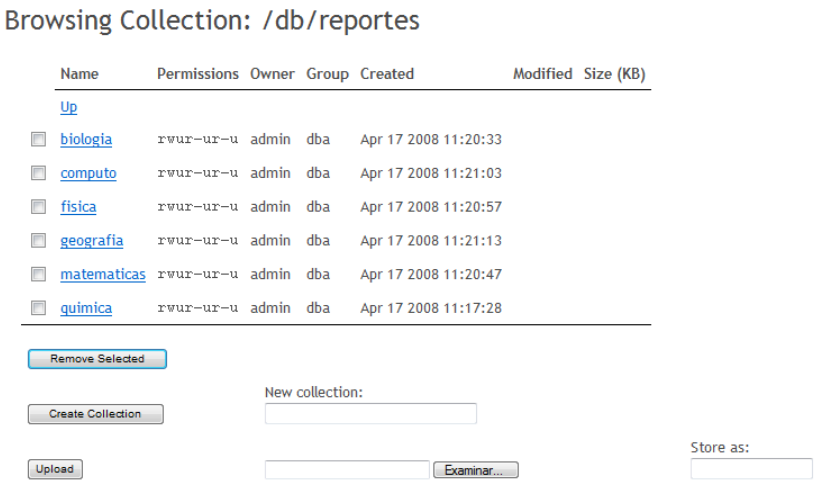


Fig. 4-11 Colecciones de la Base de Datos

Para almacenar los Documentos XML en una Base se Datos XML Habilitada, se desarrollo una base de datos en el SMBD Postgres (Fig. 4-12).

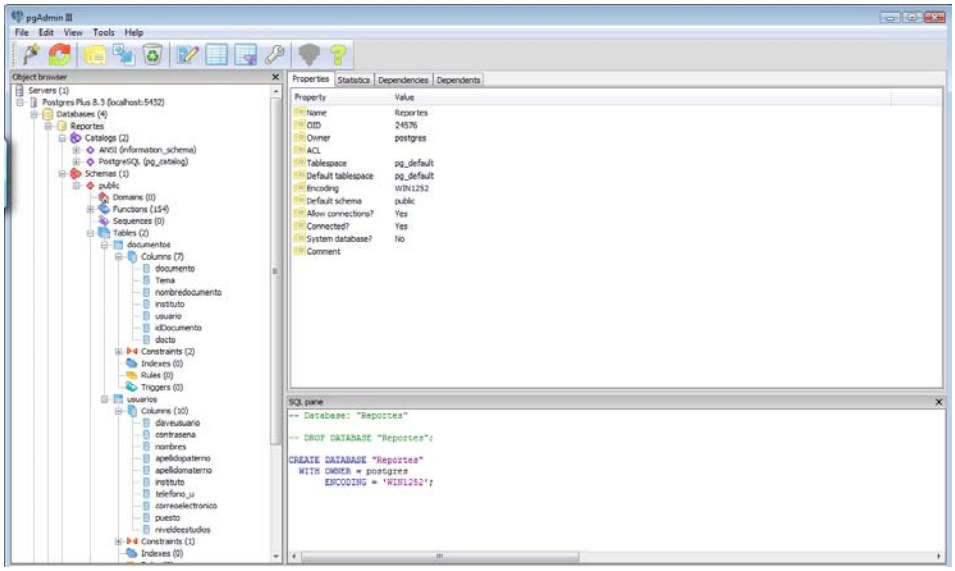


Fig. 4-12 Pantalla de PostgreSQL

Como se ha mencionado anteriormente, el utilizar bases de datos XML habilitadas, incrementa la dificultad para manipular documentos XML, y con ello aumenta el trabajo del desarrollador.

La interfaz se desarrolló (fig.4-13), utilizando una de las herramientas de programación que actualmente esta cobrando mucha relevancia, como lo es el uso de Patrones de Diseño, en este caso Modelo Vista Controlador (MVC).

Fig. 4-13 Sistema de Reportes XML

El diseño de pantallas se realizó para que sea amigable con el usuario y el administrador y de ésta forma se invierta poco tiempo en la capacitación de los usuarios. Como se muestra en la figura 4-14.

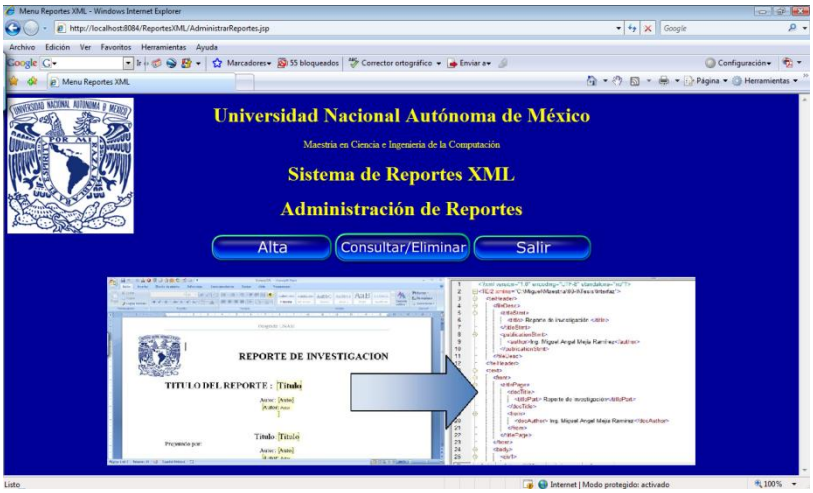


Fig. 4-14 Menú Reportes XML

Para facilitar el almacenamiento de los Reportes de investigación, la pantalla de captura solicita algunos datos y la ruta donde se encuentra el archivo que contiene la información del Reporte de investigación en formato XML (Fig. 4-15).

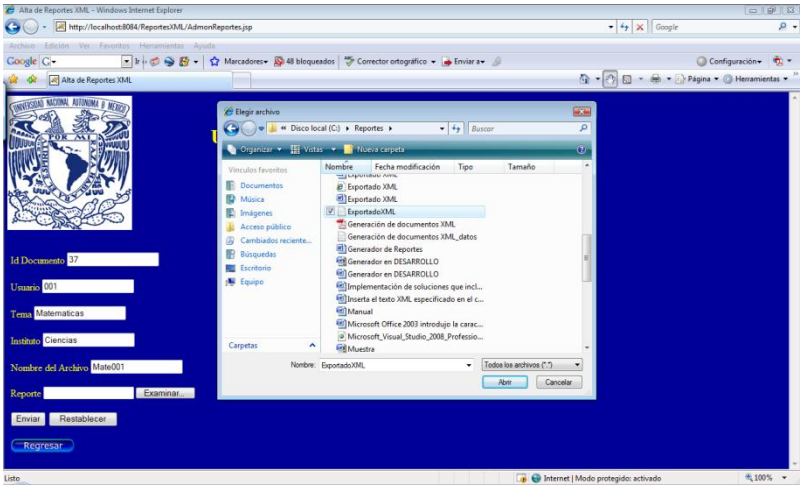


Fig. 4-15 Captura de Reportes XML

La pantalla de administración de usuarios se muestra en la figura 4-16.

Fig. 4-16 Administrar Usuarios (pantalla del Administrador)

Según el patrón de diseño MVC (fig. 4-17), la capa intermedia de una aplicación Web puede ser dividida en tres partes funcionales:

- Controlador
- Vista
- Modelo

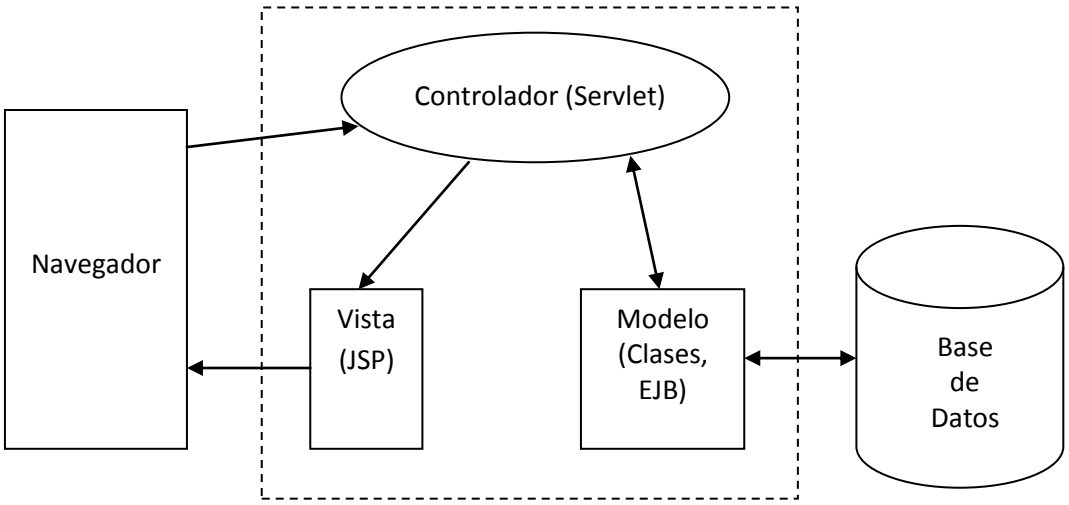


Fig. 4-17 Esquema de una aplicación MVC

Controlador

Todas las peticiones a la capa intermedia que se realicen desde el cliente son dirigidas al Controlador, cuya misión es determinar las acciones a realizar para cada una de estas peticiones e invocar al resto de los componentes de la aplicación (Modelo y Vista) para que realicen las acciones requeridas en cada caso, encargándose también de la coordinación de todo el proceso.

Vista

La Vista es la encargada de generar las respuestas, que deben ser enviadas al cliente. Cuando esta respuesta tiene que incluir datos proporcionados por el Controlador, el código de la página no será fijo si no que deberá ser generado de forma dinámica, por lo que su implementación correrá a cargo de una página JSP.

Modelo

En la arquitectura MVC la lógica de negocio de la aplicación, incluyendo el acceso a los datos y su manipulación, está encapsulada dentro del Modelo, el cual esta formado por una serie de componentes de negocio independientes del Controlador y la Vista, permitiendo así su reutilización y el desacoplamiento entre las capas.

Los problemas que se presentaron para realizar la aplicación, fueron principalmente la captura de los documentos XML en los campos establecidos para ello, de tal manera que el usuario sólo indicara la ruta del archivo que contenía el documento XML, para solucionar este problema se realizó la programación de una rutina para lectura de archivos, línea por línea, asignando esta lectura a una variable de tipo StringBuffer, una vez alcanzado el final del archivo esta variable se asigna al campo de tipo XML.

Una vez que se guardó el archivo se observó que existía el problema de que no aceptaba caracteres acentuados, ni letras ñ o Ñ, por lo que se tuvo que realizar una modificación en la rutina de lectura de archivos, insertando la siguiente línea de código:

```
String UTF8Str = new String (line.getBytes(),"UTF-8" );
```

Con ésta línea, se lee un conjunto de bytes con formato UTF-8, que reconoce los caracteres acentuados y las ñ o Ñ.

4.2 Pruebas

Para esta etapa, se consiguieron algunos reportes de investigación (impresos) y otros en documentos XML, para tener información de captura y medios de comparación con los nuevos documentos XML generados a través de la interfaz diseñada.

Se realizaron algunas pruebas al esquema XML, para verificar que éste se adaptara a las necesidades y pudiera servir de base para la generación de otros documentos XML.

También se realizaron pruebas en la generación de documentos XML mediante el uso de las herramientas diseñadas para Microsoft Word 2007 y Microsoft Word 2003, así como en Adobe Acrobat, verificando que los documentos resultantes estén de acuerdo al esquema antes mencionado.

Una vez generados los documentos XML, se procedió a cargarlos en las bases de datos eXist y Postgres, verificando que las estructuras de dichas bases facilitaran el manejo de la información almacenada.

Una vez realizadas las pruebas en la etapa de desarrollo, se procedió a instalar la aplicación (a mediados del mes de noviembre de 2008) para que usuarios del Instituto de Ciencias de la Atmosfera lo utilizaran para observar su desempeño.

Sólo se solicitó una corrección, la cual consistió en la reprogramación de un botón de comando.

El detalle de las pruebas se muestra en el apéndice E “Diseño de pruebas realizadas”.

4.3 Ajustes finales

De acuerdo al resultado de las primeras pruebas, fue necesario realizar pequeños ajustes a la estructura del esquema XML, y con ello también hacer algunas modificaciones en el formato de los documentos de Microsoft Word y Adobe Acrobat, así como la incorporación de nuevas funciones que agregan otros elementos del esquema.

Para la interfaz de las bases de datos sólo fue necesario realizar modificaciones para la lectura de acentos, ñ y Ñ.

4.4 Pruebas finales

Una vez realizados los ajustes finales, se procedió nuevamente a realizar pruebas en el sistema, tanto en la generación de documentos XML, como en el almacenamiento en las bases de datos, verificando la correcta captura y el almacenamiento de los documentos con todos y caracteres acentuados, dichas pruebas también se detallan en el apéndice E.

Conclusiones

Si se desea intercambiar información en forma segura y precisa, sin tener problemas, es de vital importancia que la información tenga un formato estándar, para ello, el uso de esquemas XML representa grandes ventajas.

Actualmente existe un gran número de esquemas XML para la generación de documentos, cada uno de ellos ofrece ciertas ventajas, de acuerdo al tipo de documento final que se desea obtener. Después de un análisis detallado, se decidió utilizar el esquema TEI2, el cual define todos los elementos necesarios para poder almacenar los Reportes de Investigación que se realizan en el Instituto de Ciencias de la Atmosfera (lugar donde se está utilizando con éxito este trabajo).

Aprovechando el conocimiento sobre la herramienta Microsoft Word que posee el personal del Instituto de Ciencias de la Atmosfera, se desarrolló una interfaz para facilitar la captura de información de las investigaciones realizadas en este Instituto, incorporando el esquema TEI2 en un documento aparentemente común de Microsoft Word 2007 (también se realizó una versión para Word 2003) y se programó una barra de herramientas que permite la inserción o eliminación (según sea el caso) de elementos definidos en el esquema TEI2 que representan secciones en un Reporte de investigación.

Una vez capturada la información en el archivo de Microsoft Word, éste puede ser guardado como un documento normal de Word, lo que facilitará su posterior edición y/o impresión; o generar de una manera relativamente sencilla documentos XML, para su incorporación en una base de datos.

De manera similar, se realizó un documento en Adobe Designer 7.0 para que se pueda utilizar la interfaz de Adobe Acrobat para la captura de información de los Reportes de Investigación y almacenarlo en formato PDF y como documento XML.

Independientemente de la herramienta utilizada (Microsoft Word o Adobe Acrobat), para generar los documentos XML, éstos se pueden almacenar en una base de datos, y de esta manera estar en posibilidad de compartir la información con otros institutos u organizaciones mediante la asignación de claves de acceso, de igual manera se puede asegurar la integridad de la información conservando los estándares establecidos, y en su caso exportar los documentos XML para que puedan ser utilizados de manera independiente.

Analizando diversas bases de datos tanto nativas como habilitadas se decidió utilizar una de cada tipo, para poder comparar las ventajas y desventajas de dichos modelos de bases de datos.

Para tal efecto, se diseñaron dos bases de datos; una en el Sistema Manejador de Bases de Datos (SMBD) eXist y otra en el SMBD PostgreSQL, ambas bases permiten almacenar los Reportes de investigación (en forma de documentos XML).

Al usar el SMBD eXist se tienen todas las ventajas de una base de datos nativa de XML, aunado a esto, se observó que la estructura de catálogos que utiliza eXist facilita de sobremanera la administración de los documentos XML, también contiene una interfaz para realizar consultas utilizando el lenguaje XQuery, así como, un visor de documentos XML. Una de las desventajas que presenta eXist, es que al ser un proyecto relativamente nuevo, tiene poca difusión.

Al utilizar el SMBD PostgreSQL, que es una base de datos habilitada para XML, se tienen las ventajas de un modelo relacional y un tipo especial de datos “xml”, el cual permite almacenar un documento de tipo XML. Sin embargo una gran desventaja que presentó PostgreSQL es que fue necesario diseñar un sistema para administrar los reportes de investigación. Lo que no fue obligado en eXist.

Al comparar las ventajas y desventajas que proporcionan eXist y PostgreSQL, se observó que las bases de datos nativas representan una mejor opción para el almacenamiento y recuperación de documentos XML, ya que proporcionan herramientas superiores para su organización y para la explotación de la información contenida haciendo uso de toda la tecnología XML (XQuery, XPath, XSLT, etc.), sin la necesidad de desarrollar aplicaciones especiales.

Trabajos Futuros

Si bien, en las interfaces de Microsoft Word o Adobe Acrobat, se utiliza el esquema TEI2, para una futura versión se podría pensar en realizar la incorporación de diferentes esquemas XML, lo que permitiría generar diversos tipos de documentos XML, incrementando la posibilidad de utilizar esta herramienta en un mayor número de usuarios.

Al tener almacenados los reportes de investigación en bases de datos, se podría desarrollar una herramienta que incorpore técnicas avanzadas de recuperación de información, por ejemplo la consulta basada en palabras clave y la consulta con procesamiento de lenguaje natural.

Bibliografía

Referencias bibliográficas

1. Benoit Marchal, *XML con ejemplos*, Prentice Hall, 2004.
2. Adolfo Vázquez, *XML*, Alfaomega, 2002.
3. Akmal B. Chaudhri, Awais Rashid and Roberto Zicari. *XML Data Management. Native XML and XML-enabled Database Systems.* Addison-Wesley. 2003.
4. Graves, Mark. *Designing XML Databases.* Prentice Hall. 2002.
5. Maruyama, Hiroshi and Tamura, Kent and Uramoto, Naohiko. *XML and Java. Developing Web Applications*. Second Edition. Addison-Wesley. 2002.
6. Gavin Powell. *Beginning XML Databases.* Wrox Press Ltd. 2007.
7. Leung Theodore W., *Professional XML Development with Apache Tools: Xerces, Xalan, FOP, Cocoon, Axis, Xindice*, Wrox Press Ltd. 2004.
8. Ray T. Erik, *Learning XML*, O'Reilly & Associates 2001
9. Stelting S., Maassen O., *Applied Java Patterns*, Sun Microsystems Press, 2002.
10. Fowler M. *Analysis Patterns. Reusable Object Models*. Addison Wesley, 1997.
11. Charles f. Goldfarb & Paul Prescod, *Manual de XML*, Prentice Hall, 2005.
12. Michael J. Young, *Aprenda XML ya*, Mc Graw Hill-Microsoft , 2001.

Sitios Web relacionados:

1. American Psychological Association **APA**
   Http://www.apa.org/journals/webref.html
   Http://owl.english.purdue.edu/Files/34.html
2. Dublin Core Metadata **DC**
   Http://purl.org/DC/
3. Digital Object Identifier System **DOI**
   Http://www.doi.org/
4. Encoded Archival Description **EAD**
   Http://lcweb.loc.gov/ead/
5. Content Standards for Digital Geospatial Metadata **FGDC**
   Http://www.fgdc.gov/Metadata
6. Government Information Locator Service **GILS**
   Http://www.gils.net
7. Handle System **HS**
   Http://www.handle.net
8. ISO 690-2 **ISO**
   Http://www.nlc-bnc.ca/iso/tc46sc9/standard/690-2e.htm

Sitios Web relacionados:

9. Modern Language Association **MLA**
   Http://www.mla.org/publications/stylemanual_index.htm
10. Platform for Privacy Preferences **P3P**
    Http://www.w3.org/P3P/
11. Platform for Internet Content Selection **PICS**
    Http://www.w3.org/PICS/
12. Publisher Item Identifier **PII**
    Http://www.elsevier.nl/inca/homepage/about/pii/
13. Persistent Uniform Resource Locator **PURL**
    Http://purl.org/
14. Resource Description Framework **RDF**
    Http://www.w3.org/RDF
15. Serial Item and Contribution Identifier **SICI**
    Http://sunsite.berkeley.edu/SICI/
16. Text Encoding Initiative **TEI**
    Http://www-tei.uic.edu/orgs/tei/
    Http://etext.virginia.edu/TEI.html
17. Uniform Resource Identifiers Working Group **URI**
    Http://www.ics.uci.edu/pub/ietf/uri/
18. Uniform Resource Name Working Group **URN**
    Http://www.ietf.org/html.charters/urn-charter.html
19. eXtensible Markup Language **XML**
    Http://www.w3.org/XML

Apéndice A

Manual del usuario

MANUAL DEL USUARIO

Sistema para generar documentos XML

Versión para Microsoft Word 2003

Marzo 2009

Introducción

El presente sistema tiene por objeto proporcionar una herramienta que genere documentos XML, con un esquema predefinido para su posterior incorporación en una base de datos.

Requerimientos básicos

Se deberá contar con Microsoft Word 2003 o posterior

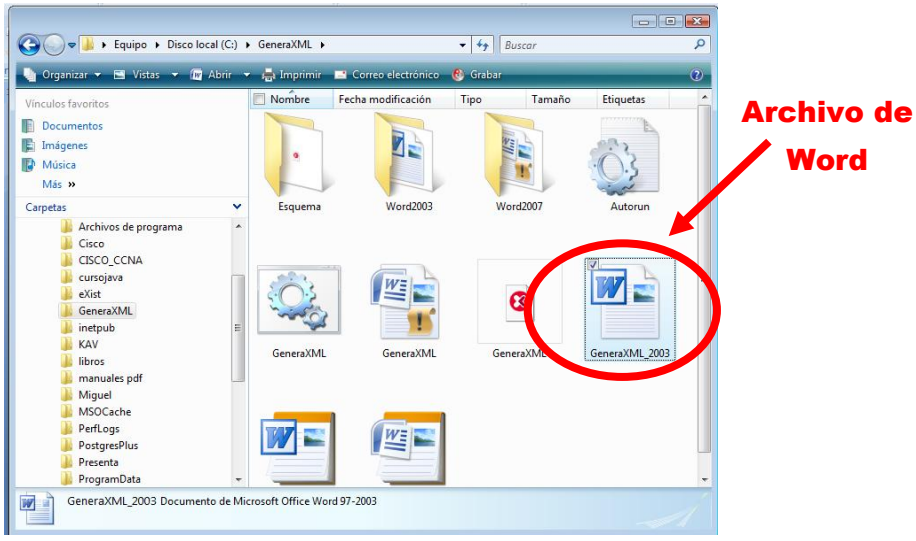
Espacio en disco duro de 500 KB

Instalación

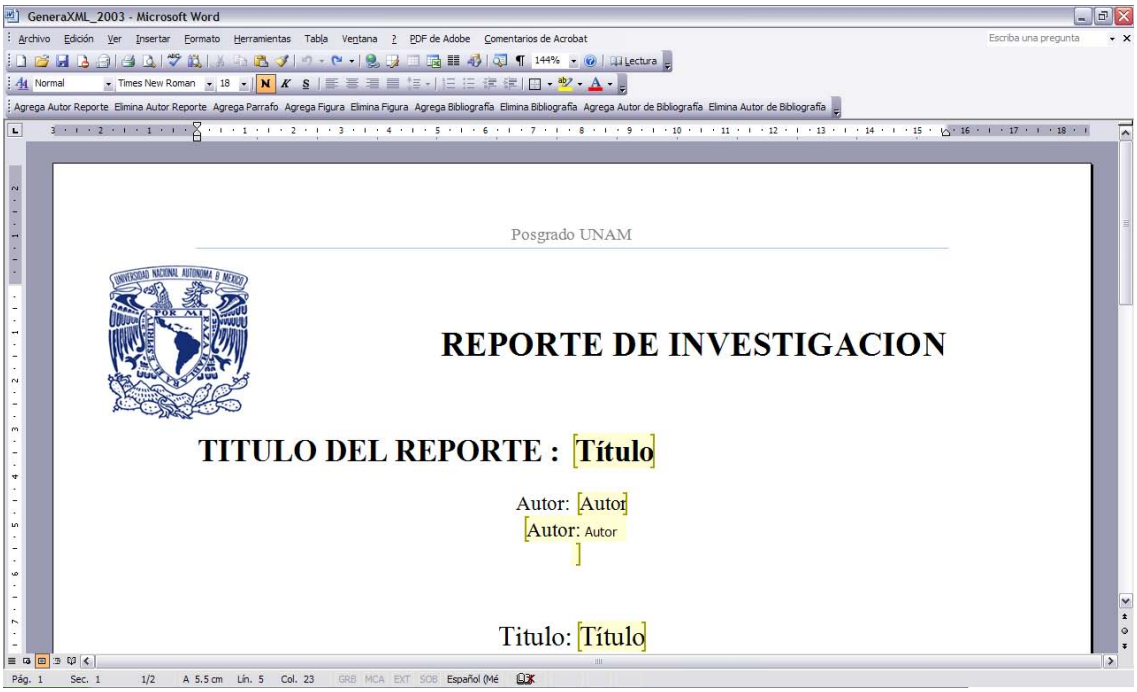
El disco contiene un programa de instalación que genera la carpeta **GeneraXML** y copia los archivos necesarios para su ejecución, en caso de tener deshabilitada la opción de Autorun, solo es necesario copiar la carpeta **GeneraXML** incluida en el disco, a un lugar de su disco duro.

Aplicación

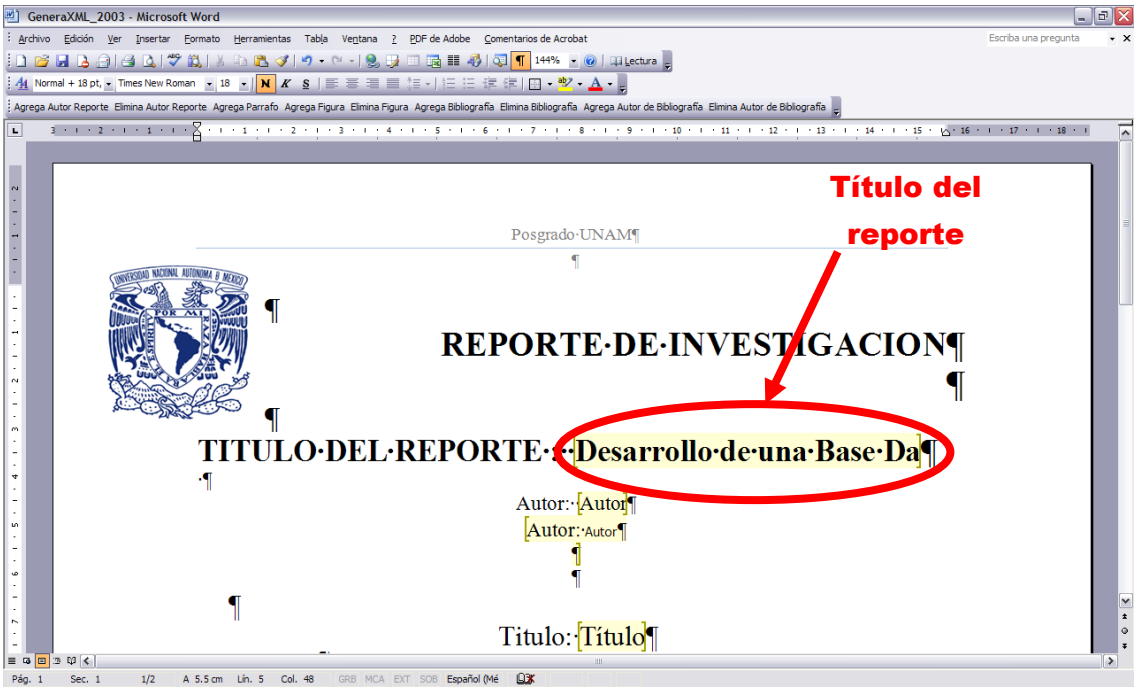
Para comenzar a utilizar la aplicación, sólo se tiene que abrir el archivo de Microsoft Word "GeneraXML_2003".



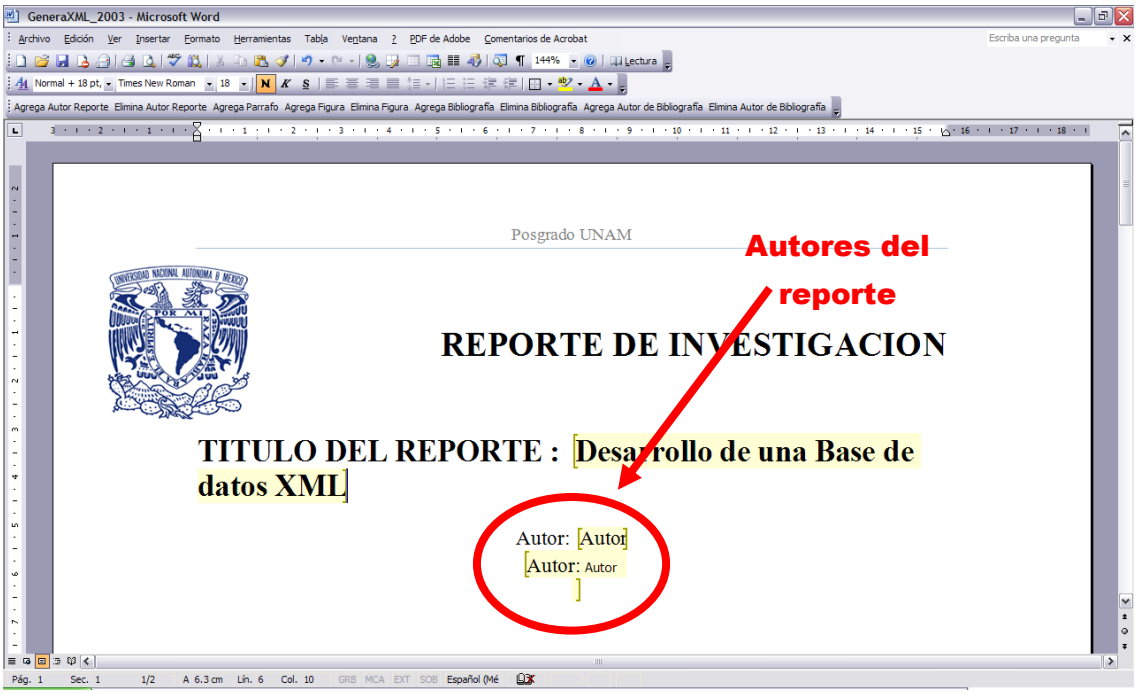
Una vez abierto el archivo se presenta la siguiente pantalla



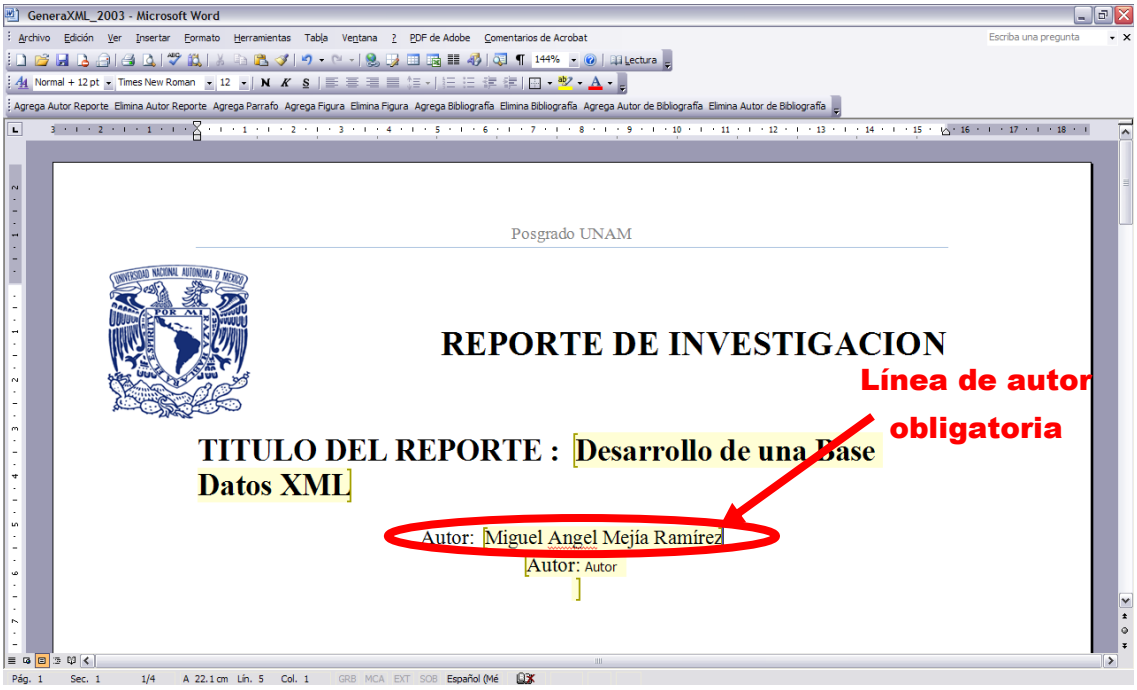
La primera información que se debe capturar es el título del Reporte



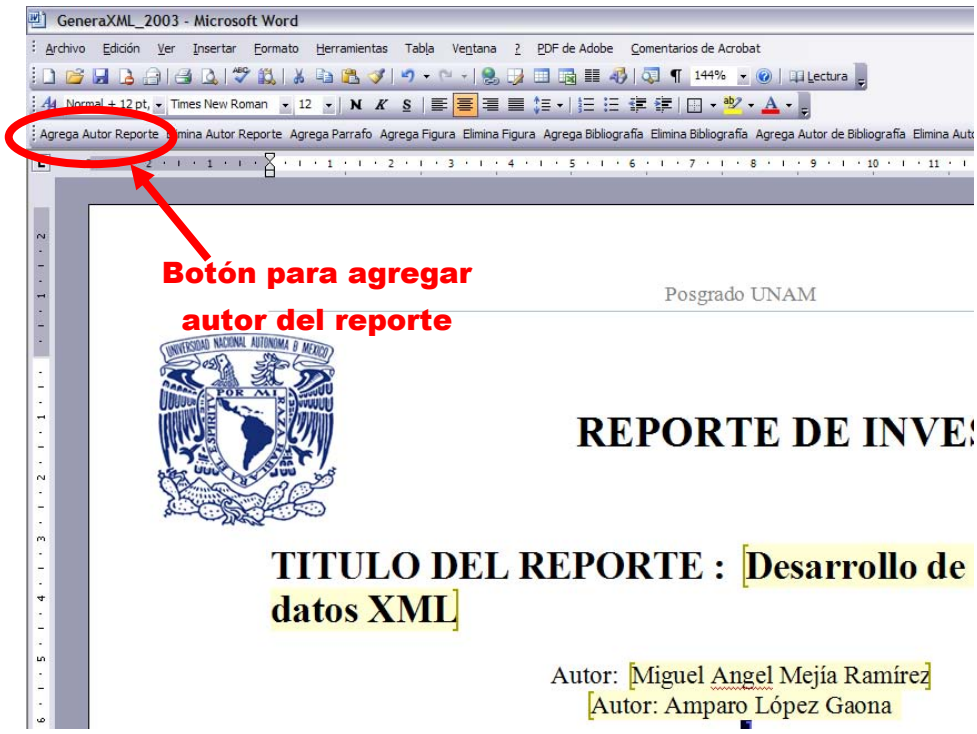
Después deberá capturar el nombre del o los autores del reporte



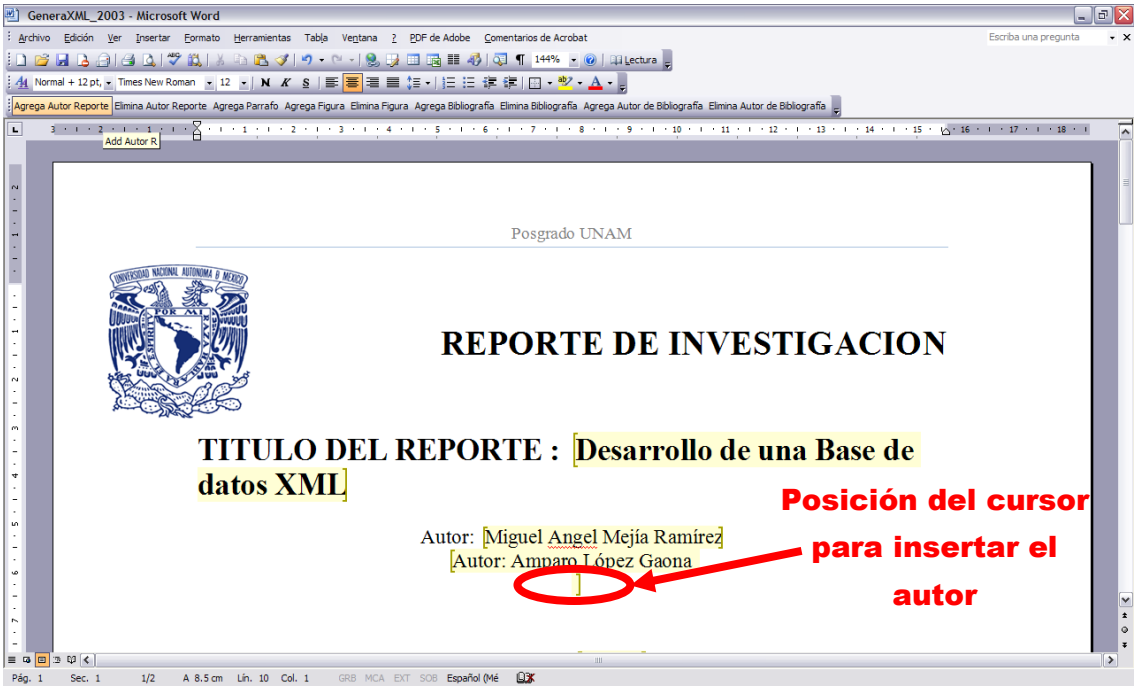
Es importante mencionar que por lo menos debe existir un autor del reporte, por lo que la primera línea de Autor **no** puede ser eliminada.



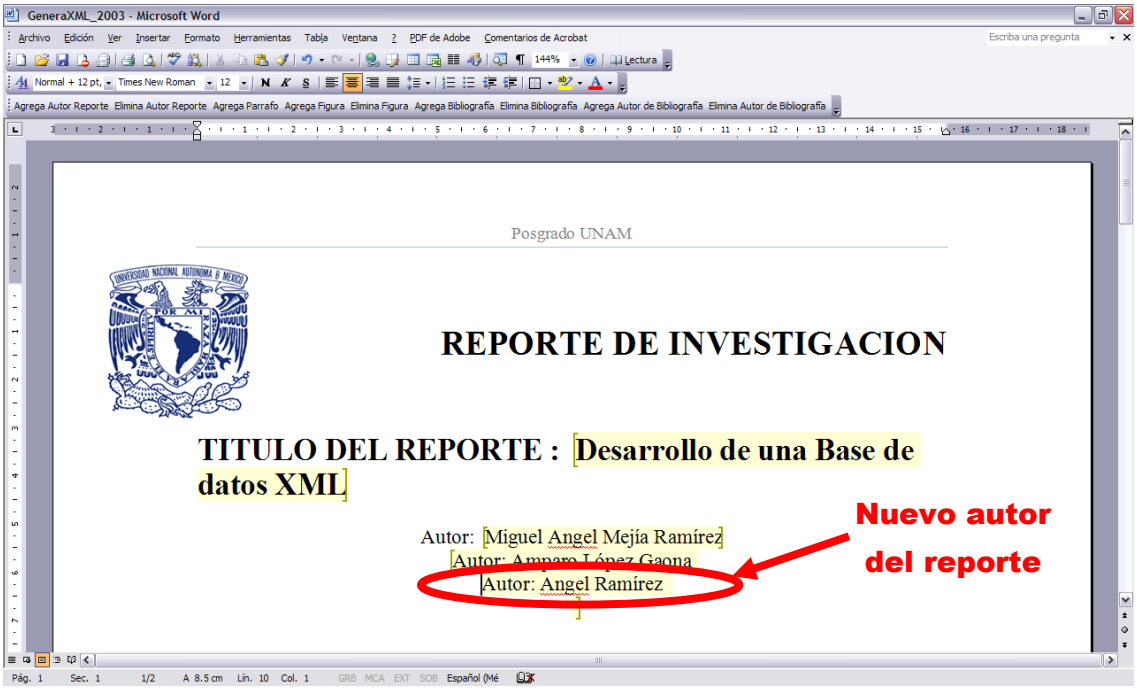
El reporte tiene espacio para dos autores, pero en caso de necesitar más autores podrá utilizar el botón "**Agrega Autor Reporte**", colocado en la barra de herramientas GeneraXML.



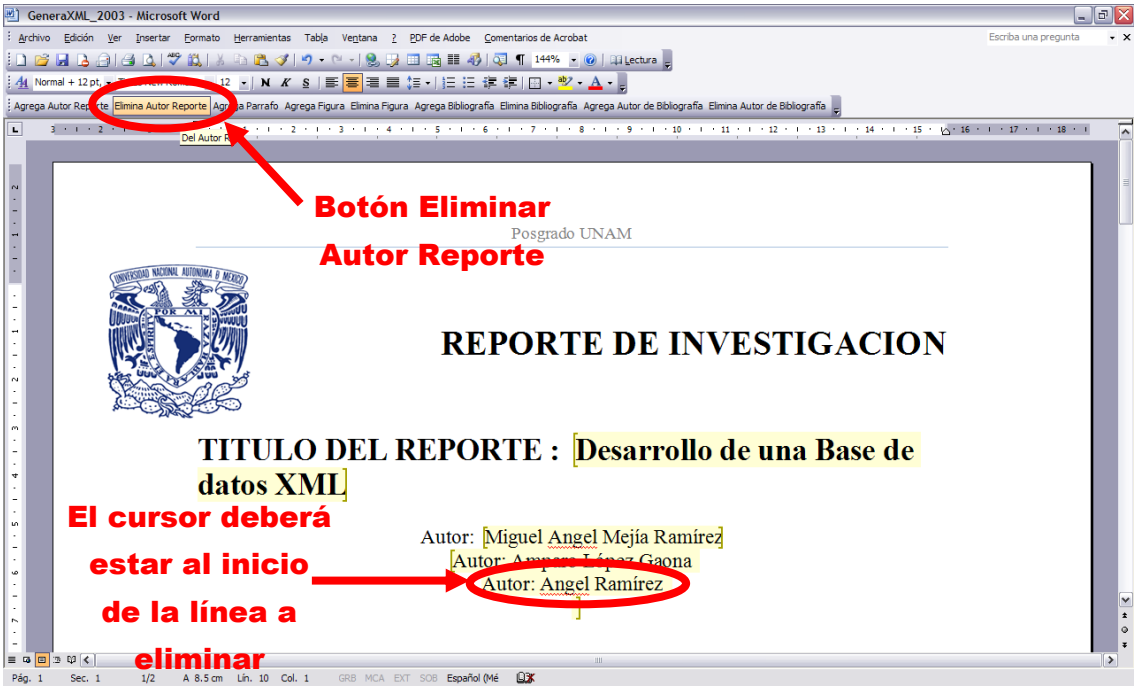
Es necesario que coloque el cursor en el lugar donde desea insertar el nuevo autor del reporte



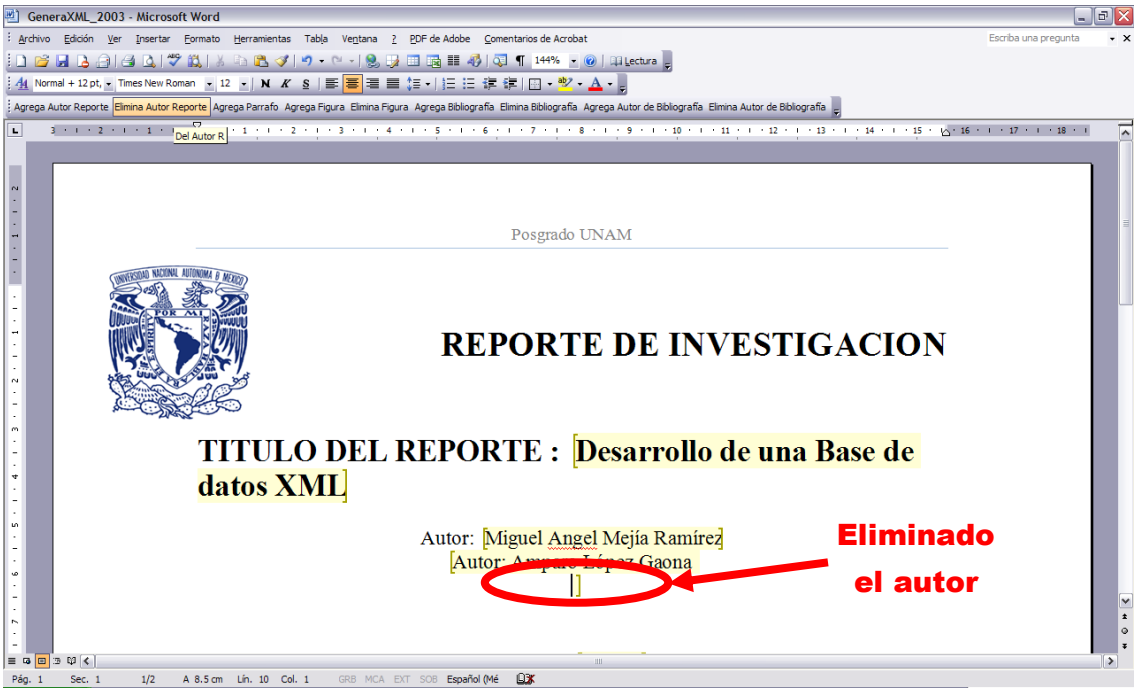
El sistema insertará la etiqueta “Autor” y el espacio para que el nombre del autor pueda ser capturado



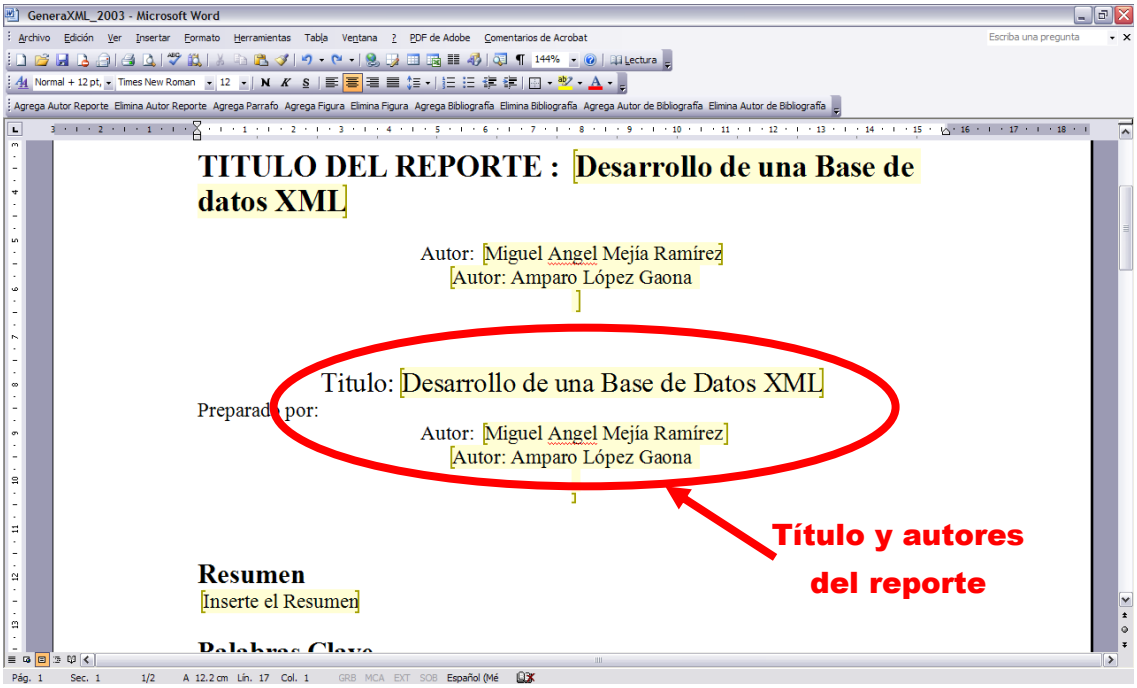
En caso de que sea necesario eliminar uno o varios autores, deberá utilizar el botón "**Eliminar Autor Reporte**", para lo cual el cursor deberá estar al inicio de la línea del autor a eliminar.



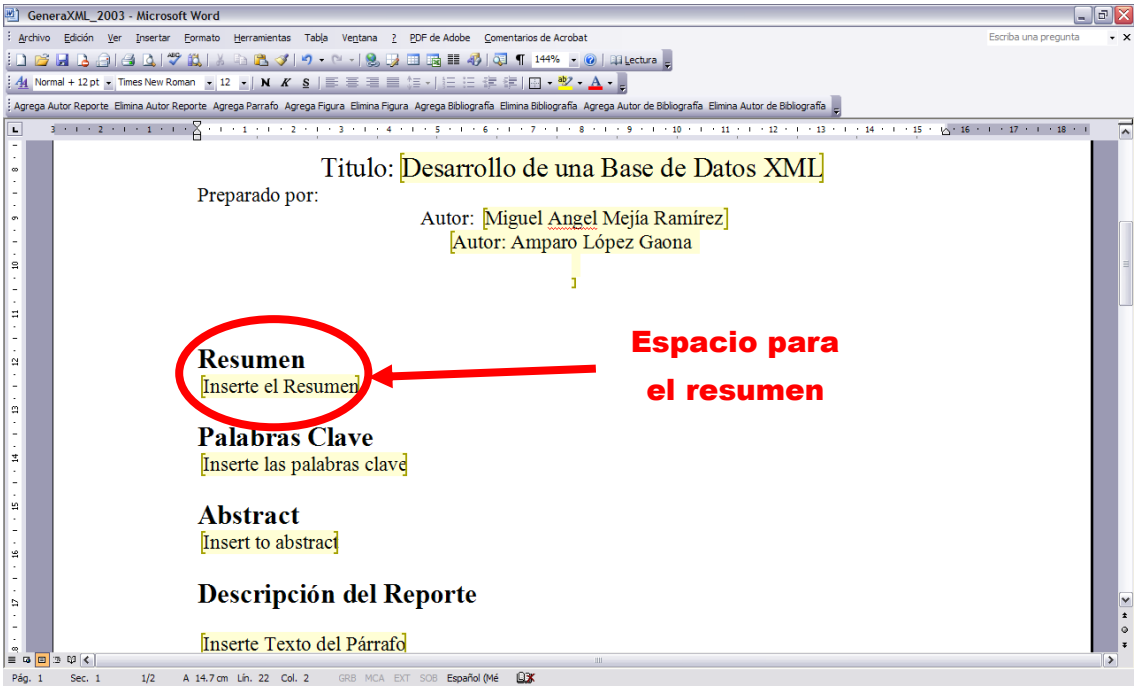
Una vez utilizado el botón "Eliminar Autor Reporte", se eliminará la etiqueta "Autor" y el contenido o nombre del autor.



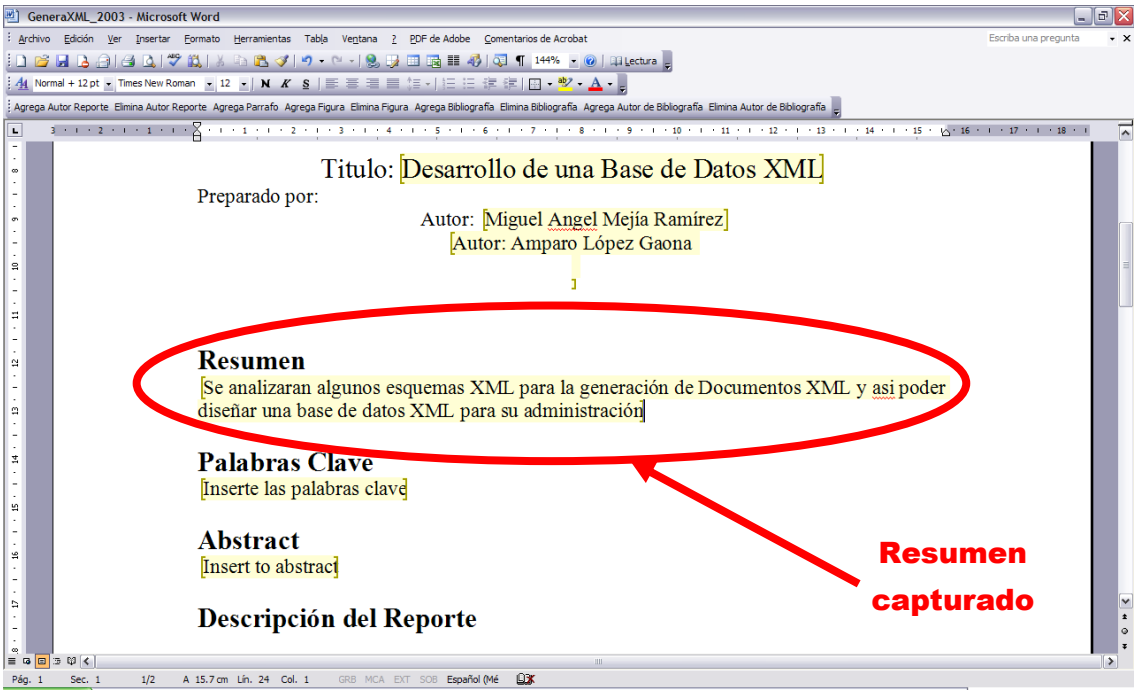
Para que el documento se adapte al esquema XML, es necesario capturar nuevamente el título del reporte y el o los autores; recordando que se tienen las mismas opciones para los autores.



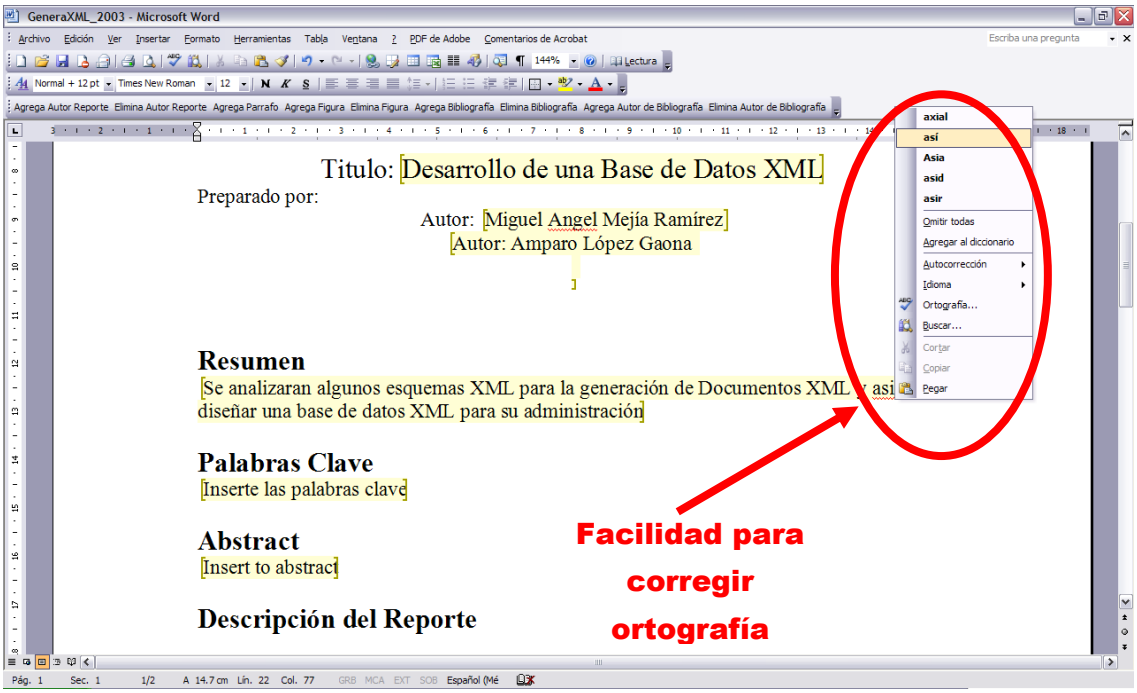
Enseguida deberá capturar un resumen del reporte de investigación



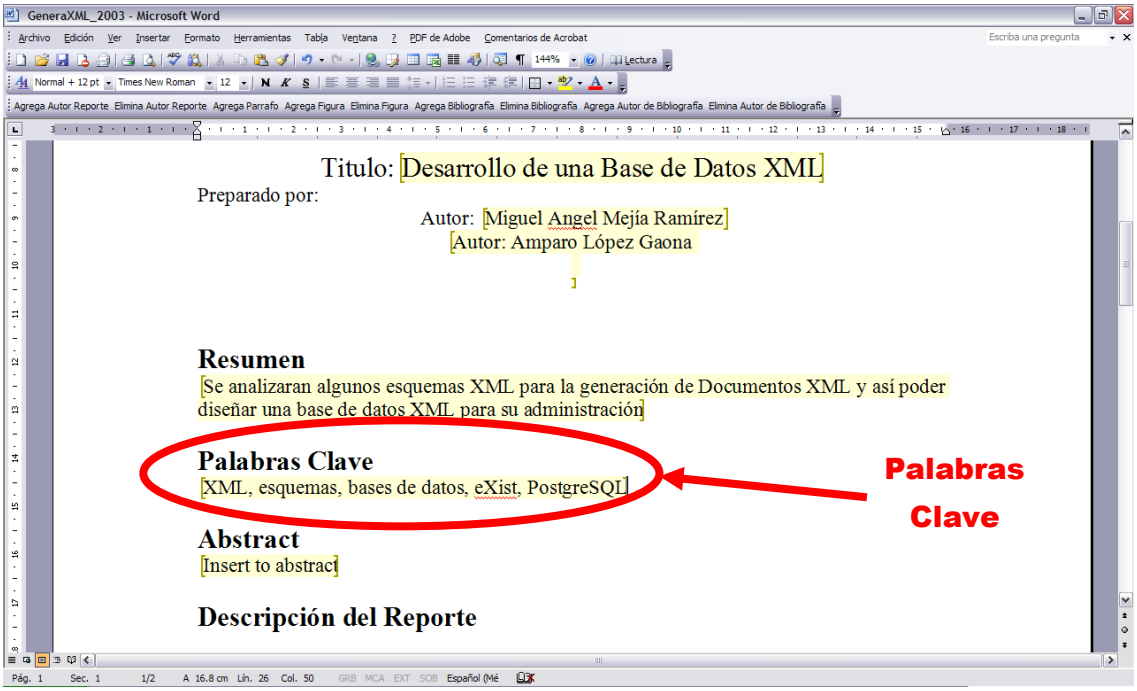
Es importante mencionar que al utilizar esta aplicación en Microsoft Word, se tienen todas las ventajas del procesador de textos, como es el uso letras **bold**, *itálica*, etc.



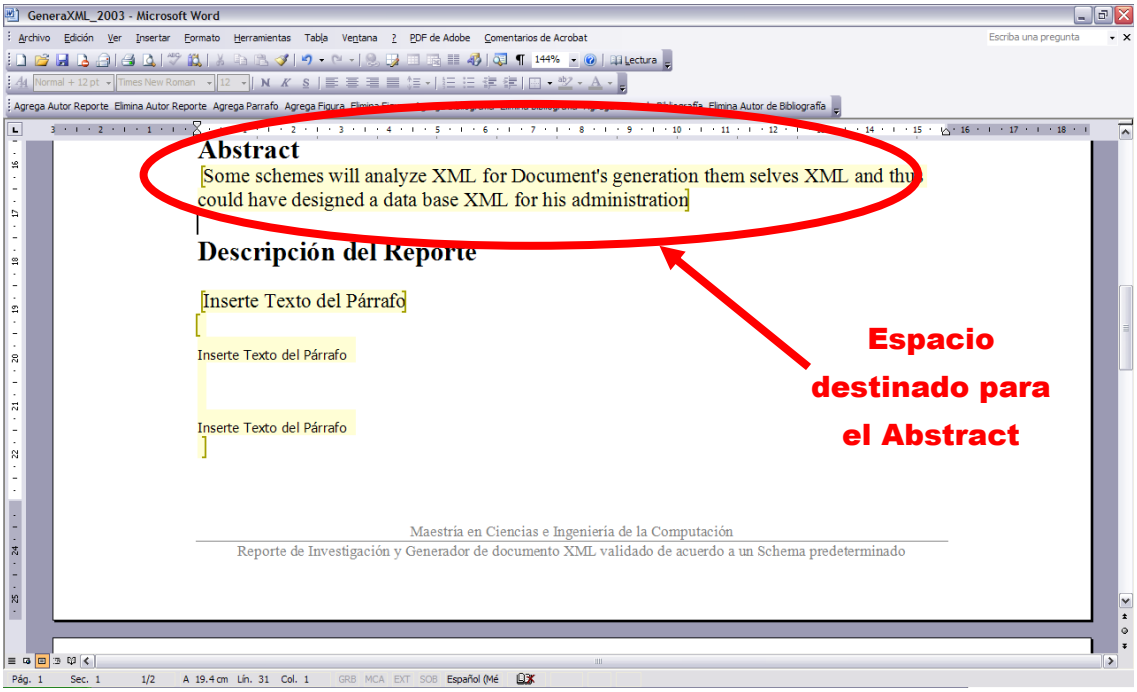
De igual manera se puede hacer uso de las facilidades para la verificación y corrección ortográfica.



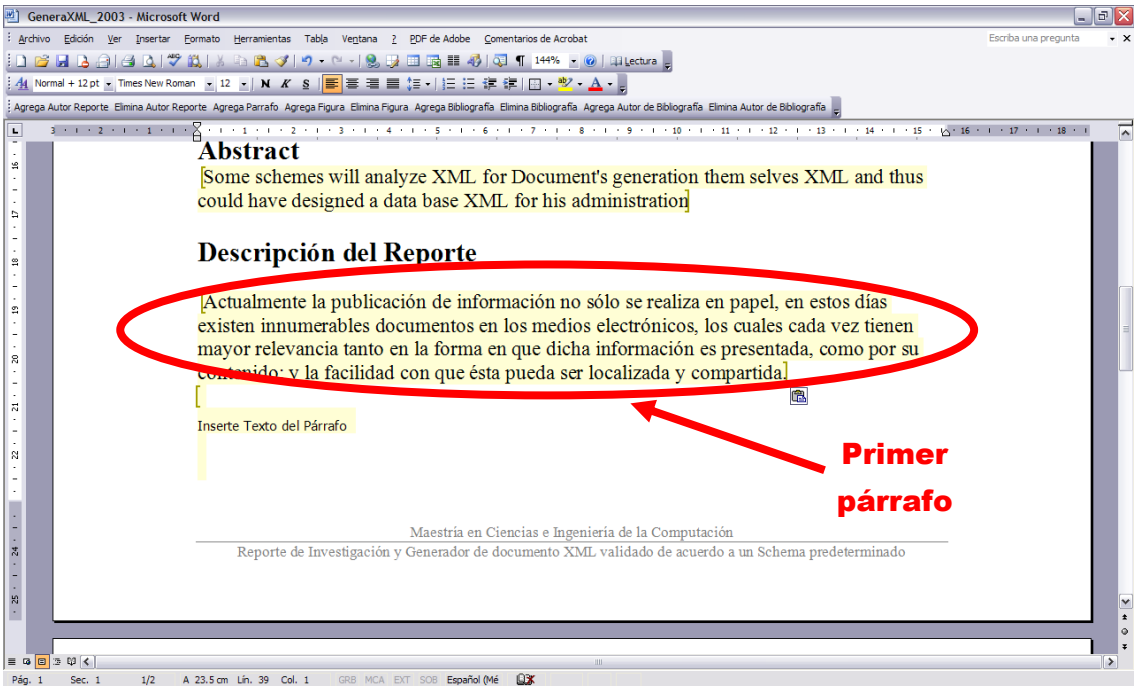
Continuando con la captura de información del reporte, lo siguiente es insertar las "**Palabras Clave**", que también son consideradas un campo obligatorio, por lo tanto tampoco puede ser eliminado.



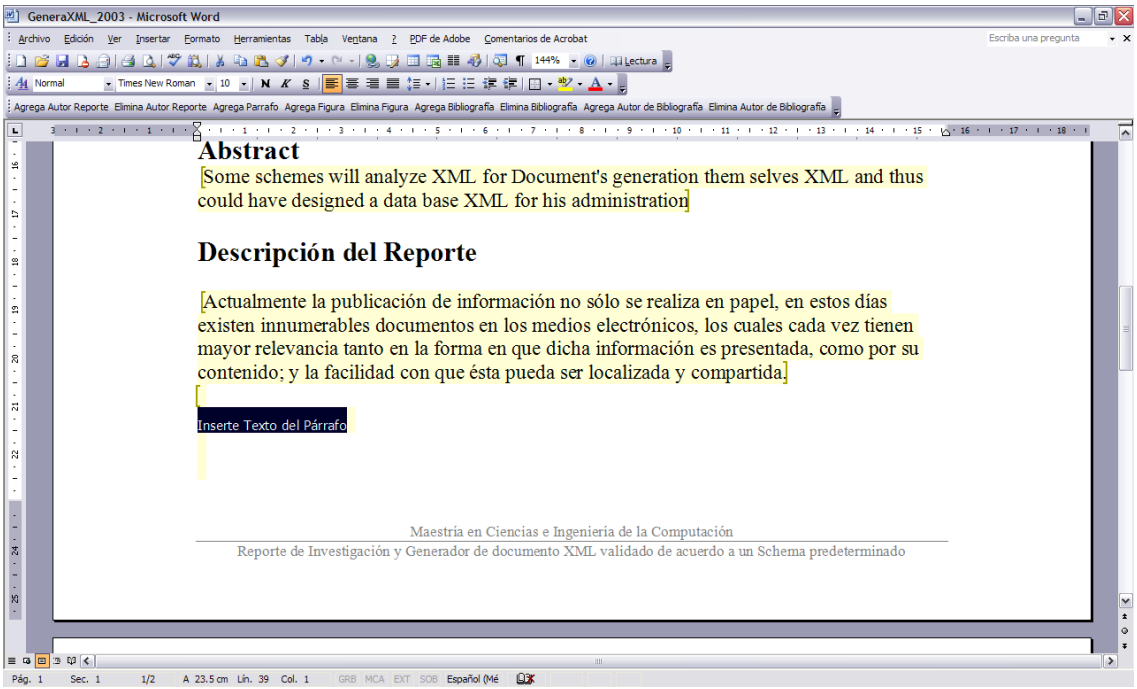
De igual manera se capturará la información correspondiente al "**Abstract**", esta información deberá ser en inglés.



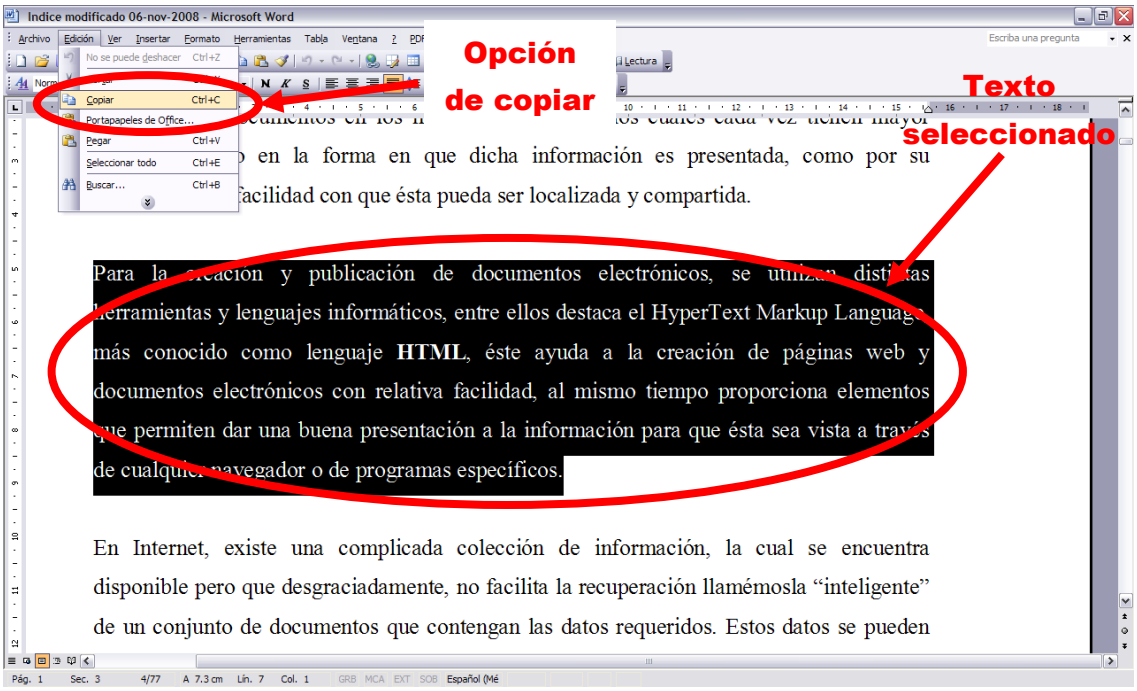
Lo siguiente en capturar es la "**Descripción del Reporte**", la cual esta compuesta de párrafos que deberán ser insertados de acuerdo a la necesidad del usuario.



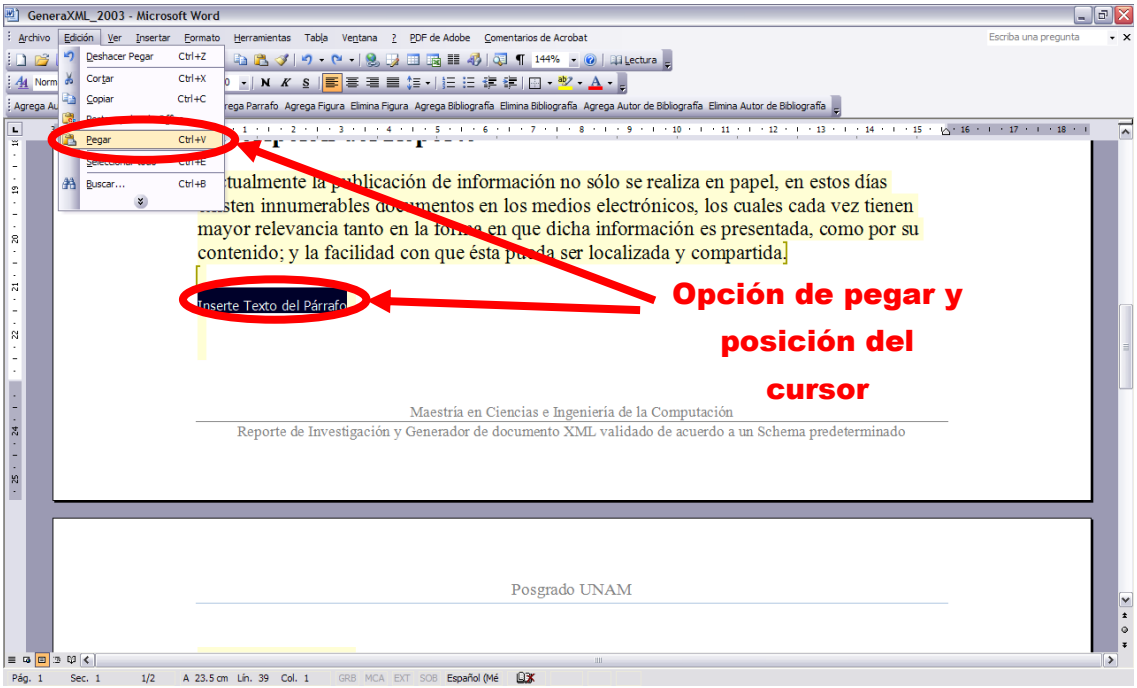
Es importante mencionar que el primer párrafo también es un campo obligatorio, por lo tanto no puede ser eliminado. El o los siguientes párrafos son opcionales.



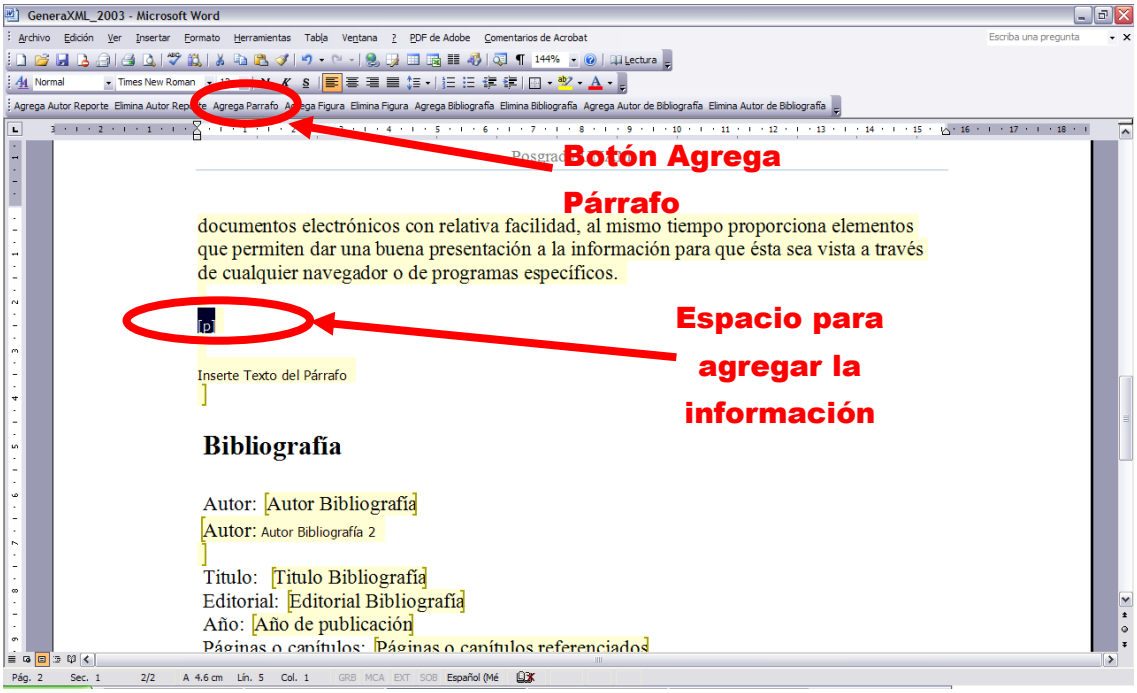
Si la información se tiene en otros archivos, se podrá utilizar la opción de copiar y después pegar, de ésta manera es posible ahorrar trabajo y tiempo en la captura de datos.



Pegado del texto seleccionado en un espacio de párrafo.

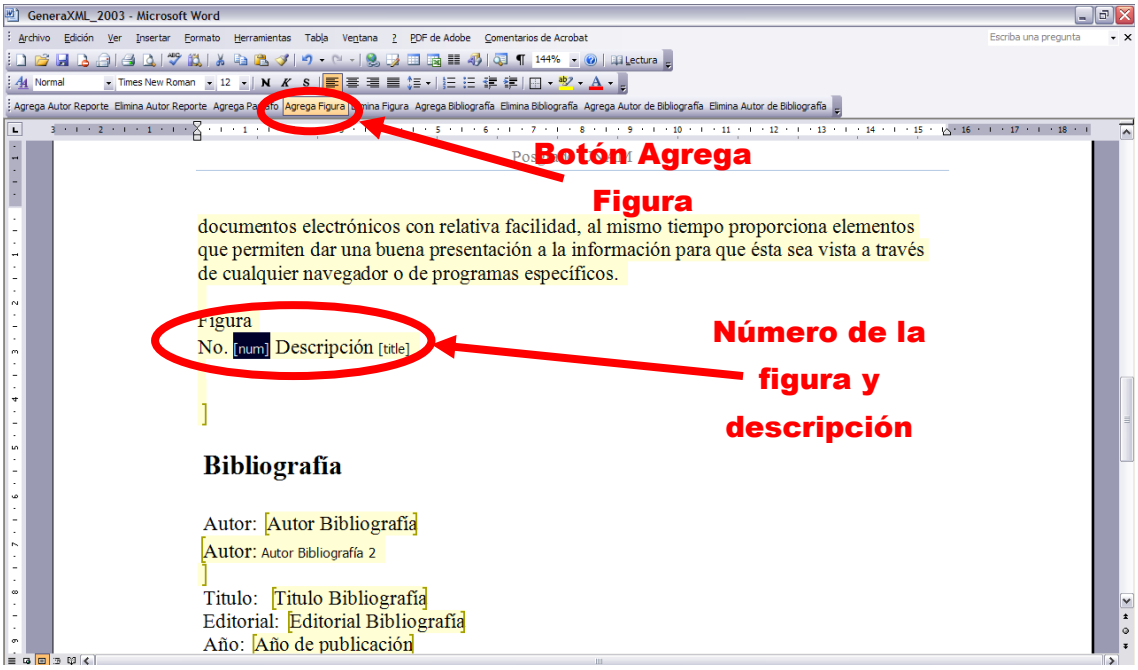


Para agregar otro párrafo en caso de ser necesario, se coloca el cursor en el lugar donde se desea insertar la información y se utiliza el botón "**Agrega Párrafo**" y posteriormente se captura o pega la información correspondiente.



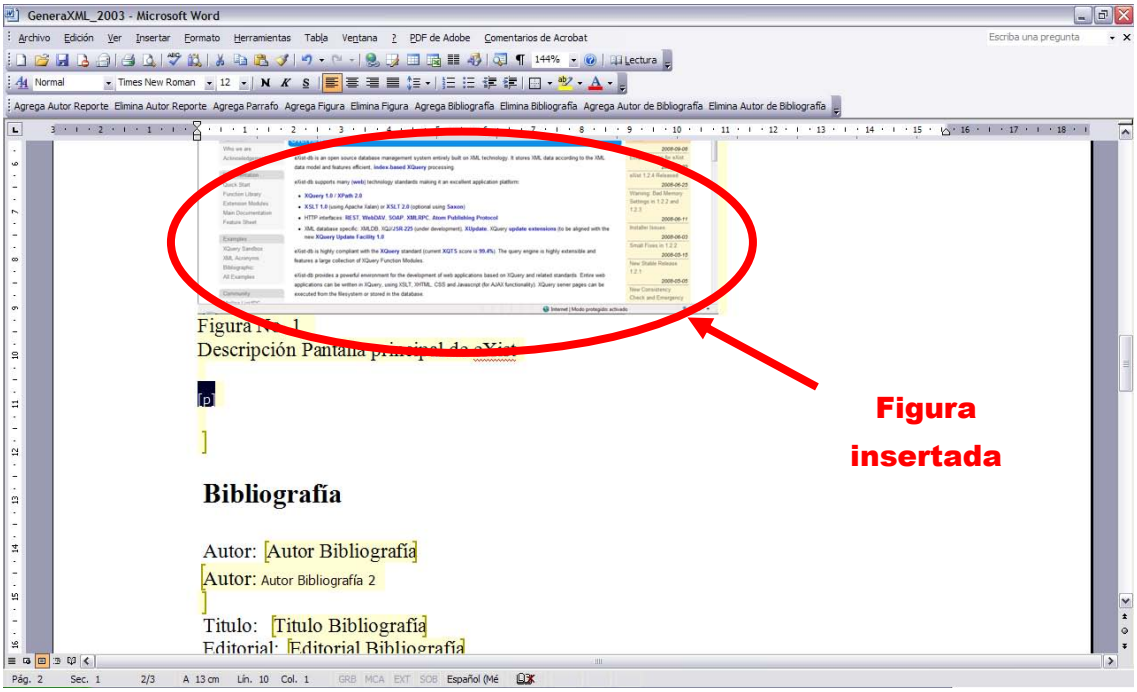
Para eliminar un párrafo, solo basta con seleccionarlo y presionar la tecla "**Supr**"

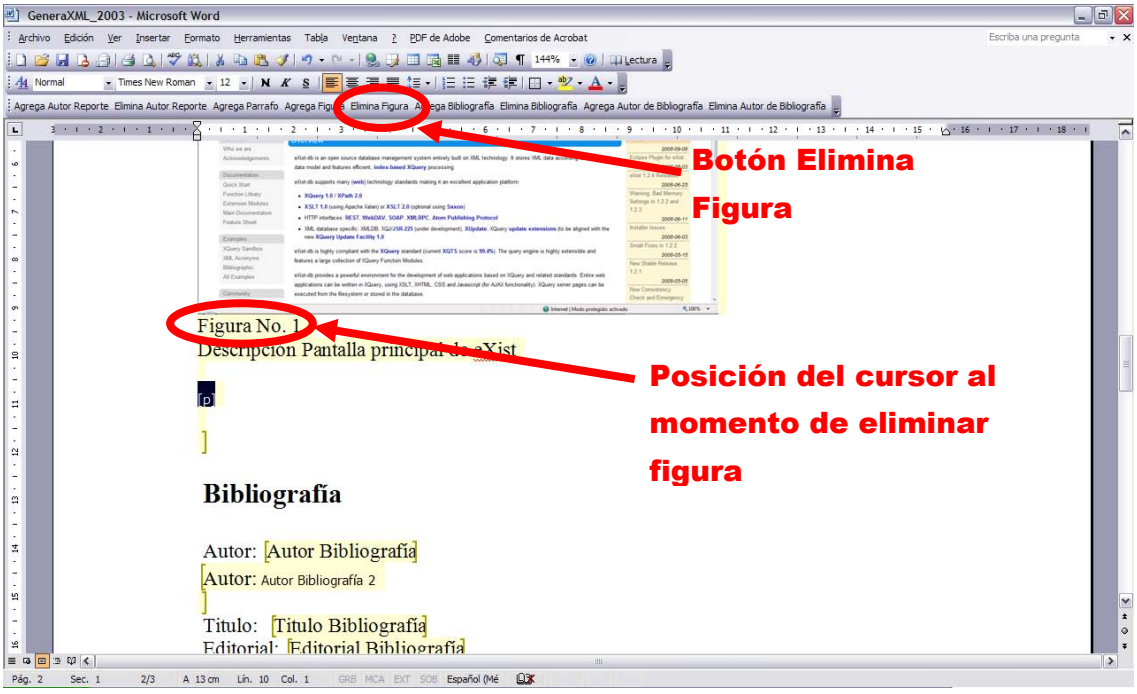
Si se desea agregar alguna figura, existe el botón de "**Agrega Figura**", el cual inserta los campos para indicar el número de la figura y una breve descripción de la misma.



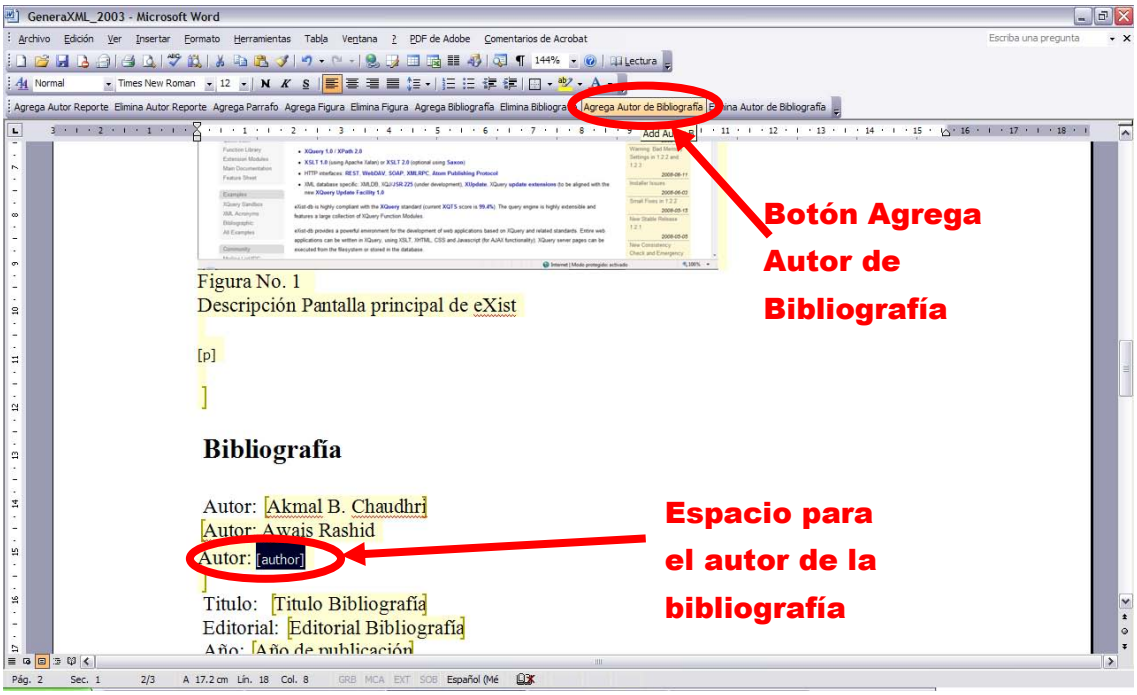
Después deberá insertar la figura correspondiente, de la manera tradicional como lo hace en Microsoft Word.

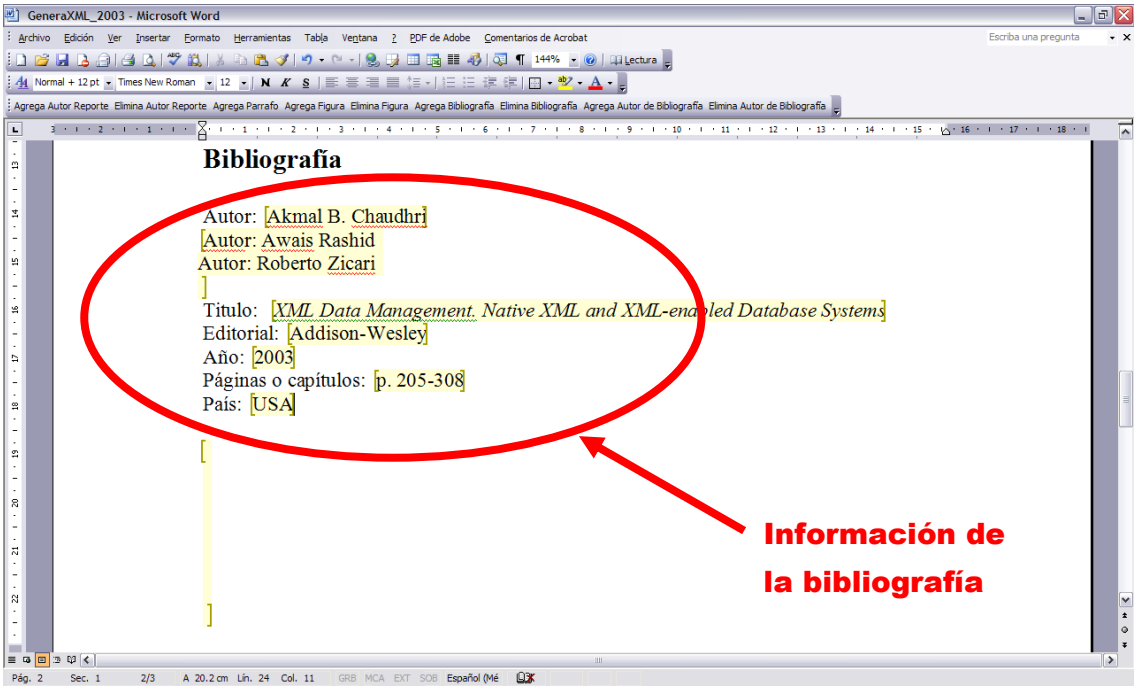
Para eliminar la figura, en caso de que sea necesario, se deberá colocar el cursor en el inicio de la etiqueta "**Figura**" y presionar el botón "**Elimina Figura**", y después eliminar la figura de manera normal; o en su defecto seleccionar la figura y las etiquetas de **No.** y D**escripción**, así como la figura y presionar la tecla "**Supr**"



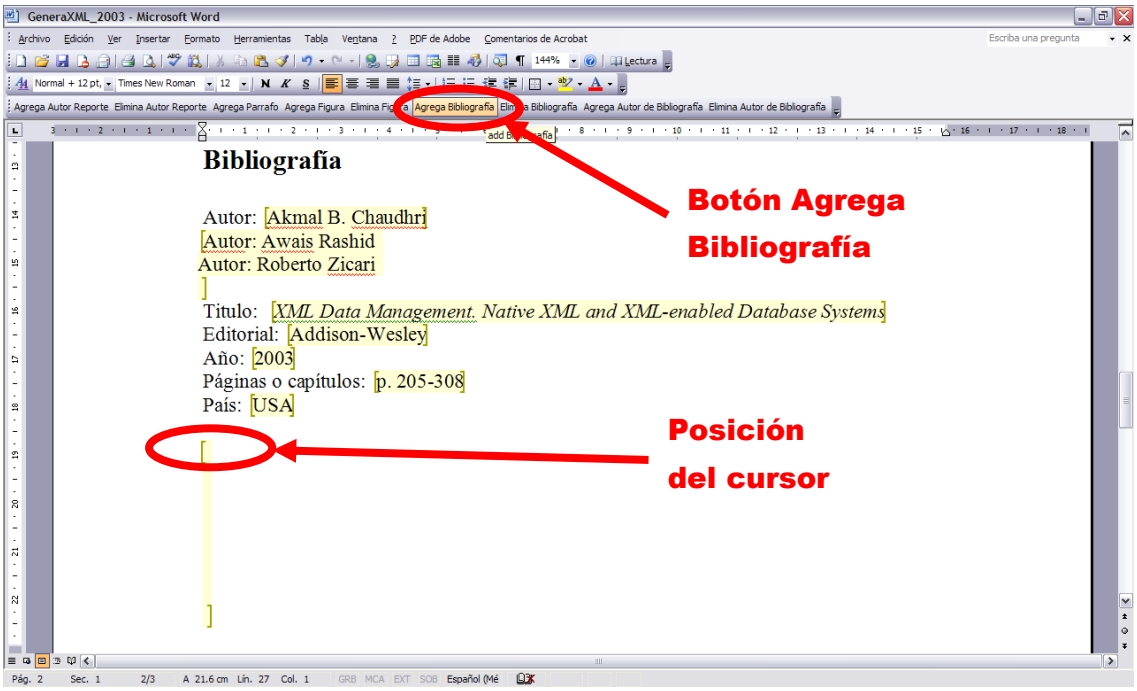


Para finalizar la captura de la información, es necesario detallar la bibliografía utilizada, la cual de manera obligatoria deberá contener al menos una. En el caso de los autores de la bibliografía, sucede lo mismo que con los autores de reporte, sólo que al tratarse de elementos distintos, es necesario utilizar el botón de "**Agrega Autor de Bibliografía**" o "**Elimina Autor de Bibliografía**", es importante mencionar que la posición del cursor es fundamental para el buen funcionamiento del sistema.

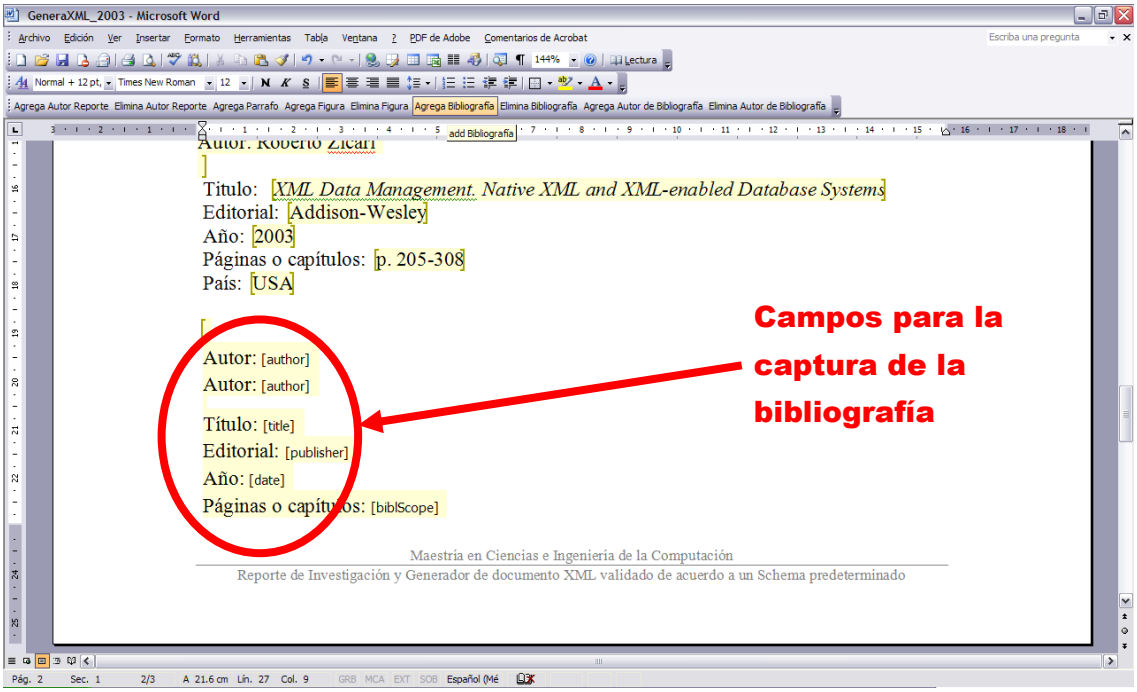




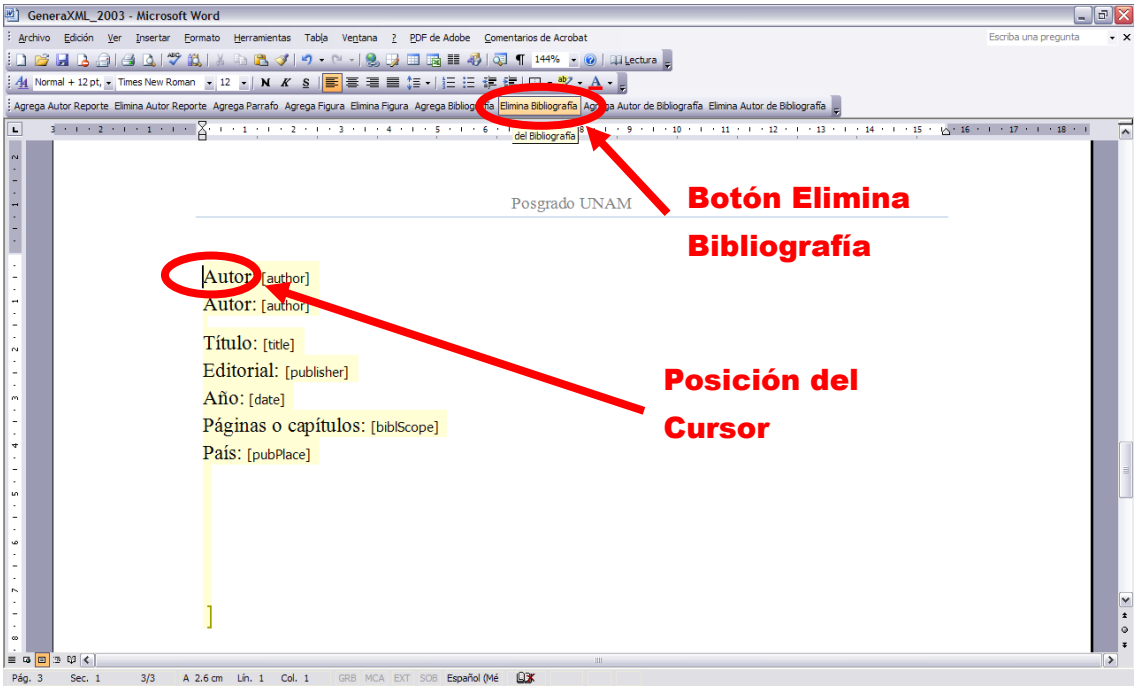
Para agregar otra referencia bibliográfica, es necesario colocar el cursor en el lugar adecuado y presionar el botón "**Agrega Bibliografía**".



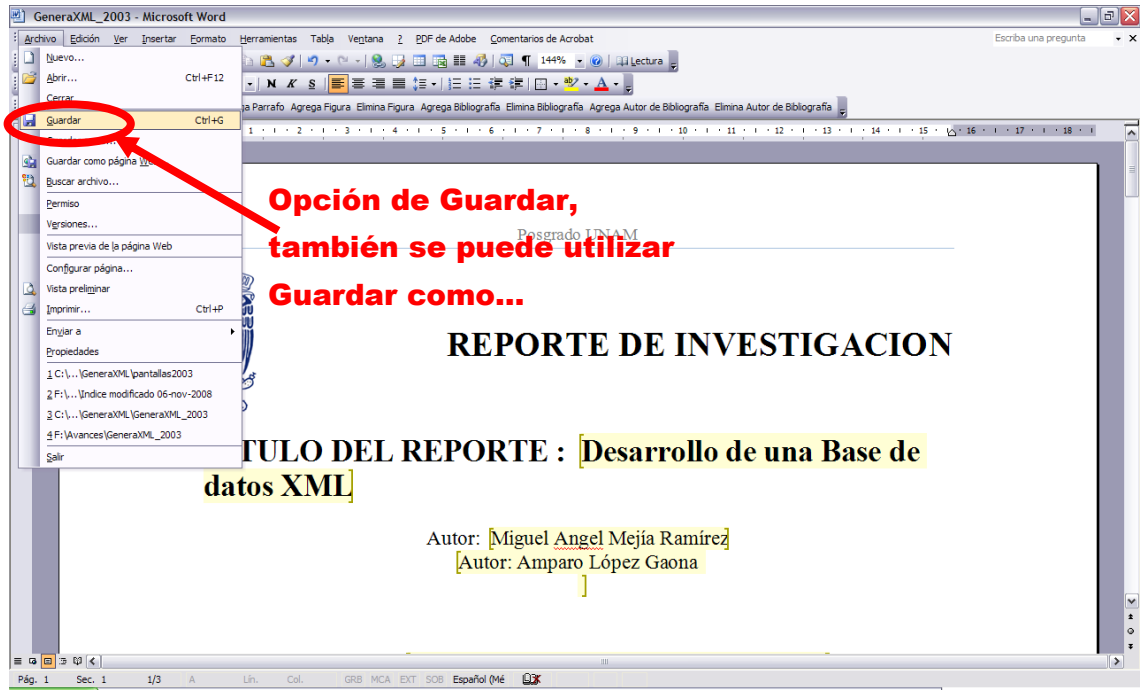
Dicho botón insertará los campos necesarios para la captura de la información.



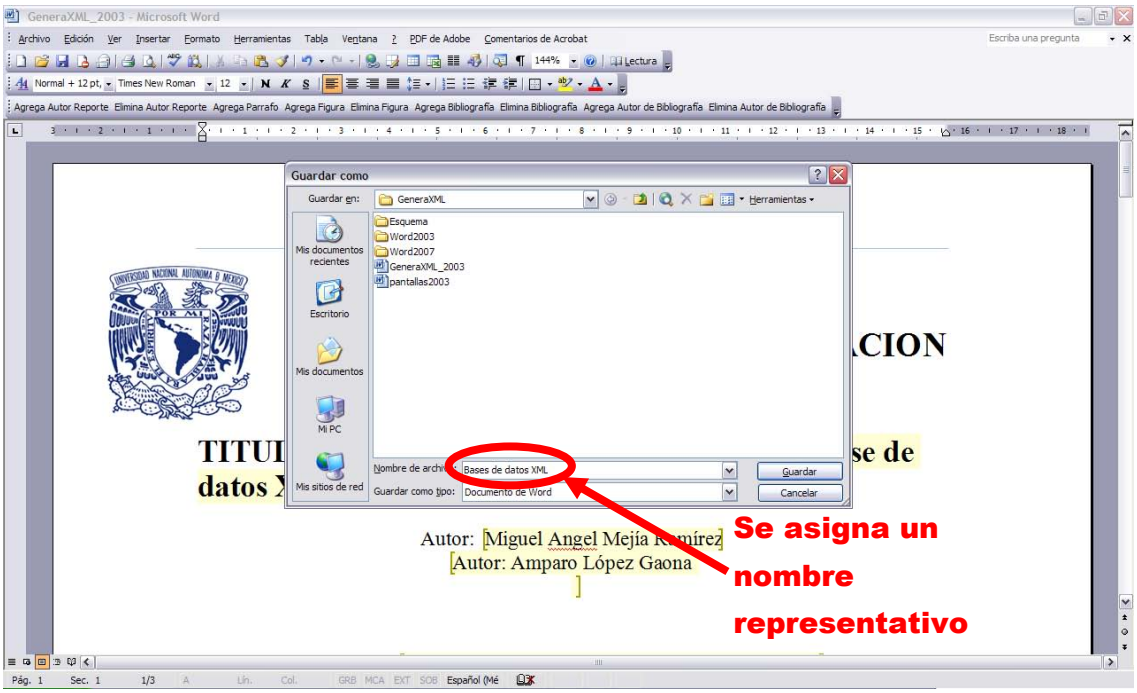
Para eliminar una bibliografía, coloque el cursor al inicio de la etiqueta del primer Autor, y presione el botón "**Elimina Bibliografía**", o en su caso seleccione todos los campos de la bibliografía y presione la tecla "**Supr**".



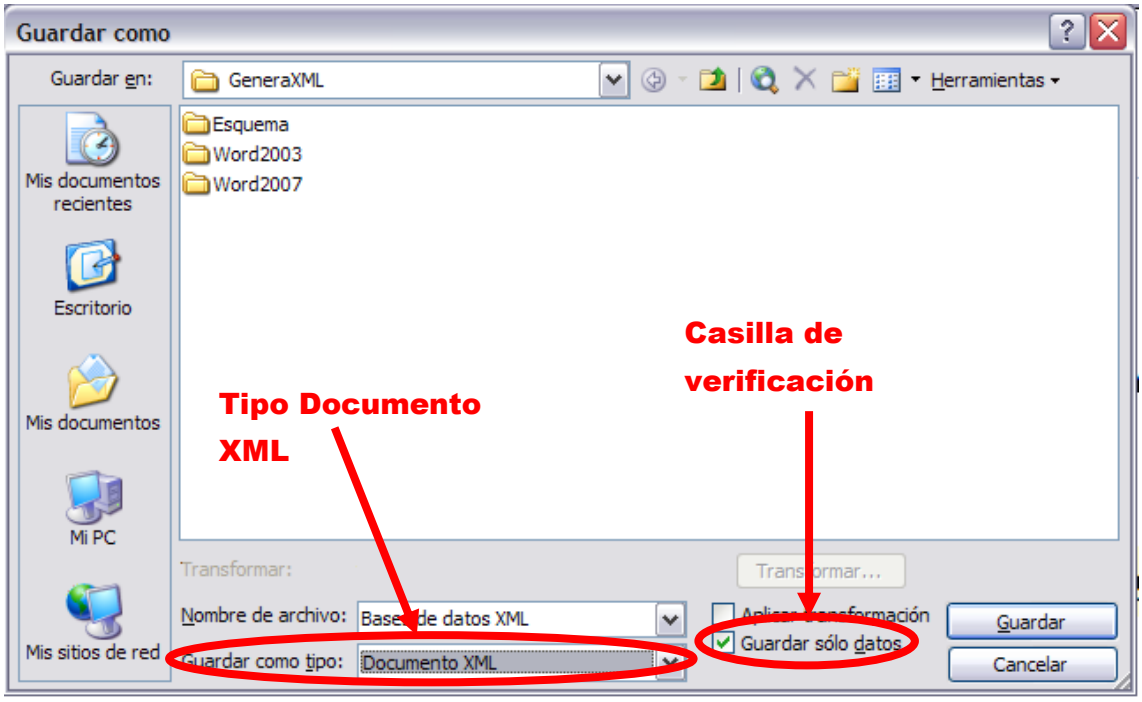
Una vez que se ha capturado toda la información del reporte de investigación, es necesario guardar el archivo como un documento normal de Microsoft Word, para lo que se recomienda utilizar un nombre representativo de acuerdo a la información que contiene.



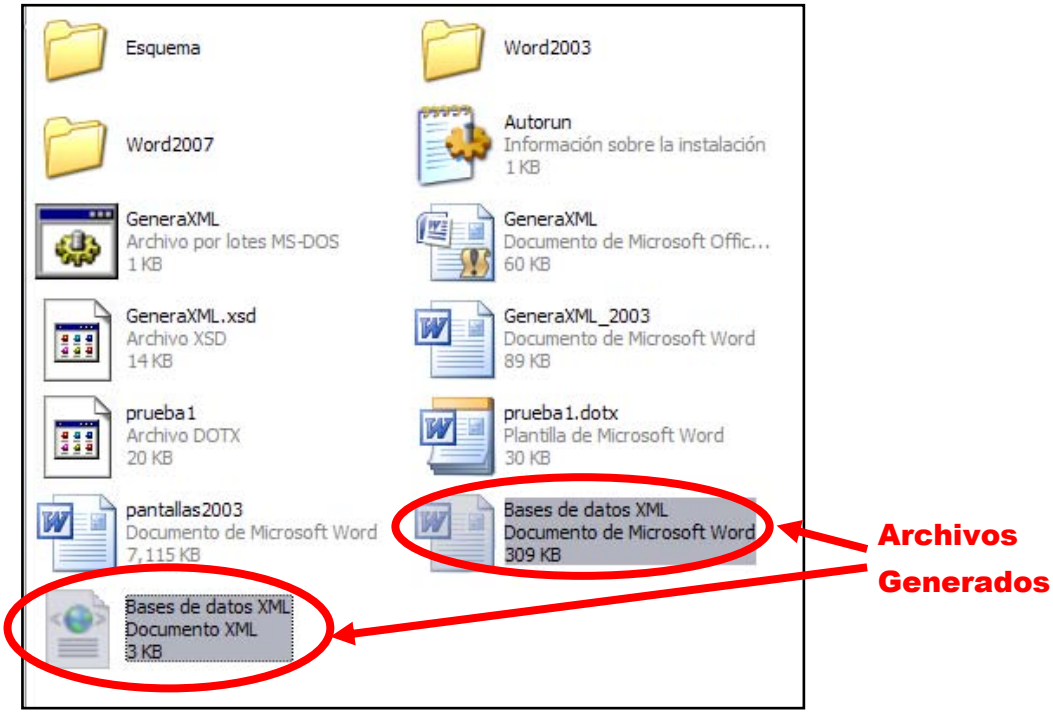
Seleccionando la opción “Guardar como…”, recuerde que se recomienda utilizar un nombre representativo.



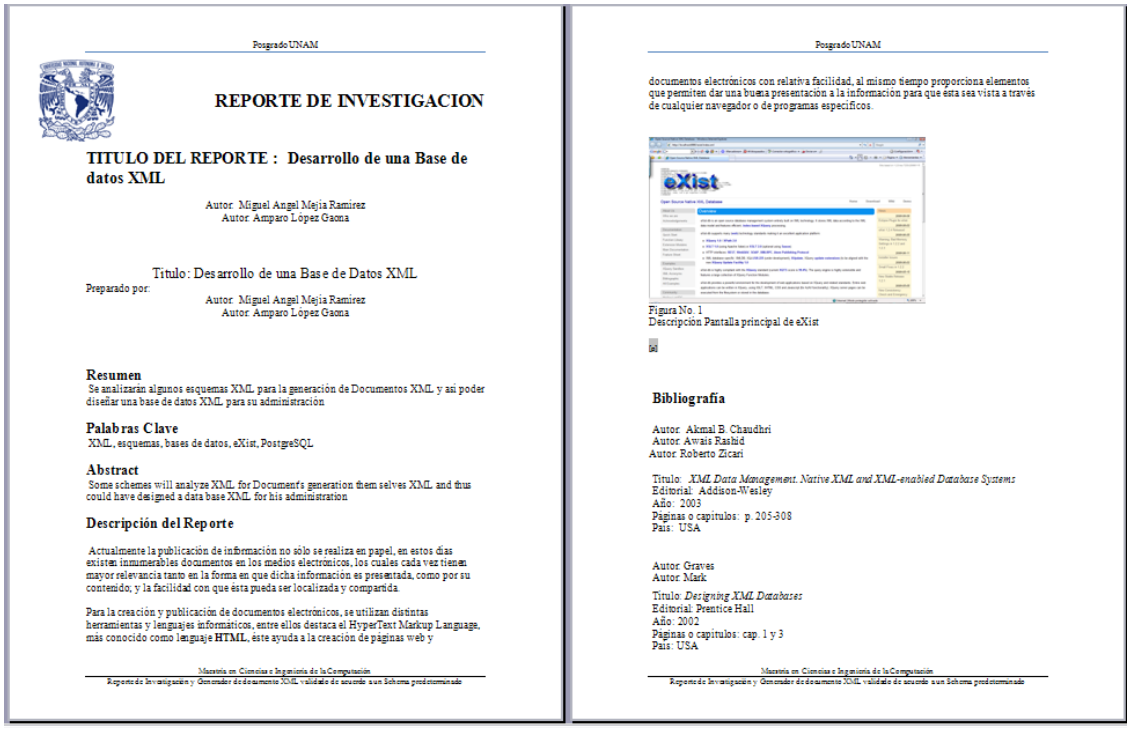
Una vez que se ha guardado el archivo, es indispensable generar el documento XML, para ello se selecciona nuevamente la opción de “**Guardar como…**”, pero en este caso, se indicará el formato “**Documento XML**”, observando que la casilla de verificación de “Guardar sólo datos” este seleccionada.



De esta manera se tienen finalmente dos archivos, uno de Microsoft Word que contiene todo el reporte de investigación y otro que es un documento XML, que almacena los datos de acuerdo a un esquema específico.



El documento de Microsoft Word puede ser modificado las veces que sea necesario y tendría una estructura similar a la mostrada en la siguiente imagen.



Posgrado UNAM

REPORTE DE INVESTIGACION

TITULO DEL REPORTE : Desarrollo de una Base de datos XML

Autor: Miguel Angel Mejia Ramirez
Autor: Amparo Lopez Gaona

Titulo: Desarrollo de una Base de Datos XML
Preparado por:
Autor: Miguel Angel Mejia Ramirez
Autor: Amparo Lopez Gaona

Resumen
Se analizaran algunos esquemas XML para la generación de Documentos XML y asi poder diseñar una base de datos XML para su administración

Palabras Clave
XML, esquemas, bases de datos, eXist, PostgreSQL

Abstract
Some schemes will analyze XML for Documenfs generation them selves XML and thus could have designed a data base XML for his administration

Descripción del Reporte

Actualmente la publicación de infirmación no sólo se realiza en papel, en estos días existen innumerables documentos en los medios electronicos, los cuales cada vez tienen mayor relevancia tanto en la forma en que dicha informacion es presentada, como por su contenido; y la facilidad con que ésta pueda ser localizada y compartida.

Para la creación y publicación de documentos electrónicos, se utilizan distintas herramientas y lenguajes informáticos, entre ellos destaca el HyperText Markup Language, más conocido como lenguaje **HTML**, éste ayuda a la creación de páginas web y

Maestría en Ciencias e Ingeniería de la Computación
Reporte de Investigación y Generador de documento XML validado de acuerdo a un Schema predeterminado

Posgrado UNAM

documentos electrónicos con relativa facilidad, al mismo tiempo proporciona elementos que permiten dar una buena presentación a la información para que ésta sea vista a través de cualquier navegador o de programas específicos.

Figura No. 1
Descripción Pantalla principal de eXist

Bibliografía

Autor: Akmal B. Chaudhri
Autor: Awais Rashid
Autor: Roberto Zicari

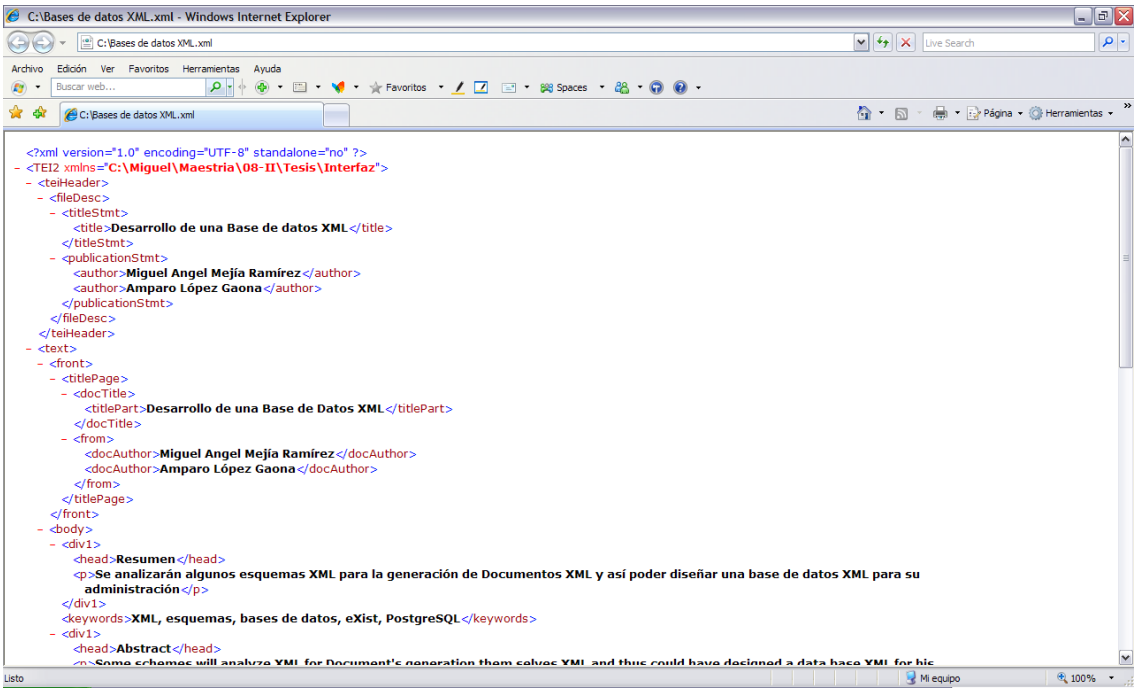
Titulo: *XML Data Management. Native XML and XML-enabled Database Systems*
Editorial: Addison-Wesley
Año: 2003
Paginas o capitulos: p. 205-308
País: USA

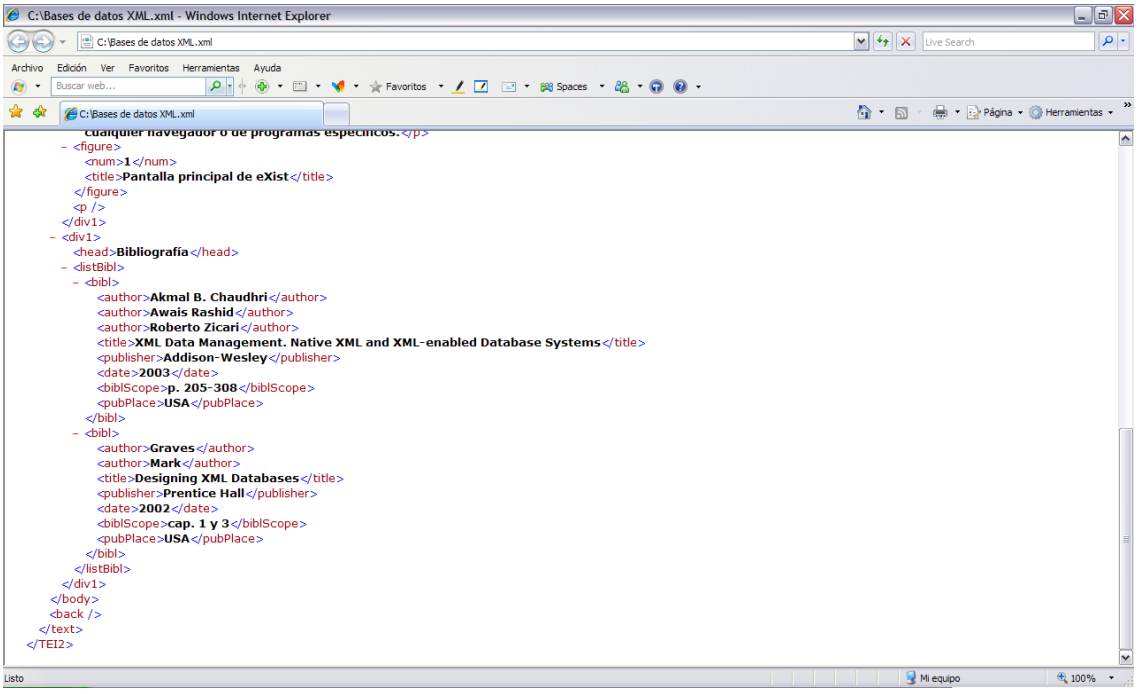
Autor: Graves
Autor: Mark

Titulo: *Designing XML Databases*
Editorial: Prentice Hall
Año: 2002
Paginas o capitulos: cap. 1 y 3
País: USA

Maestría en Ciencias e Ingeniería de la Computación
Reporte de Investigación y Generador de documento XML validado de acuerdo a un Schema predeterminado

El documento XML resultante, tendría una descripción similar a la de las siguientes figuras.





Apéndice B

Manual del Sistema (mantenimiento)

MANUAL DEL SISTEMA

Sistema para generar documentos XML

GeneraXML

Marzo 2009

Introducción

El presente manual tiene por objeto mostrar y documentar el proceso llevado a cabo para la creación del sistema "**GeneraXML**" y con ello facilitar su mantenimiento.

Especificaciones del sistema

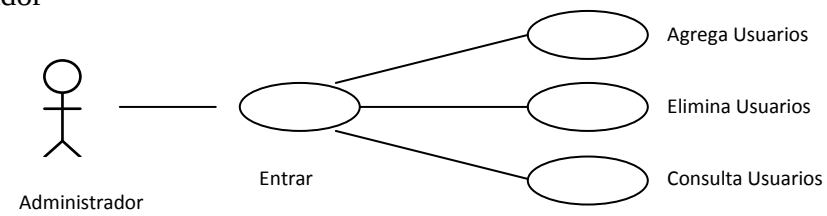
1. Contar con un esquema que permita la especificación de documentos XML.
2. El sistema permite generar Reportes de Investigación, los cuales a su vez son documentos XML, bien formados y validos.
3. Estos documentos XML son almacenados en una bases de datos especial, para su posterior explotación.
4. Se diseñó una base de datos XML para el almacenamiento y recuperación de reportes de investigación.
5. Se diseñó una herramienta en el programa Microsoft Word para la generación de documentos XML a partir de los reportes de investigación.
6. Se diseñó una herramienta en el programa Adobe Designer para la generación de documentos XML a partir de los reportes de investigación.

Análisis

Casos de Uso

Caso de uso: Entrar (al sistema)

Actor: Administrador



Caso de uso Entrar (al Sistema)

Descripción: El Administrador desea entrar al sistema para dar de Alta, Eliminar o Consultar usuarios.

Precondiciones:

- El Administrador debe tener abierto un navegador .
- El Administrador debe teclear la dirección adecuada para acceder a la página de la Base de Datos de Reportes XML.

Flujo:

ACTOR		SISTEMA		
Paso	**Acción**	**Paso**	**Acción**	**Excepción**
1	Teclea en su navegador la dirección de la página del sistema	2	Despliega la interfaz principal para seleccionar entre Entrar, Salir o Acerca de…	E1
3	Selecciona el botón Entrar	4	Despliega la interfaz de Administrador, solicitando los datos para su ingreso	

Excepciones:

Id	Nombre	Acción
E1	Dirección incorrecta	La página no se despliega en el navegador

Postcondiciones:

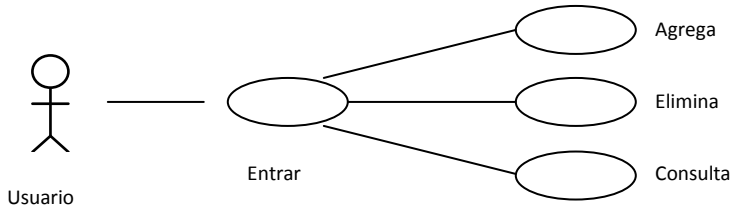
- El sistema ha desplegado la interfaz de Administrador.

Plan de Pruebas del Sistema

Caso de uso	Entrada	Esperado	Obtenido
Entrar	Dirección de la página	Dirección correcta	Despliega interfaz inicial
	Dirección de la página	Dirección incorrecta	El navegador no despliega interfaz inicial

Caso de uso: Entrar (al sistema)

Actor: Usuario



Caso de uso Entrar (al Sistema)

Descripción: El Usuario desea entrar al sistema para dar del Alta, Eliminar o Consultar un Documento XML.

Precondiciones:

- El Usuario debe tener abierto un navegador y además de estar conectado a Internet.
- El Usuario debe teclear la dirección adecuada para acceder a la página de La Base de Datos de Reportes XML.

Flujo:

ACTOR		SISTEMA		
Paso	**Acción**	**Paso**	**Acción**	**Excepción**
1	Teclea en su navegador la dirección de la página del sistema	2	Despliega la interfaz principal para seleccionar entre Entrar, Salir o Acerca de…	E1
3	Selecciona el botón Entrar	4	Despliega la interfaz de Documentos XML, solicitando los datos para su ingreso	

Excepciones:

Id	Nombre	Acción
E1	Dirección incorrecta	La página no se despliega en el navegador

Postcondiciones:

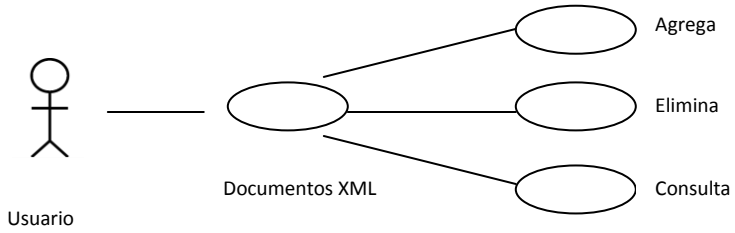
- El sistema ha desplegado la interfaz de Documentos XML.

Plan de Pruebas del Sistema

Caso de uso	Entrada	Esperado	Obtenido
Entrar	Dirección de la página	Dirección correcta	Despliega interfaz inicial
	Dirección de la página	Dirección incorrecta	El navegador no despliega interfaz inicial

Caso de uso: Administrar Documentos XML

Actor: Usuario



Caso de uso Administrar Documentos XML

Descripción: El Usuario debe elegir una opción del menú Documentos XML, las cuales pueden ser Agrega, Elimina o Consulta.

Precondiciones:

- El sistema despliega interfaz del menú Documentos XML.

Flujo:

ACTOR		SISTEMA		
Paso	**Acción**	**Paso**	**Acción**	**Excepción**
1	Elige opción Agrega	2	Despliega la interfaz Agrega Documento XML	E1
1	Elige opción Elimina	2	Despliega la interfaz Elimina Documento XML	E1
1	Elige opción Consulta	2	Despliega la interfaz Consulta Documento XML	E1
1	Elige Botón Salir	2	Sale del Sistema	E1

Excepciones:

Id	Nombre	Acción
E1	Red desconectada	Mostrar Mensaje de "Red desconectada"

Postcondiciones:

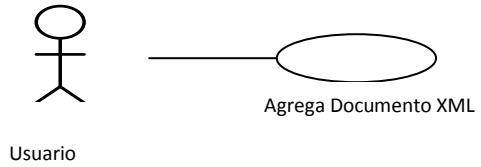
- El sistema presenta la Interfaz de Agrega, Elimina o Consulta Documentos XML.
- El sistema se cierra.

Plan de Pruebas del Sistema

Caso de uso	Entrada	Esperado	Obtenido
Administrar Documentos XML	Agrega Documento XML	Opción Agrega Documento XML	Interfaz Agrega Documento XML
	Elimina Documento XML	Opción Elimina Documento XML	Interfaz Elimina Documento XML
	Consulta Documento XML	Opción Consulta Documento XML	Interfaz Consulta Documento XML
	Salir	Botón Salir	Cierre del Sistema

Caso de uso: Agrega Documento XML

Actor: Usuario



Caso de uso Agrega Documento XML.

Descripción: El Usuario proporcionará los datos necesarios (Nombre u ubicación del Documento XML) para Agregar el Documento XML a la Base de Datos.

Precondiciones:

- El sistema presenta la Interfaz de Agrega Documento XML.

Flujo:

ACTOR		SISTEMA		
Paso	**Acción**	**Paso**	**Acción**	**Excepción**
1	Teclea todos los datos solicitados para Agregar un Documento XML a la Base de Datos			
2	Presiona Botón Aceptar	3	Verifica que todos los datos se hayan capturado de manera correcta y que el Documento XML exista	E1, E2
		4	Verifica que el Documento XML no exista	E3
		5	Guarda la información y el Documento XML en la Base de Datos	
1	Presiona Botón Cancelar	2	Regresa a la Interfaz del menú Documentos XML	

Excepciones:

Id	Nombre	Acción
E1	Faltan datos	Presenta mensaje de Error “Falta el dato _____, favor de capturarlo”
E2	Documento XML no existe en esa ruta	Presenta mensaje “Documento XML no encontrado”
E3	Documento XML duplicado	Presenta mensaje “Documento duplicado”

Postcondiciones:

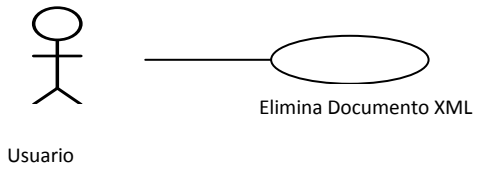
- El sistema guarda la información y el Documento XML en la Base de Datos.
- El sistema regresa a la interfaz de menú Documentos XML.

Plan de Pruebas del Sistema

Caso de uso	Entrada	Esperado	Obtenido
Agrega Documento XML	Botón Aceptar	Datos completos y correctos del Documento XML	Guarda de Información en la Base de Datos
	Botón Aceptar	Datos incompletos y correctos del Documento XML	Mensaje de Error "Falta el dato _____, favor de capturarlo"
	Botón Aceptar	Datos completos e incorrectos del Documento XML	Mensaje de Error "Documento XML no encontrado"
	Botón Aceptar	Documento XML duplicado	Presenta mensaje "Documento duplicado"
	Cancelar	Botón Cancelar	Regresa a la Interfaz del menú Documentos XML

Caso de uso: Elimina Documento XML

Actor: Usuario



Caso de uso Elimina Documento XML.

Descripción: El Usuario proporcionará los datos necesarios (Nombre del Documento XML) para Eliminar el Documento XML de la Base de Datos.

Precondiciones:

- El sistema presenta la Interfaz de Elimina Documento XML.

Flujo:

ACTOR		SISTEMA		
Paso	**Acción**	**Paso**	**Acción**	**Excepción**
1	Teclea todos los datos solicitados para Eliminar el Documento XML			
2	Presiona Botón Aceptar	3	Verifica que exista el Documento XML	E1
		4	Muestra mensaje de Advertencia “Esta usted seguro? S/N”	
5	Elige opción S	6	Elimina la información del Documento XML de la Base de Datos	
5	Elige opción N	6	Cancela opción de Elimina Documento XML	
1	Presiona Botón Cancelar	2	Regresa a la Interfaz del menú Documentos XML	

Excepciones:

Id	Nombre	Acción
E1	Documento XML Inexistente	Presenta mensaje de Error “El Documento XML no existe, corrija por favor”

Postcondiciones:

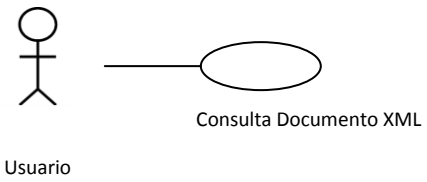
- El sistema elimina la información y el Documento XML de la Base de Datos.
- El sistema regresa a la interfaz de menú Documentos XML.

Plan de Pruebas del Sistema

Caso de uso	Entrada	Esperado	Obtenido
Elimina Documento XML	Botón Aceptar	Datos correctos del Documento XML	Elimina la información y el Documento XML de la Base de Datos
	Botón Aceptar	Datos incorrectos del Documento XML	Mensaje de Error "El Documento XML no existe, corrija por favor"
	Cancelar	Botón Cancelar	Regresa a la Interfaz del menú Documentos XML

Caso de uso: Consulta Documento XML

Actor: Usuario



Caso de uso Consulta Documento XML.

Descripción: El Usuario proporcionará los datos necesarios (Nombre del Documento XML) para Consultar el Documento XML en la Base de Datos.

Precondiciones:

- El sistema presenta la Interfaz de Consulta Documento XML.

Flujo:

ACTOR		SISTEMA		
Paso	**Acción**	**Paso**	**Acción**	**Excepción**
1	Teclea todos los datos solicitados para realizar la Consulta del Documento XML			
2	Presiona Botón Aceptar	3	Verifica que exista el Documento XML	E1
		4	Muestra información del Documento XML	
1	Presiona Botón Cancelar	2	Regresa a la Interfaz del menú Documentos XML	

Excepciones:

Id	Nombre	Acción
E1	Documento XML Inexistente	Presenta mensaje de Error "El Documento XML no existe, corrija por favor"

Postcondiciones:

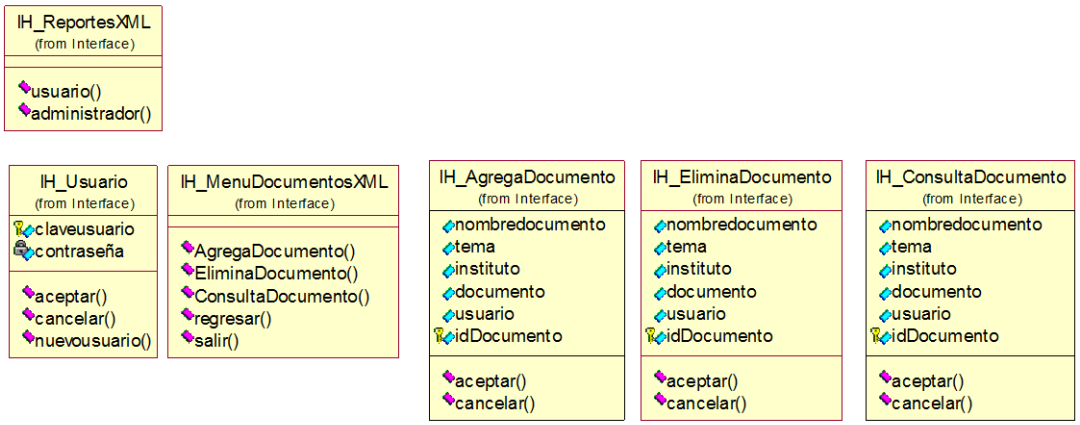
- El sistema presenta la información del Documento XML, almacenado en la Base de Datos.
- El sistema regresa a la interfaz de menú Documentos XML.

Plan de Pruebas del Sistema

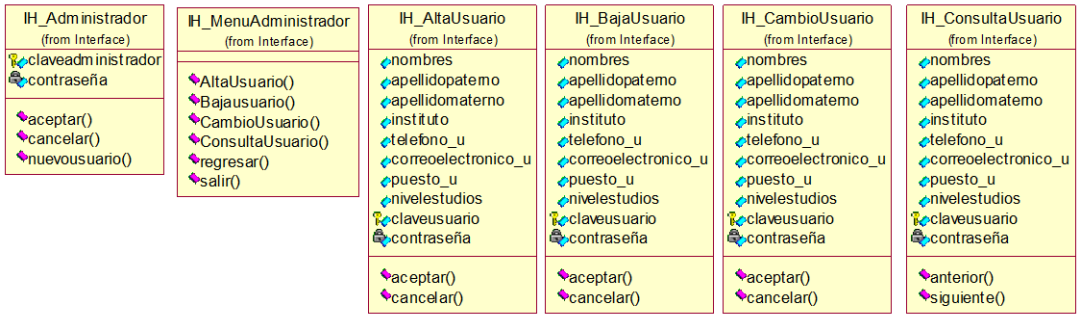
Caso de uso	Entrada	Esperado	Obtenido
Consulta de Documento XML	Botón Aceptar	Datos correctos del Documento XML	Muestra información del Documento XML
	Botón Aceptar	Datos incorrectos del Documento XML	Mensaje de Error "El Documento XML no existe, corrija por favor"
	Cancelar	Botón Cancelar	Regresa a la Interfaz del menú Documentos XML

Diagramas de Clases de Interfaz

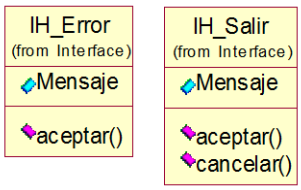
Interfaz principal y Administración de Documentos XML



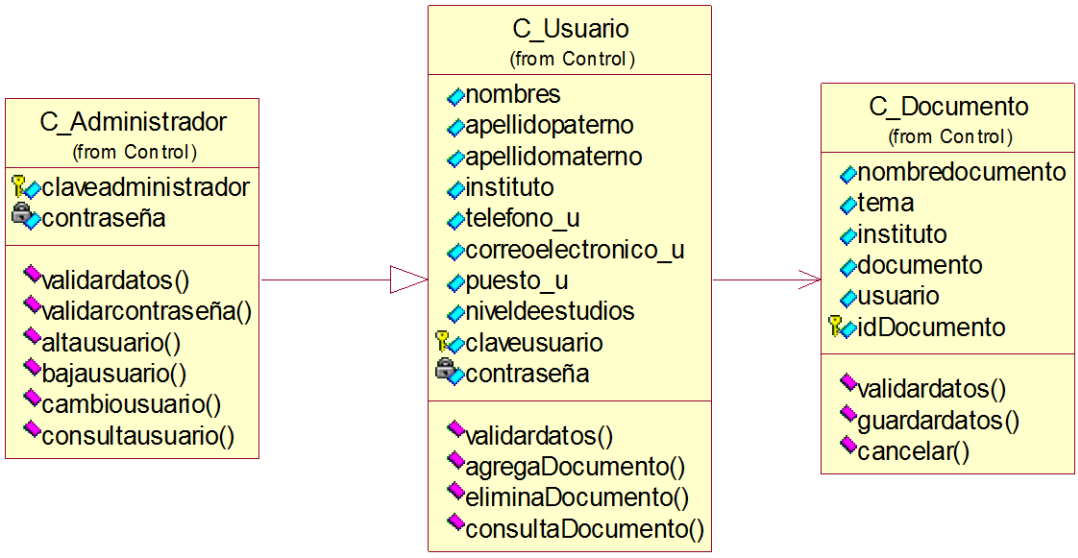
Interfaz de Administrador



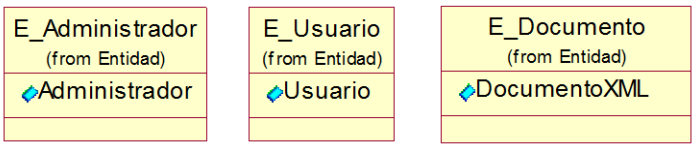
Interfaces de Error y Salir



Diagramas de Clases de Control



Diagramas de Clases de Entidad



Diagramas de Secuencias

Diagrama de secuencia para el caso de uso Entrar al sistema Reportes XML (Usuario).

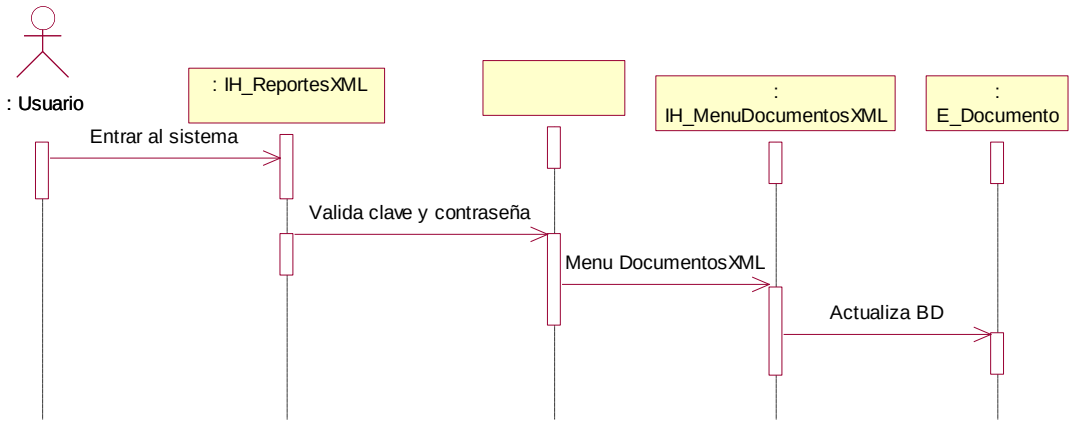


Diagrama de secuencia para el caso de uso Agregar Documento

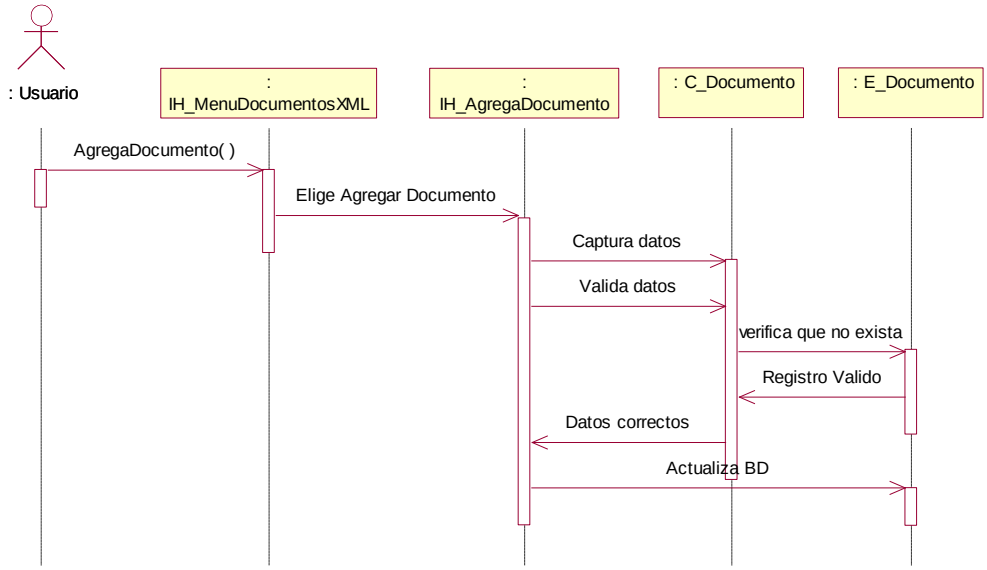


Diagrama de secuencia para el caso de uso Elimina Documento

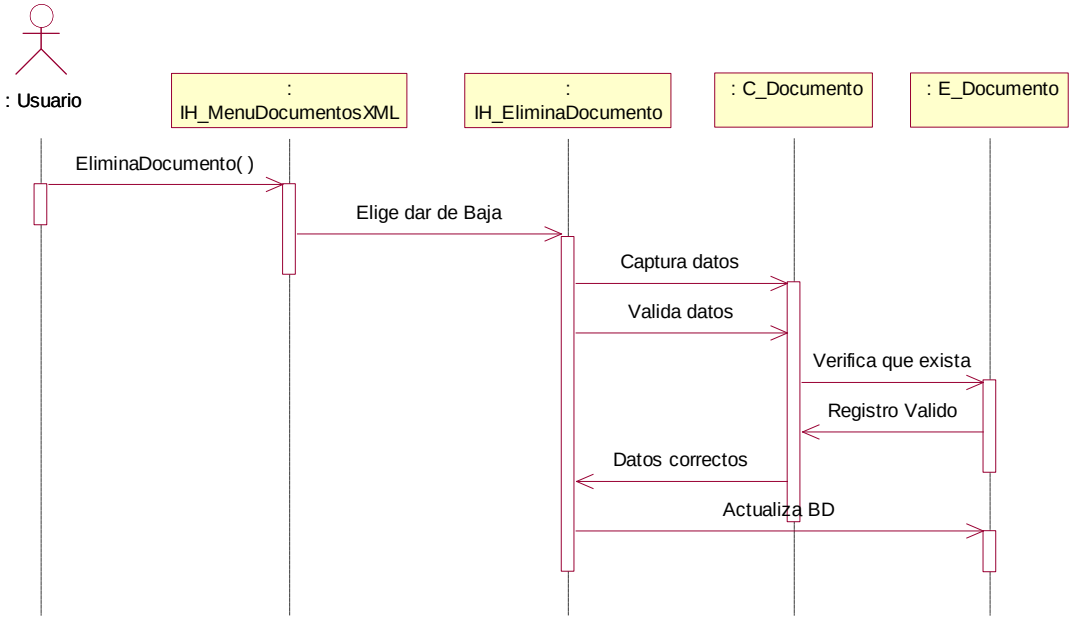


Diagrama de secuencia para el caso de uso Consulta Documento

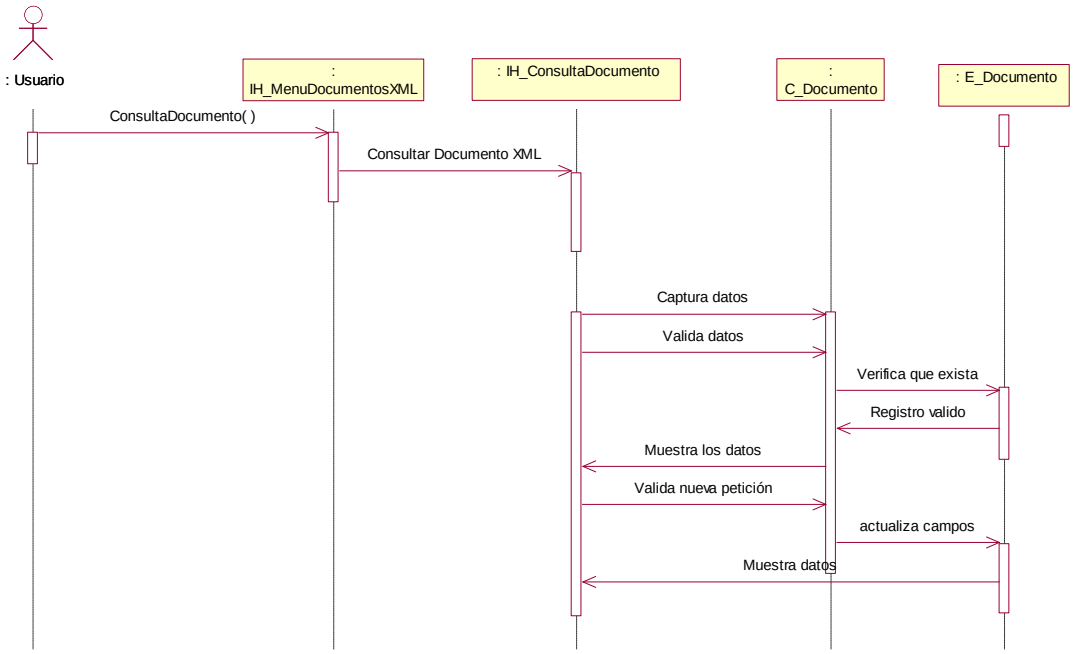


Diagrama de secuencia para el caso de uso Salir del sistema (Usuario)

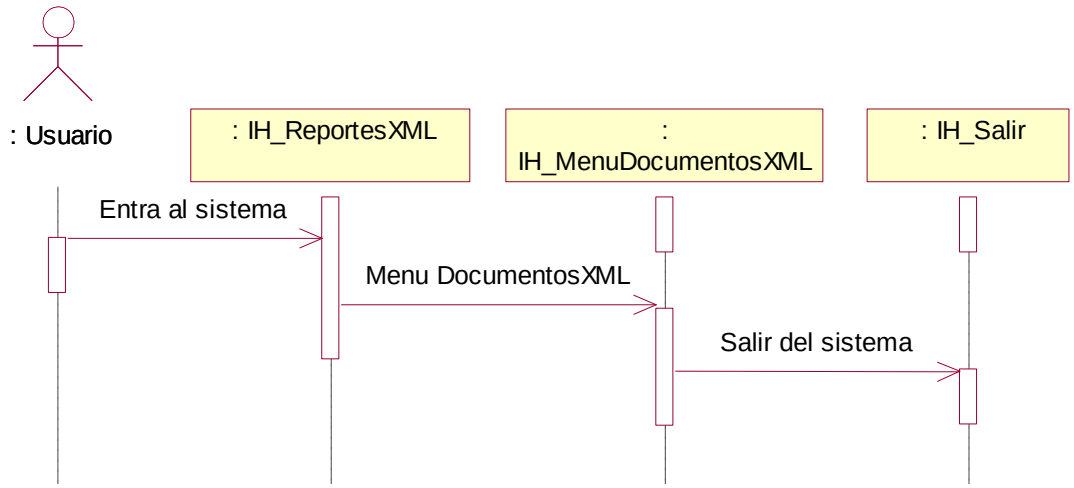


Diagrama de secuencia para el caso de uso Entrar al sistema Reportes XML (Administrador).

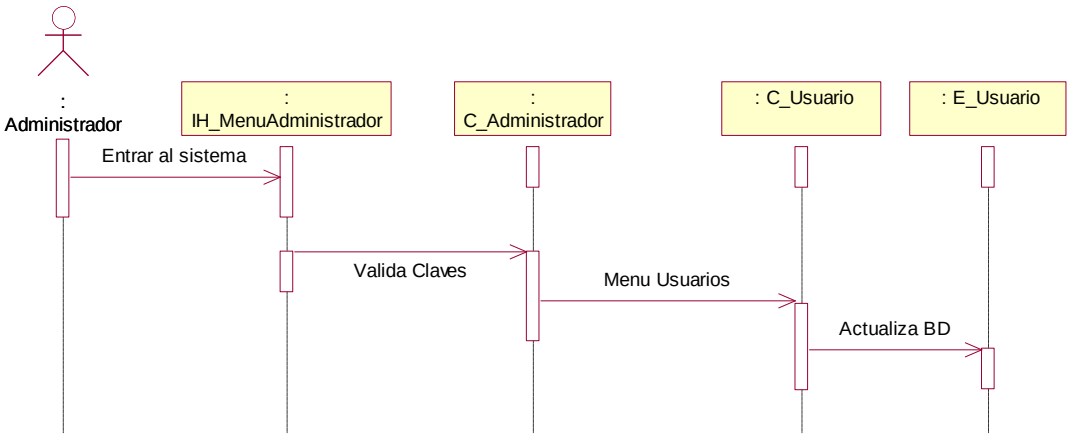


Diagrama de secuencia para el caso de uso Alta Usuario

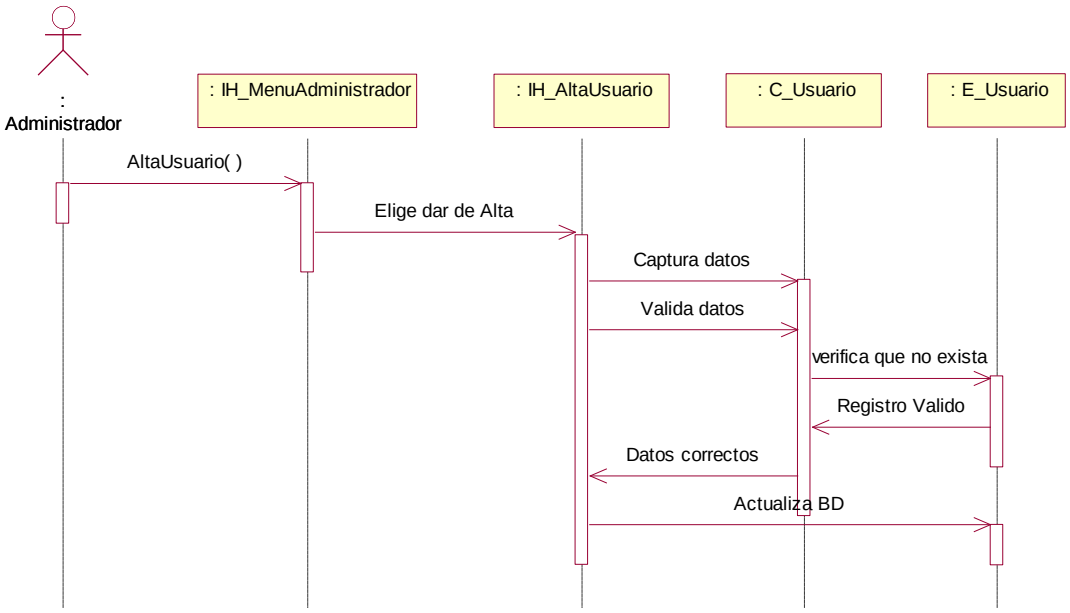


Diagrama de secuencia para el caso de uso Baja Usuario

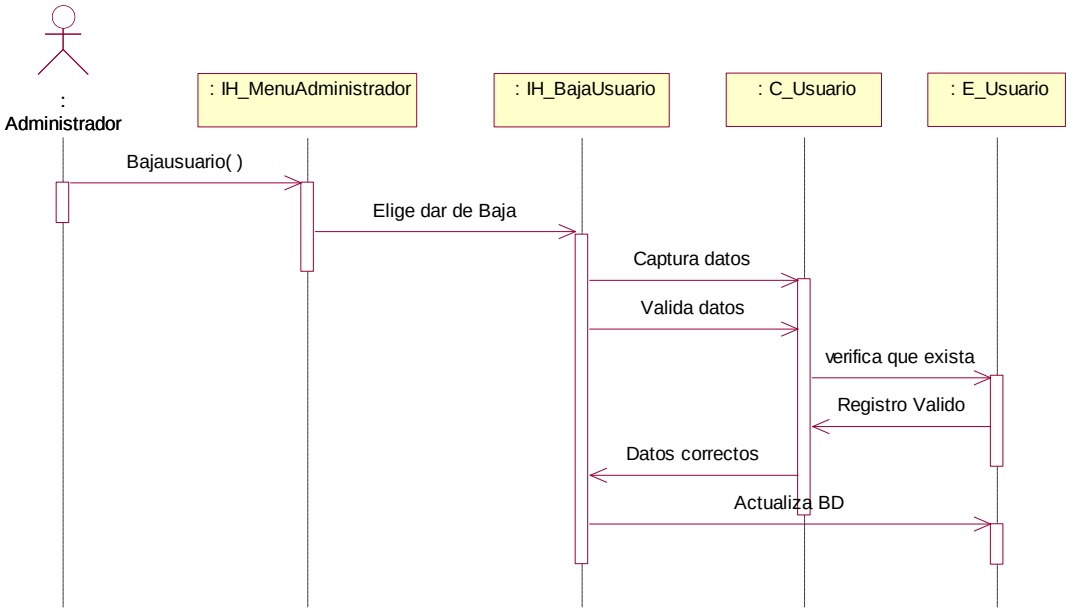


Diagrama de secuencia para el caso de uso Cambio Usuario

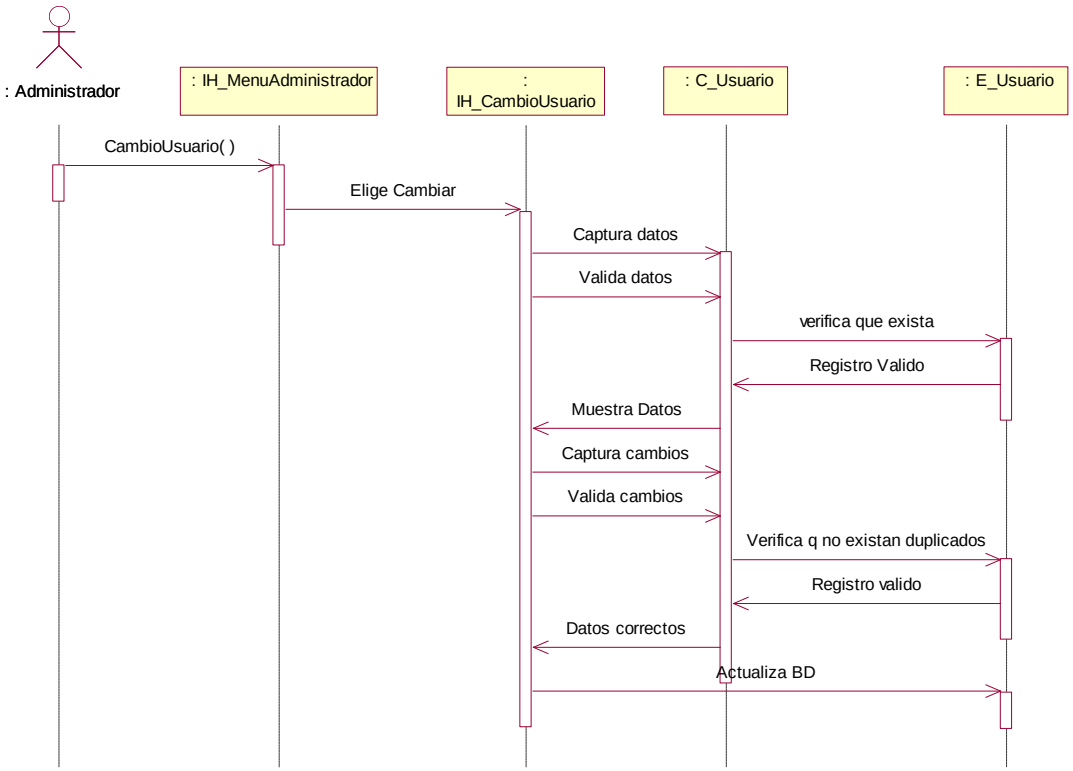


Diagrama de secuencia para el caso de uso Consulta Usuario

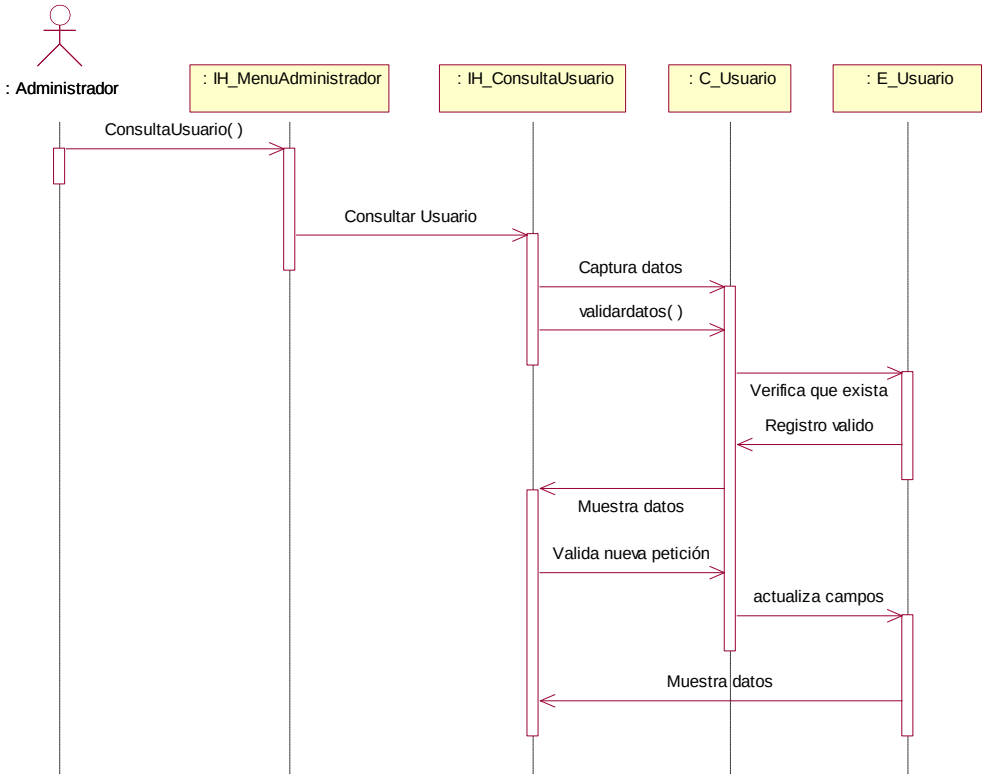


Diagrama de secuencia para el caso de uso Salir del sistema (Administrador)

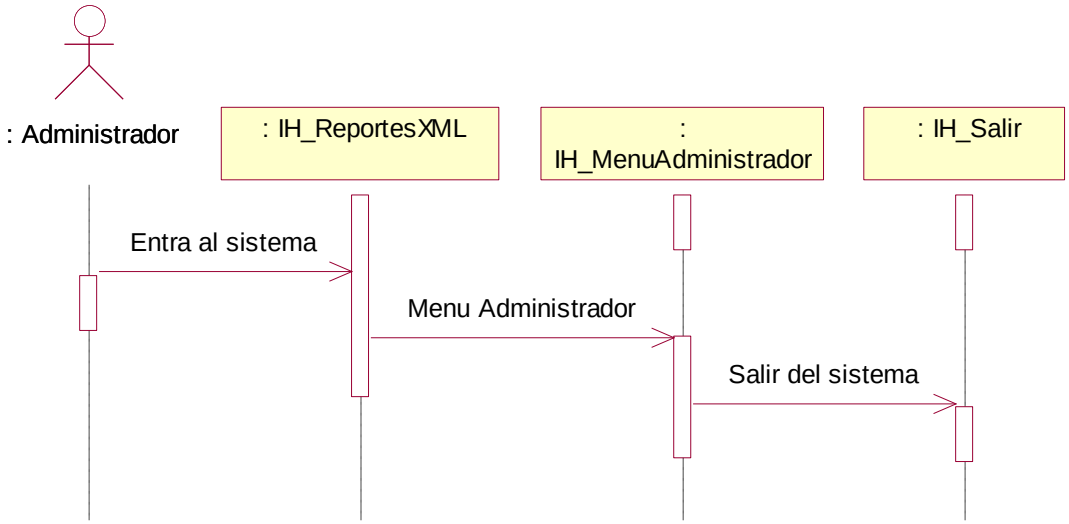
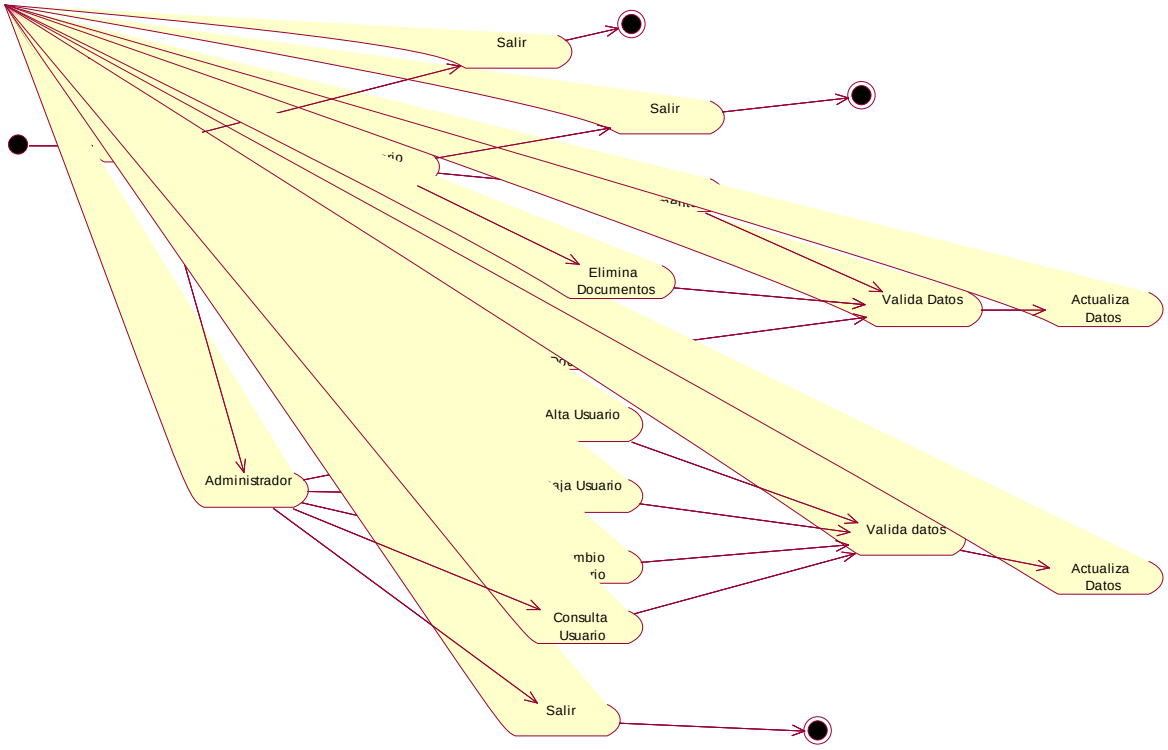


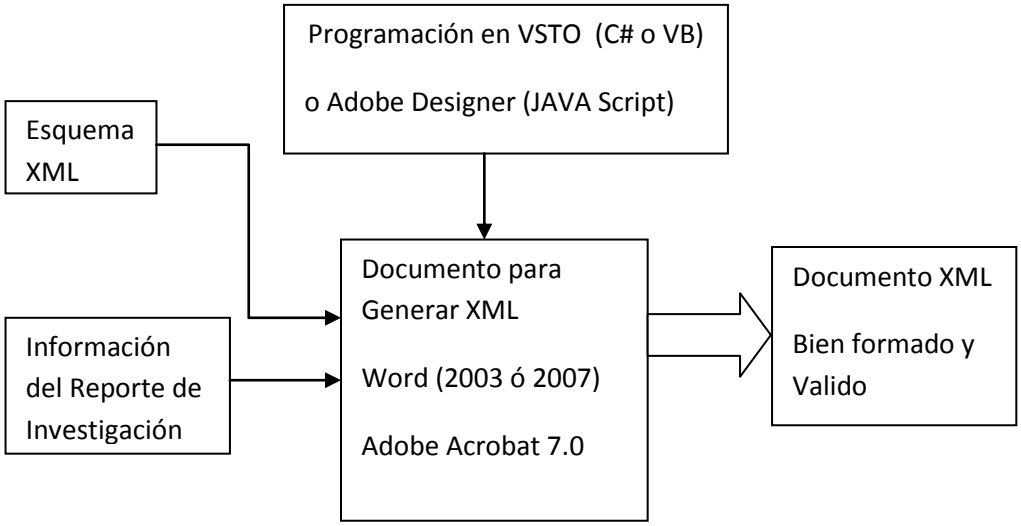
Diagrama de Estados



Generación de Documentos XML "Bien formados y Validos"

Para la realización de los documentos XML, se aprovechó el conocimiento que poseen los usuarios en herramientas de procesamiento de texto, como son Microsoft Word y Adobe Acrobat.

Se utilizaron los paquetes Visual Studio Tools for Office y Adobe Designer, para realizar una programación a nivel de aplicación, y proporcionar un documento de Microsoft Word y otro de en formato PDF respectivamente, incluyendo un esquema XML predeterminado para verificar la información que se va capturando, y así generar un Documento XML bien formado y valido.



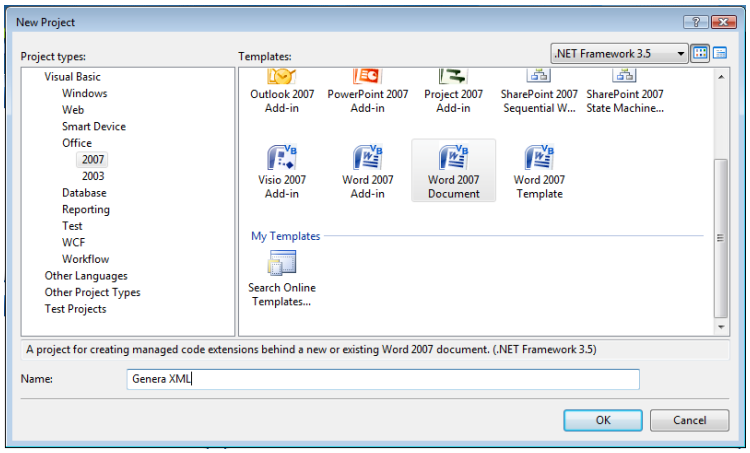
Generador de Documentos XML

Desarrollo

Se utilizo la versión 2008 de Microsoft Visual Studio Professional Edition, con su complemento Visual Studio Tools for Office, el cual permite realizar programación a nivel de aplicaciones para los documentos de Microsoft Office.

Pantalla de Visual Studio 2008 con el complemento VSTO

Como Visual Studio Tools for Office permite la programación tanto en lenguaje Visual Basic como en C#, se tiene una mayor gama de funciones para realizar la aplicación, con fines de asegurar la compatibilidad, se diseñaron dos versiones, una para Microsoft Word 2007 y otra para Microsoft Word 2003.



Pantalla para crear un proyecto de Word 2007 en Visual Basic

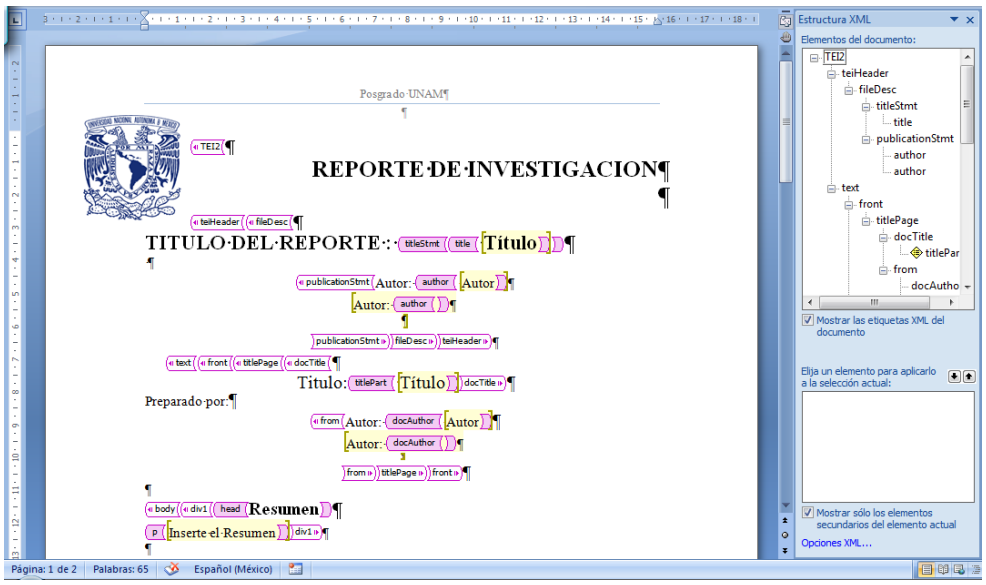
Se tiene el esquema GeneraXML.xsd el cual detalla de manera pormenorizada la estructura que deberán tener los documentos XML resultantes.

A continuación se muestra una parte del esquema.

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
	<xs:import namespace="http://www.w3.org/XML/1998/namespace"/>
	<xs:element name="TEI2">
		<xs:complexType>
			<xs:sequence>
				<xs:element ref="teiHeader"/>
				<xs:element ref="text"/>
			</xs:sequence>
		</xs:complexType>
	</xs:element>
```

```
<xs:element name="abbr">
        <xs:complexType mixed="true"/>
</xs:element>
<xs:element name="author">
        <xs:complexType mixed="true"/>
</xs:element>
<xs:element name="back">
        <xs:complexType>
                <xs:sequence>
                        <xs:element ref="div1" minOccurs="0" maxOccurs="unbounded"/>
                </xs:sequence>
        </xs:complexType>
</xs:element>
<xs:element name="bibl">
        <xs:complexType mixed="true">
                <xs:choice minOccurs="0" maxOccurs="unbounded">
                        <xs:element ref="author"/>
                        <xs:element ref="biblScope"/>
                        <xs:element ref="date"/>
                        <xs:element ref="pubPlace"/>
                        <xs:element ref="title"/>
                        <xs:element ref="publisher"/>
                        <xs:element ref="url"/>
                </xs:choice>
        </xs:complexType>
</xs:element>
<xs:element name="biblScope">
        <xs:complexType mixed="true"/>
</xs:element>
<xs:element name="body">
        <xs:complexType>
                <xs:choice minOccurs="0" maxOccurs="unbounded">
                        <xs:element ref="div1"/>
                        <xs:element ref="keywords"/>
                        <xs:element ref="list"/>
                        <xs:element ref="p"/>
                </xs:choice>
        </xs:complexType>
</xs:element>
```

Este esquema se incorpora al documento de Microsoft Word, para que los elementos puedan ser insertados, con todo y sus etiquetas.



Documento de Word 2007, que muestra etiquetas XML

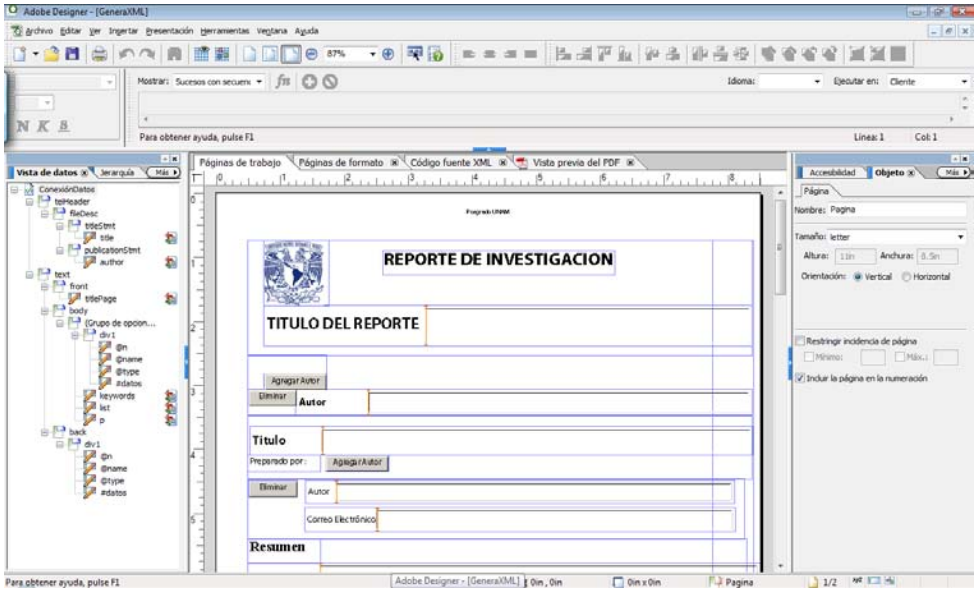
El documento de Word resultante se utilizó como base para programar la barra de herramientas que contiene los botones de **Agregar** y **Eliminar** elementos

Documento de Word 2007

La solución generada, protege la integridad del Esquema XML, para que no pueda ser modificado de manera involuntaria o por un error del usuario.

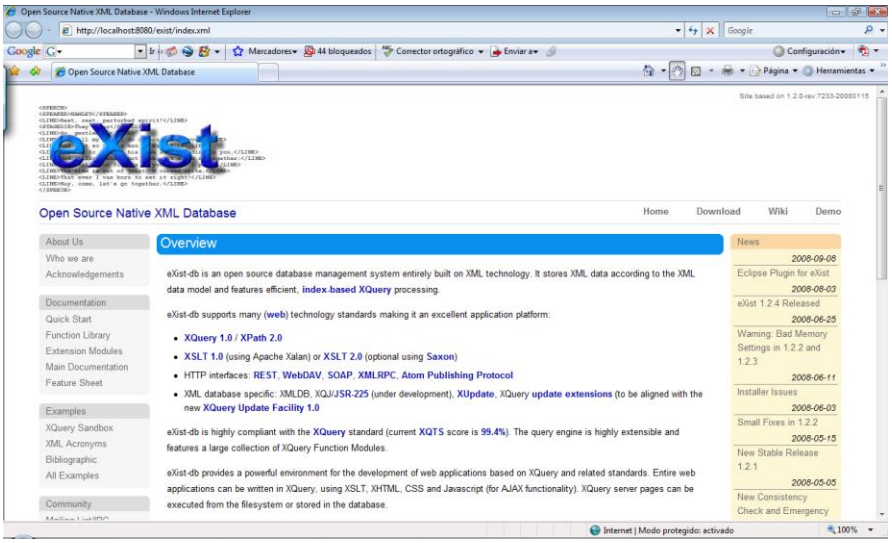
Independientemente de las soluciones en Microsoft Word, se realizó otra aplicación, utilizando el sistema Adobe Designer 7.0, para realizar una versión que permita generar documentos en formato PDF.

Pantalla de Adobe Designer 7.0



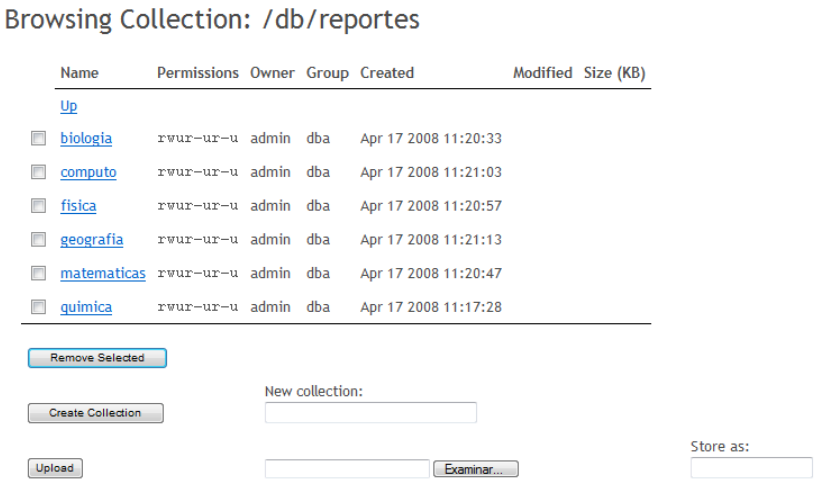
Documento de Adobe Designer en desarrollo

Para el almacenamiento de los Documentos XML, se diseño una Base de Datos en **eXist** para facilitar su administración, dicho manejador posee una interfaz de usuario y de administrador que permite manipular los documentos XML de manera fácil y amigable.



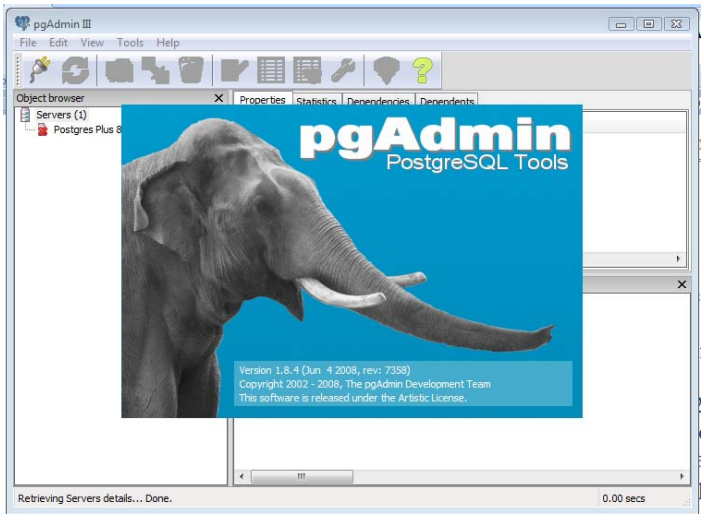
Pantalla de inicio de eXist

La Base de Datos diseñada, se encuentra organizada en colecciones por rama o materia de investigación, y se pueden generar las colecciones que sean necesarias y de una manera muy sencilla cargar un documento XML.



Colecciones de la Base de Datos

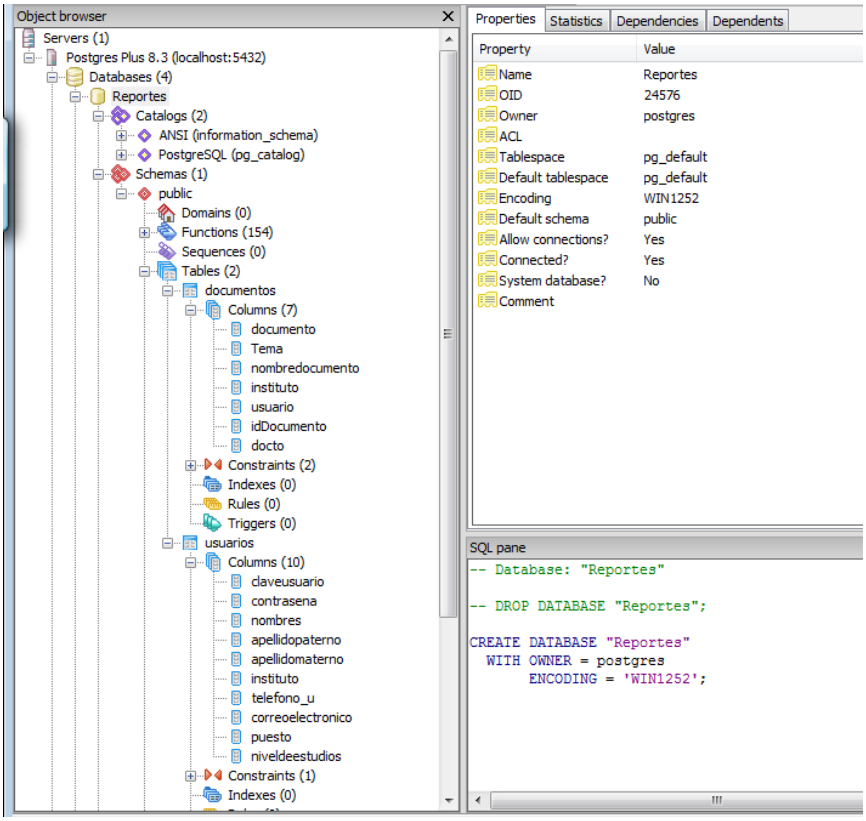
De igual manera se diseñó una base de datos en el SGBD Postgres, para almacenar los Documentos XML.

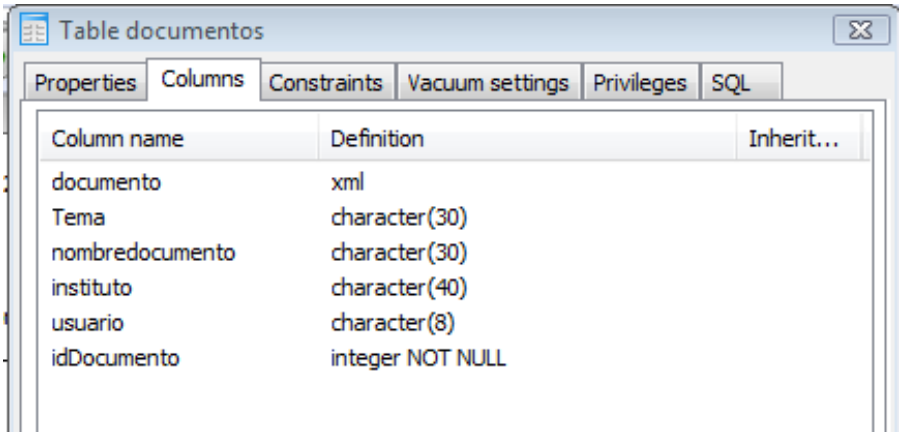


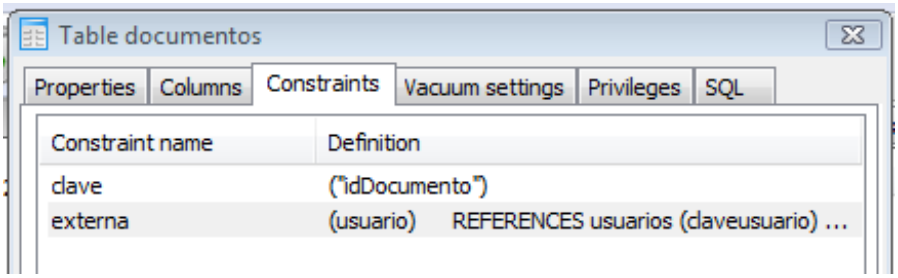
Pantalla de PostgreSQL

Diseño de la base de datos

Se tienen dos tablas: documentos y usuarios, a continuación se muestran las estructuras en el sistema Postgres.

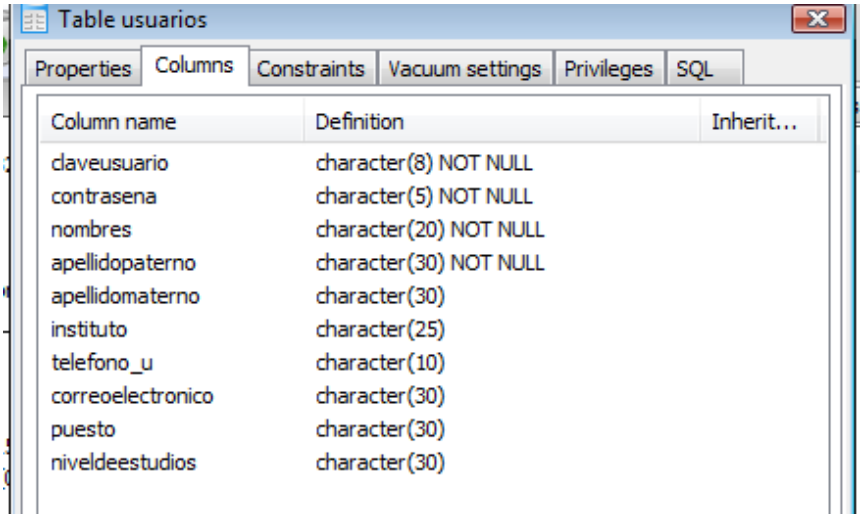


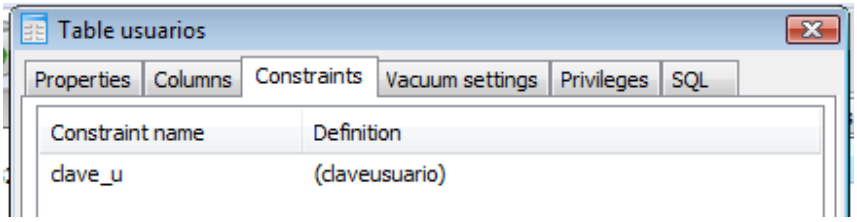




SQL pane

```
-- Table: documentos

-- DROP TABLE documentos;

CREATE TABLE documentos
(
  documento xml,
  "Tema" character(30),
  nombredocumento character(30),
  instituto character(40),
  usuario character(8),
  "idDocumento" integer NOT NULL,
  docto oid,
  CONSTRAINT clave PRIMARY KEY ("idDocumento"),
  CONSTRAINT externa FOREIGN KEY (usuario)
      REFERENCES usuarios (claveusuario) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION
)
WITH (OIDS=FALSE);
ALTER TABLE documentos OWNER TO postgres;
```





SQL pane

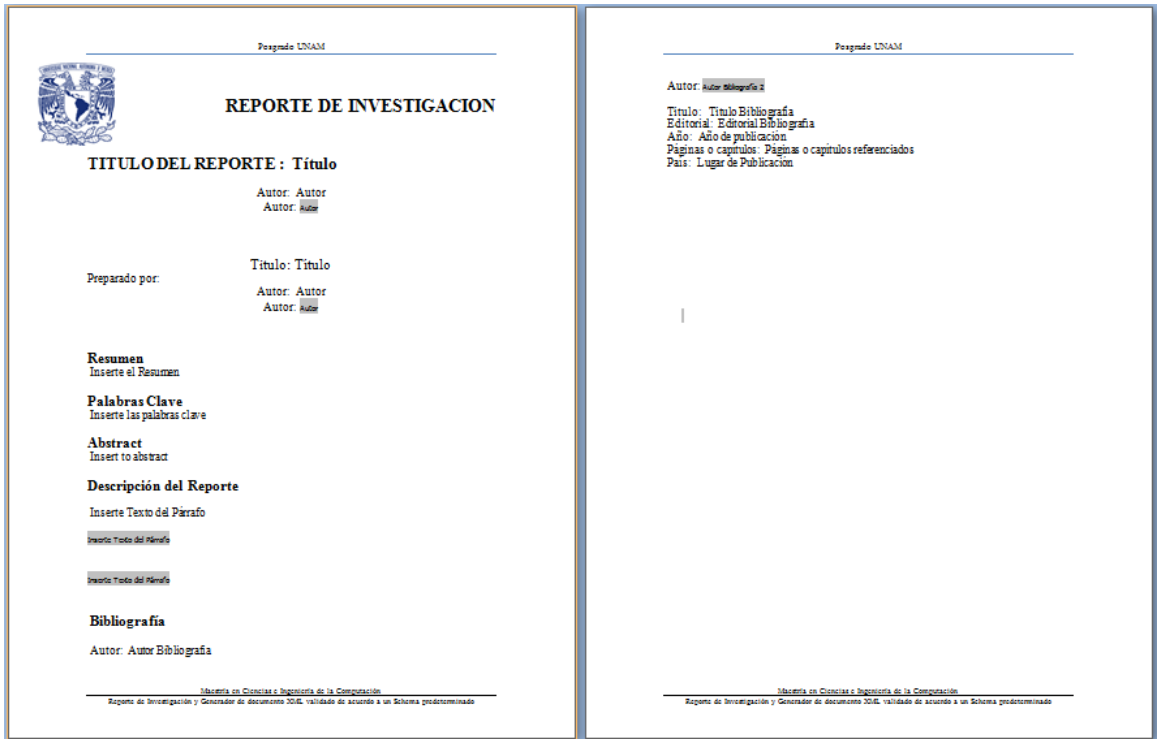
```
-- Table: usuarios

-- DROP TABLE usuarios;

CREATE TABLE usuarios
(
  claveusuario character(8) NOT NULL,
  contrasena character(5) NOT NULL,
  nombres character(20) NOT NULL,
  apellidopaterno character(30) NOT NULL,
  apellidomaterno character(30),
  instituto character(25),
  telefono_u character(10),
  correoelectronico character(30),
  puesto character(30),
  niveldeestudios character(30),
  CONSTRAINT clave_u PRIMARY KEY (claveusuario)
)
WITH (OIDS=FALSE);
ALTER TABLE usuarios OWNER TO postgres;
```

Apéndice C

Documentos diseñados

Word 2007

El documento para Microsoft Word 2007 tiene una estructura como la siguiente.



Posgrado UNAM

REPORTE DE INVESTIGACION

TITULO DEL REPORTE : Título

Autor: Autor
Autor: Autor

Titulo: Título

Preparado por:

Autor: Autor
Autor: Autor

Resumen
Inserte el Resumen

Palabras Clave
Inserte las palabras clave

Abstract
Insert to abstract

Descripción del Reporte

Inserte Texto del Párrafo

Inserte Texto del Párrafo

Inserte Texto del Párrafo

Bibliografía

Autor: Autor Bibliografia

Maestría en Ciencias e Ingeniería de la Computación
Reporte de Investigación y Generador de documento XML validado de acuerdo a un Schema predeterminado

Posgrado UNAM

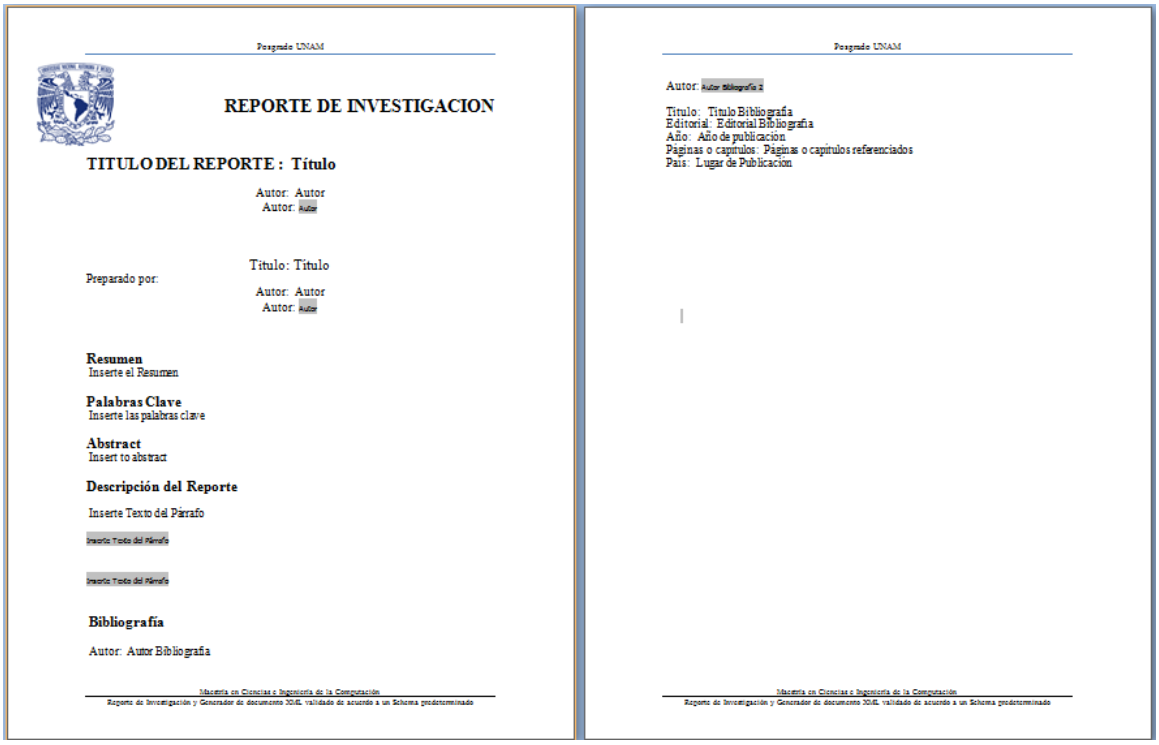
Autor: Autor Bibliografia 2

Titulo: Titulo Bibliografia
Editorial: Editorial Bibliografia
Año: Año de publicacion
Páginas o capitulos: Páginas o capitulos referenciados
País: Lugar de Publicacion

Maestría en Ciencias e Ingeniería de la Computación
Reporte de Investigación y Generador de documento XML validado de acuerdo a un Schema predeterminado

Word 2003

El documento para Microsoft Word 2003 tiene una estructura idéntica a la de la versión 2007, como se muestra a continuación, el cambio sólo es en las herramientas que proporciona la versión utilizada.



Posgrado UNAM

REPORTE DE INVESTIGACION

TITULO DEL REPORTE : Título

Autor: Autor
Autor: Autor

Titulo: Título

Preparado por:

Autor: Autor
Autor: Autor

Resumen
Inserte el Resumen

Palabras Clave
Inserte las palabras clave

Abstract
Insert to abstract

Descripción del Reporte

Inserte Texto del Párrafo

Inserte Texto del Párrafo

Inserte Texto del Párrafo

Bibliografía

Autor: Autor Bibliografia

Maestría en Ciencias e Ingeniería de la Computación
Reporte de Investigación y Generador de documento XML validado de acuerdo a un Schema predeterminado

Posgrado UNAM

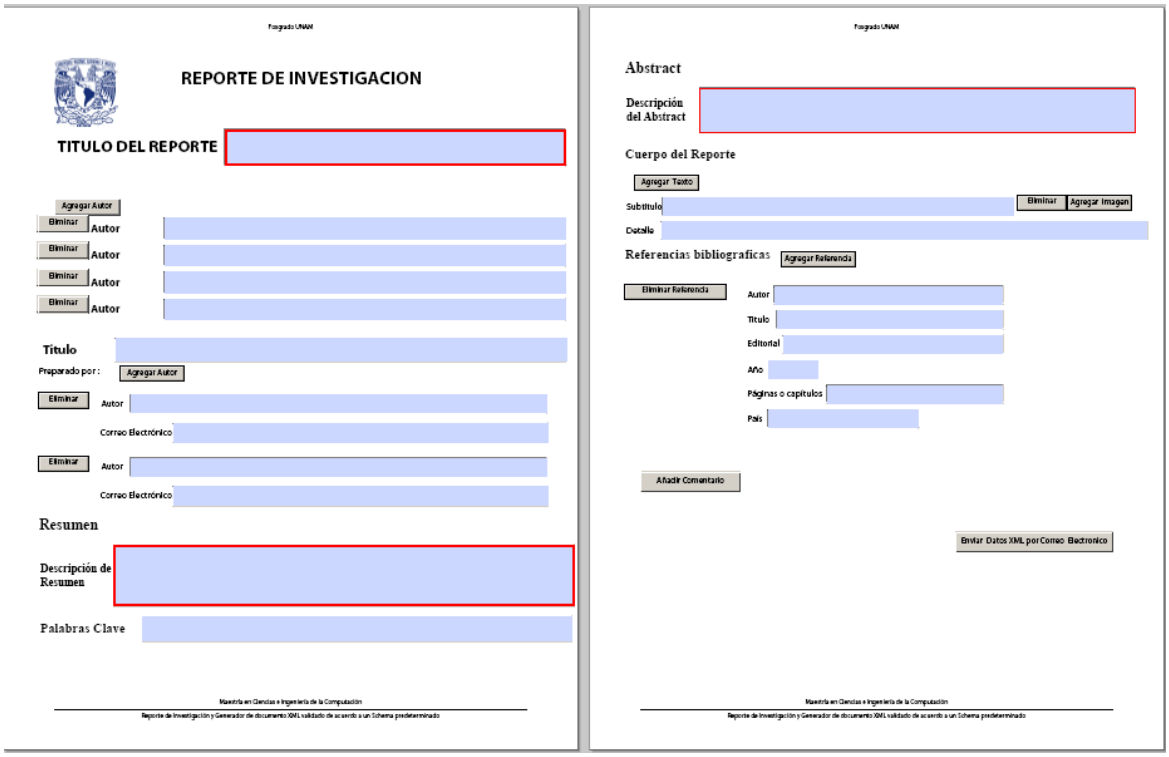
Autor: Autor Bibliografia 2

Titulo: Titulo Bibliografia
Editorial: Editorial Bibliografia
Año: Año de publicacion
Paginas o capitulos: Paginas o capitulos referenciados
Pais: Lugar de Publicacion

Maestría en Ciencias e Ingeniería de la Computación
Reporte de Investigación y Generador de documento XML validado de acuerdo a un Schema predeterminado

Adobe Acrobat

El documento para Adobe Acrobat posee la siguiente estructura.



Es importante mencionar que los tres archivos pueden generar documentos XML de acuerdo al esquema predeterminado.

Apéndice D

Código Fuente

La programación se llevó a cabo en Visual Studio 2008, Adobe Designer 7.0 y en NetBeans 5.5, por lo tanto se tiene código fuente en Visual Basic, C#, JavaScript, JAVA y XML.

Todo el código esta implícito en el documento generado.

A continuación se muestran algunos fragmentos de la programación.

```
' Programación para Microsoft Word
Imports System.Collections
Imports System.Object
Imports Microsoft.VisualStudio.Tools.Applications.Runtime

Public Class ThisDocument
    Dim herra As New PanelXML

    Friend WithEvents showCheck As System.Windows.Forms.CheckBox


    Private Sub ThisDocument_Startup(ByVal sender As Object, ByVal e As System.EventArgs)
Handles Me.Startup
        Me.ActionsPane.Controls.Add(herra)

    End Sub

    Private Sub ThisDocument_Shutdown(ByVal sender As Object, ByVal e As System.EventArgs)
Handles Me.Shutdown

    End Sub

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
        Dim name As String = "Insertar Autor"

        InsertControlAtSelection(New Button(), name)

    End Sub

    Private Shared Sub InsertControlAtSelection(ByVal c As Control, ByVal name As String)

        Dim selection As Word.Range = SelectedRange

        c.Name = name
        If selection IsNot Nothing Then
            ' Guardar el extensor del control como una etiqueta en el control para que
            ' podamos tener acceso a las propiedades extendidas del control cuando las
necesitemos.
            c.Tag = Globals.ThisDocument.Controls.AddControl(c, _
                selection, _
                Globals.ThisDocument.Application.PixelsToPoints(c.Width, False), _
                Globals.ThisDocument.Application.PixelsToPoints(c.Height, True), _
                name)
        End If
    End Sub

    Private Shared ReadOnly Property SelectedRange() As Word.Range

        Get

            Dim selection As Word.Selection = Globals.ThisDocument.Application.Selection
```

```
            If selection IsNot Nothing AndAlso selection.Document Is
Globals.ThisDocument.InnerObject Then
                Return selection.Range
            End If

            Return Nothing
        End Get
    End Property

    Private Sub btnAgregaAutor2_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs)
        'Dim instexto As Word.Range = Me.Application.ActiveDocument.Range(Start:=0, End:=0)
        Dim instexto As Word.Range = SelectedRange
        instexto.Text = " Autor : correo electronico "

    End Sub


    'Private Sub TEI2Node_ContextEnter(ByVal sender As Object, _
    ' ByVal e As Microsoft.Office.Tools.Word.ContextChangeEventArgs) _
    'Handles TEI2Node.ContextEnter()

    'If showAll.showCheck.Checked = False Then
    '        Me.ActionsPane.Controls.Add(addText)
    'Me.ActionsPane.Controls.Remove(showProperties)
    'End If
    '    End Sub


    Private Sub AgregaAutor(ByVal document As Word.Document)
        Dim xmlString As String = _
            "<?xml version=""1.0"" encoding=""utf-8"" ?>" & _
                "<TEI2 xmlns=""C:\Miguel\Maestria\08-II\Tesis\Interfaz"">" & _
                    "<publicationStmt>" & _
                        "<author> </name>" & _
                    "</publicationStmt>" & _
                "</TEI2>"

        Dim AutorXMLPart As Office.CustomXMLPart = _
            document.CustomXMLParts.Add(xmlString)
    End Sub

    Private Sub btnAgreAutor_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnAgreAutor.Click
        AgregaAutor(ThisDocument)
    End Sub

End Class
```

Fragmento de código en Adobe Designer

```
<?xml version="1.0" encoding="UTF-8"?>
<?xfa generator="AdobeDesigner_V7.0" APIVersion="2.2.4330.0"?>
<xdp:xdp xmlns:xdp="http://ns.adobe.com/xdp/">
<template xmlns="http://www.xfa.org/schema/xfa-template/2.2/">
  <subform layout="tb" locale="ambient" name="TEI2">
    <pageSet name="Formato">
      <pageArea id="Page1" name="Pagina">
        <draw minH="0mm" name="StaticText1" w="25.4mm" x="3.75in" y="0.25in">
          <ui>
            <textEdit>
            </textEdit>
          </ui>
          <value>
            <text>Posgrado UNAM</text>
          </value>
          <font size="7pt" typeface="Myriad Pro"/>
```

```
            <para hAlign="center" vAlign="middle"/>
          </draw>
          <contentArea h="238.76mm" w="184.15mm" x="12.7mm" y="19.05mm"/>
          <medium long="11in" short="8.5in" stock="letter"/>
          <?templateDesigner expand 1?>
          <draw minH="0.9906mm" name="StaticText2" w="177.8mm" x="19.05mm" y="260.35mm">
            <ui>
              <textEdit>
              </textEdit>
            </ui>
            <value>
              <text>Maestría en Ciencias e Ingeniería de la Computación</text>
            </value>
            <font baselineShift="0pt" size="7pt" typeface="Myriad Pro"/>
            <para hAlign="center" marginLeft="0pt" marginRight="0pt" spaceAbove="0pt" spaceBelow="0pt"
textIndent="0pt"/>
            <margin bottomInset="0.9906mm" leftInset="0mm" rightInset="0mm" topInset="0mm"/>
            <border>
              <edge presence="hidden"/>
              <edge presence="hidden"/>
              <edge/>
              <edge presence="hidden"/>
              <corner presence="hidden"/>
              <corner presence="hidden"/>
              <corner/>
              <corner presence="hidden"/>
            </border>
          </draw>
          <draw minH="0mm" name="StaticText3" w="177.8mm" x="19.05mm" y="265.43mm">
            <ui>
              <textEdit>
              </textEdit>
            </ui>
            <value>
              <text>Reporte de Investigación y Generador de documento XML validado de acuerdo a un Schema
predeterminado</text>
            </value>
            <font baselineShift="0pt" size="7pt" typeface="Myriad Pro"/>
            <para hAlign="center" marginLeft="0pt" marginRight="0pt" spaceAbove="0pt" spaceBelow="0pt"
textIndent="0pt"/>
          </draw>
        </pageArea>
        <?templateDesigner expand 1?></pageSet>
      <subform layout="tb" name="Reporte" w="200.55mm">
        <subform h="45.24mm" name="cabecera" w="200.025mm">
          <draw h="25.4mm" name="imgUNAM" w="25.4mm">
            <value>
```

Apéndice E

Diseño de Pruebas

Prueba de acuerdo a los casos de uso

Caso de uso: Entrar (al sistema)

Pruebas del Sistema

Caso de uso	Entrada	Esperado	Obtenido
Entrar	Dirección de la página	Dirección correcta	Despliega interfaz inicial
	Dirección de la página	Dirección incorrecta	El navegador no despliega interfaz inicial

Caso de uso: Administrar Documentos XML

Pruebas del Sistema

Caso de uso	Entrada	Esperado	Obtenido
Administrar Documentos XML	Agrega Documento XML	Opción Agrega Documento XML	Interfaz Agrega Documento XML
	Elimina Documento XML	Opción Elimina Documento XML	Interfaz Elimina Documento XML
	Consulta Documento XML	Opción Consulta Documento XML	Interfaz Consulta Documento XML
	Salir	Botón Salir	Cierre del Sistema

Caso de uso: Agrega Documento XML

Pruebas del Sistema

Caso de uso	Entrada	Esperado	Obtenido
Agrega Documento XML	Botón Aceptar	Datos completos y correctos del Documento XML	Guarda de Información en la Base de Datos
	Botón Aceptar	Datos incompletos y correctos del Documento XML	Mensaje de Error “Falta el dato _____, favor de capturarlo”

Caso de uso	Entrada	Esperado	Obtenido
	Botón Aceptar	Datos completos e incorrectos del Documento XML	Mensaje de Error "Documento XML no encontrado"
	Botón Aceptar	Documento XML duplicado	Presenta mensaje "Documento duplicado"
	Cancelar	Botón Cancelar	Regresa a la Interfaz del menú Documentos XML

Caso de uso: Elimina Documento XML

Pruebas del Sistema

Caso de uso	Entrada	Esperado	Obtenido
Elimina Documento XML	Botón Aceptar	Datos correctos del Documento XML	Elimina la información y el Documento XML de la Base de Datos
	Botón Aceptar	Datos incorrectos del Documento XML	Mensaje de Error "El Documento XML no existe, corrija por favor"
	Cancelar	Botón Cancelar	Regresa a la Interfaz del menú Documentos XML

Caso de uso: Consulta Documento XML

Pruebas del Sistema

Caso de uso	Entrada	Esperado	Obtenido
Consulta de Documento XML	Botón Aceptar	Datos correctos del Documento XML	Muestra información del Documento XML
	Botón Aceptar	Datos incorrectos del Documento XML	Mensaje de Error "El Documento XML no existe, corrija por favor"
	Cancelar	Botón Cancelar	Regresa a la Interfaz del menú Documentos XML

Pruebas para la Interfaz para generar los documentos XML

Se capturó información en los documentos de Microsoft Word y Adobe Acrobat, observando los resultados y comparándolos con documentos XML.

Una vez generados los documentos, estos fueron analizados con el programa XML Spy, para ver si estaban bien formados y eran validos.

Apéndice F

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
	<xs:import namespace="http://www.w3.org/XML/1998/namespace"/>
	<xs:element name="TEI2">
		<xs:complexType>
			<xs:sequence>
				<xs:element ref="teiHeader"/>
				<xs:element ref="text"/>
			</xs:sequence>
		</xs:complexType>
	</xs:element>
	<xs:element name="abbr">
		<xs:complexType mixed="true"/>
	</xs:element>
	<xs:element name="author">
		<xs:complexType mixed="true"/>
	</xs:element>
	<xs:element name="back">
		<xs:complexType>
			<xs:sequence>
				<xs:element ref="div1" minOccurs="0" maxOccurs="unbounded"/>
			</xs:sequence>
		</xs:complexType>
	</xs:element>
	<xs:element name="bibl">
		<xs:complexType mixed="true">
			<xs:choice minOccurs="0" maxOccurs="unbounded">
				<xs:element ref="author"/>
				<xs:element ref="biblScope"/>
				<xs:element ref="date"/>
				<xs:element ref="pubPlace"/>
				<xs:element ref="title"/>
				<xs:element ref="publisher"/>
				<xs:element ref="url"/>
			</xs:choice>
		</xs:complexType>
	</xs:element>
	<xs:element name="biblScope">
		<xs:complexType mixed="true"/>
	</xs:element>
	<xs:element name="body">
		<xs:complexType>
			<xs:choice minOccurs="0" maxOccurs="unbounded">
				<xs:element ref="div1"/>
				<xs:element ref="keywords"/>
				<xs:element ref="list"/>
				<xs:element ref="p"/>
			</xs:choice>
		</xs:complexType>
	</xs:element>
	<xs:element name="campo">
		<xs:complexType mixed="true"/>
	</xs:element>
	<xs:element name="cell">
		<xs:complexType mixed="true">
			<xs:choice minOccurs="0" maxOccurs="unbounded">
				<xs:element ref="p"/>
			</xs:choice>
```

```
                    <xs:attribute name="corresp" type="xs:IDREFS"/>
                    <xs:attribute name="next" type="xs:IDREF"/>
                    <xs:attribute name="prev" type="xs:IDREF"/>
                    <xs:attribute name="ana" type="xs:IDREFS"/>
                    <xs:attribute name="id" type="xs:ID"/>
                    <xs:attribute name="n" type="xs:anySimpleType"/>
                    <xs:attribute name="lang" type="xs:IDREF"/>
                    <xs:attribute name="rend" type="xs:anySimpleType"/>
                    <xs:attribute name="role" type="xs:anySimpleType" default="data"/>
                    <xs:attribute name="rows" type="xs:anySimpleType" default="1"/>
                    <xs:attribute name="cols" type="xs:anySimpleType" default="1"/>
                    <xs:attribute name="TEIform" type="xs:anySimpleType" default="cell"/>
            </xs:complexType>
    </xs:element>
    <xs:element name="date">
            <xs:complexType mixed="true"/>
    </xs:element>
    <xs:element name="publisher">
            <xs:complexType mixed="true"/>
    </xs:element>
    <xs:element name="def">
            <xs:complexType mixed="true"/>
    </xs:element>
    <xs:element name="div1">
            <xs:complexType mixed="true">
                    <xs:choice minOccurs="0" maxOccurs="unbounded">
                            <xs:element ref="def"/>
                            <xs:element ref="div2"/>
                            <xs:element ref="head"/>
                            <xs:element ref="listBibl"/>
                            <xs:element ref="listInfo"/>
                            <xs:element ref="note"/>
                            <xs:element ref="p"/>
                            <xs:element ref="table"/>
                            <xs:element ref="term"/>
                            <xs:element ref="list"/>
                            <xs:element ref="formula"/>
                            <xs:element ref="figure"/>
                    </xs:choice>
                    <xs:attribute name="n" type="xs:NMTOKEN"/>
                    <xs:attribute name="name" type="xs:anySimpleType"/>
                    <xs:attribute name="type" type="xs:NMTOKEN" use="required"/>
            </xs:complexType>
    </xs:element>
    <xs:element name="div2">
            <xs:complexType mixed="true">
                    <xs:choice minOccurs="0" maxOccurs="unbounded">
                            <xs:element ref="def"/>
                            <xs:element ref="div3"/>
                            <xs:element ref="head"/>
                            <xs:element ref="listBibl"/>
                            <xs:element ref="listInfo"/>
                            <xs:element ref="note"/>
                            <xs:element ref="p"/>
                            <xs:element ref="table"/>
                            <xs:element ref="term"/>
                            <xs:element ref="list"/>
                            <xs:element ref="formula"/>
                            <xs:element ref="figure"/>
                    </xs:choice>
                    <xs:attribute name="n" type="xs:NMTOKEN"/>
                    <xs:attribute name="name" type="xs:anySimpleType"/>
```

```
                <xs:attribute name="type" type="xs:NMTOKEN" use="required"/>
            </xs:complexType>
        </xs:element>
        <xs:element name="div3">
            <xs:complexType mixed="true">
                <xs:choice minOccurs="0" maxOccurs="unbounded">
                    <xs:element ref="def"/>
                    <xs:element ref="div4"/>
                    <xs:element ref="head"/>
                    <xs:element ref="listBibl"/>
                    <xs:element ref="listInfo"/>
                    <xs:element ref="note"/>
                    <xs:element ref="p"/>
                    <xs:element ref="table"/>
                    <xs:element ref="term"/>
                    <xs:element ref="list"/>
                    <xs:element ref="formula"/>
                    <xs:element ref="figure"/>
                </xs:choice>
                <xs:attribute name="n" type="xs:NMTOKEN"/>
                <xs:attribute name="name" type="xs:anySimpleType"/>
                <xs:attribute name="type" type="xs:NMTOKEN" use="required"/>
            </xs:complexType>
        </xs:element>
        <xs:element name="div4">
            <xs:complexType mixed="true">
                <xs:choice minOccurs="0" maxOccurs="unbounded">
                    <xs:element ref="def"/>
                    <xs:element ref="head"/>
                    <xs:element ref="listBibl"/>
                    <xs:element ref="listInfo"/>
                    <xs:element ref="note"/>
                    <xs:element ref="p"/>
                    <xs:element ref="table"/>
                    <xs:element ref="term"/>
                    <xs:element ref="list"/>
                    <xs:element ref="formula"/>
                    <xs:element ref="figure"/>
                </xs:choice>
                <xs:attribute name="n" type="xs:NMTOKEN"/>
                <xs:attribute name="name" type="xs:anySimpleType"/>
                <xs:attribute name="type" type="xs:NMTOKEN" use="required"/>
            </xs:complexType>
        </xs:element>
        <xs:element name="formula">
            <xs:complexType mixed="true">
                <xs:attribute name="corresp" type="xs:IDREFS"/>
                <xs:attribute name="next" type="xs:IDREF"/>
                <xs:attribute name="prev" type="xs:IDREF"/>
                <xs:attribute name="ana" type="xs:IDREFS"/>
                <xs:attribute name="id" type="xs:ID"/>
                <xs:attribute name="n" type="xs:anySimpleType"/>
                <xs:attribute name="lang" type="xs:IDREF"/>
                <xs:attribute name="rend" type="xs:anySimpleType"/>
                <xs:attribute name="notation" type="xs:anySimpleType" use="required"/>
                <xs:attribute name="TEIform" type="xs:anySimpleType" default="formula"/>
            </xs:complexType>
        </xs:element>
        <xs:element name="figure">
            <xs:complexType>
                <xs:sequence minOccurs="0">
                    <xs:element ref="head"/>
```

```
                        <xs:element ref="figDesc"/>
                    </xs:sequence>
                    <xs:attribute name="entity" type="xs:ENTITY"/>
                    <xs:attribute name="TEIform" type="xs:anySimpleType" default="formula"/>
                </xs:complexType>
            </xs:element>
            <xs:element name="figDesc">
                <xs:complexType mixed="true">
                    <xs:choice minOccurs="0" maxOccurs="unbounded">
                        <xs:element ref="num"/>
                        <xs:element ref="title"/>
                        <xs:element ref="figure"/>
                    </xs:choice>
                    <xs:attribute name="id" type="xs:ID"/>
                    <xs:attribute name="n" type="xs:anySimpleType"/>
                    <xs:attribute name="lang" type="xs:IDREF"/>
                    <xs:attribute name="rend" type="xs:anySimpleType"/>
                    <xs:attribute name="TEIform" type="xs:anySimpleType" default="figDesc"/>
                </xs:complexType>
            </xs:element>
            <xs:element name="list">
                <xs:complexType mixed="true">
                    <xs:choice minOccurs="0" maxOccurs="unbounded">
                        <xs:element ref="head"/>
                        <xs:element ref="label"/>
                        <xs:element ref="item"/>
                    </xs:choice>
                </xs:complexType>
            </xs:element>
            <xs:element name="label">
                <xs:complexType mixed="true"/>
            </xs:element>
            <xs:element name="item">
                <xs:complexType mixed="true">
                    <xs:choice minOccurs="0" maxOccurs="unbounded">
                        <xs:element ref="p"/>
                        <xs:element ref="list"/>
                    </xs:choice>
                </xs:complexType>
            </xs:element>
            <xs:element name="docAuthor">
                <xs:complexType mixed="true">
                    <xs:choice minOccurs="0" maxOccurs="unbounded">
                        <xs:element ref="note"/>
                    </xs:choice>
                </xs:complexType>
            </xs:element>
            <xs:element name="docDate">
                <xs:complexType mixed="true"/>
            </xs:element>
            <xs:element name="docOwner">
                <xs:complexType mixed="true"/>
            </xs:element>
            <xs:element name="docTitle">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element ref="titlePart" maxOccurs="unbounded"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
            <xs:element name="fileDesc">
                <xs:complexType>
```

```
                    <xs:sequence>
                        <xs:element ref="titleStmt"/>
                        <xs:element ref="publicationStmt"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
            <xs:element name="for">
                <xs:complexType mixed="true">
                    <xs:choice minOccurs="0" maxOccurs="unbounded">
                        <xs:element ref="docOwner"/>
                    </xs:choice>
                </xs:complexType>
            </xs:element>
            <xs:element name="from">
                <xs:complexType mixed="true">
                    <xs:choice minOccurs="0" maxOccurs="unbounded">
                        <xs:element ref="docAuthor"/>
                    </xs:choice>
                </xs:complexType>
            </xs:element>
            <xs:element name="front">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element ref="titlePage"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
            <xs:element name="head">
                <xs:complexType mixed="true">
                    <xs:choice minOccurs="0" maxOccurs="unbounded">
                        <xs:element ref="p"/>
                    </xs:choice>
                    <xs:attribute name="rend" type="xs:NMTOKEN"/>
                    <xs:attribute name="type" type="xs:NMTOKEN"/>
                </xs:complexType>
            </xs:element>
            <xs:element name="info">
                <xs:complexType mixed="true">
                    <xs:choice minOccurs="0" maxOccurs="unbounded">
                        <xs:element ref="campo"/>
                        <xs:element ref="marc"/>
                        <xs:element ref="valor"/>
                    </xs:choice>
                </xs:complexType>
            </xs:element>
            <xs:element name="keywords">
                <xs:complexType mixed="true"/>
            </xs:element>
            <xs:element name="listBibl">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element ref="bibl" maxOccurs="unbounded"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
            <xs:element name="listInfo">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element ref="info" maxOccurs="unbounded"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
```

```
<xs:element name="marc">
        <xs:complexType mixed="true"/>
</xs:element>
<xs:element name="note">
        <xs:complexType mixed="true">
                <xs:choice minOccurs="0" maxOccurs="unbounded">
                        <xs:element ref="p"/>
                </xs:choice>
                <xs:attribute name="corresp" type="xs:IDREFS"/>
                <xs:attribute name="next" type="xs:IDREF"/>
                <xs:attribute name="prev" type="xs:IDREF"/>
                <xs:attribute name="ana" type="xs:IDREFS"/>
                <xs:attribute name="id" type="xs:ID"/>
                <xs:attribute name="n" type="xs:anySimpleType"/>
                <xs:attribute name="lang" type="xs:IDREF"/>
                <xs:attribute name="rend" type="xs:anySimpleType"/>
                <xs:attribute name="type" type="xs:anySimpleType"/>
                <xs:attribute name="resp" type="xs:anySimpleType"/>
                <xs:attribute name="place" type="xs:anySimpleType" default="unspecified"/>
                <xs:attribute name="anchored" default="yes">
                        <xs:simpleType>
                                <xs:restriction base="xs:NMTOKEN">
                                        <xs:enumeration value="yes"/>
                                        <xs:enumeration value="no"/>
                                </xs:restriction>
                        </xs:simpleType>
                </xs:attribute>
                <xs:attribute name="target" type="xs:IDREFS"/>
                <xs:attribute name="targetEnd" type="xs:IDREFS"/>
                <xs:attribute name="TEIform" type="xs:anySimpleType" default="note"/>
        </xs:complexType>
</xs:element>
<xs:element name="num">
        <xs:complexType mixed="true">
                <xs:attribute name="type" type="xs:NMTOKEN" use="required"/>
        </xs:complexType>
</xs:element>
<xs:element name="p">
        <xs:complexType mixed="true">
                <xs:choice minOccurs="0" maxOccurs="unbounded">
                        <xs:element ref="abbr"/>
                        <xs:element ref="date"/>
                        <xs:element ref="num"/>
                        <xs:element ref="note"/>
                </xs:choice>
        </xs:complexType>
</xs:element>
<xs:element name="publicationStmt">
        <xs:complexType>
                <xs:sequence>
                        <xs:element ref="author" maxOccurs="unbounded"/>
                </xs:sequence>
        </xs:complexType>
</xs:element>
<xs:element name="pubPlace">
        <xs:complexType mixed="true">
                <xs:choice minOccurs="0" maxOccurs="unbounded">
                        <xs:element ref="biblScope"/>
                </xs:choice>
        </xs:complexType>
</xs:element>
<xs:element name="row">
```

```
            <xs:complexType>
                  <xs:sequence>
                        <xs:element ref="cell" maxOccurs="unbounded"/>
                  </xs:sequence>
            </xs:complexType>
      </xs:element>
      <xs:element name="table">
            <xs:complexType>
                  <xs:sequence>
                        <xs:element ref="head"/>
                        <xs:element ref="row" maxOccurs="unbounded"/>
                  </xs:sequence>
                  <xs:attribute name="cols" type="xs:NMTOKEN"/>
                  <xs:attribute name="rend" type="xs:NMTOKEN" use="required"/>
                  <xs:attribute name="rows" type="xs:NMTOKEN"/>
            </xs:complexType>
      </xs:element>
      <xs:element name="teiHeader">
            <xs:complexType>
                  <xs:sequence>
                        <xs:element ref="fileDesc"/>
                  </xs:sequence>
            </xs:complexType>
      </xs:element>
      <xs:element name="term">
            <xs:complexType mixed="true"/>
      </xs:element>
      <xs:element name="text">
            <xs:complexType>
                  <xs:sequence>
                        <xs:element ref="front"/>
                        <xs:element ref="body"/>
                        <xs:element ref="back"/>
                  </xs:sequence>
            </xs:complexType>
      </xs:element>
      <xs:element name="title">
            <xs:complexType mixed="true"/>
      </xs:element>
      <xs:element name="titlePage">
            <xs:complexType mixed="true">
                  <xs:choice minOccurs="0" maxOccurs="unbounded">
                        <xs:element ref="docDate"/>
                        <xs:element ref="docTitle"/>
                        <xs:element ref="for"/>
                        <xs:element ref="from"/>
                  </xs:choice>
            </xs:complexType>
      </xs:element>
      <xs:element name="titlePart">
            <xs:complexType mixed="true">
                  <xs:choice minOccurs="0" maxOccurs="unbounded">
                        <xs:element ref="note"/>
                  </xs:choice>
                  <xs:attribute name="type" type="xs:NMTOKEN" use="required"/>
            </xs:complexType>
      </xs:element>
      <xs:element name="titleStmt">
            <xs:complexType>
                  <xs:sequence>
                        <xs:element ref="title"/>
                  </xs:sequence>
```

```
                </xs:complexType>
            </xs:element>
            <xs:element name="url">
                <xs:complexType mixed="true"/>
            </xs:element>
            <xs:element name="valor">
                <xs:complexType mixed="true"/>
            </xs:element>
            <xs:notation name="png" public="-//TEI//NOTATION IETF RFC2083 Portable Network Graphics//EN"/>
            <xs:notation name="jpeg" public="ISO DIS 10918//NOTATION JPEG Graphics Format//EN"/>
</xs:schema>
```

Apéndice G

A

AJAX (Asynchronous JavaScript And XML) Técnica para el desarrollo de aplicaciones web interactivas que hace uso de JavaScript asíncrono más XML.

APA (American Psychological Association) es una organización científica y profesional de psicólogos.

API Interfaz de Programación de Aplicaciones conjunto de funciones y procedimientos que ofrece una biblioteca para ser utilizado por otro programa.

B

BD Base de Datos.

BLOB Binary Large Object, son elementos utilizados en las bases de datos, para almacenar datos de gran tamaño y que cambian de manera dinámica.

C

C# Lenguaje de programación derivado de C/C++ y Java.

C/C++ Lenguaje de programación.

CDATA Tipo de Dato especificado en XML.

CHICAGO Norma de descripción formal para documentos.

CIMI (Computer Interchange of Museum Information), iniciativa que se utiliza en TEI para la información de museos digitales.

CML Lenguaje de marcado en XML para fórmulas químicas.

CSS (Hojas de Estilo en Cascada), es un lenguaje para definir la presentación de documentos XML.

D

DC Esquema Dublin Core, que se utiliza en la indización para preparar la interoperabilidad semántica en la web.

DOM (Modelo de Objetos de Documento) API para el acceso y edición dinámica del contenido de documentos XML.

DTD (Definición de Tipos de Documentos) documento que sirve para definir la estructura y sintaxis de documentos XML.

E

EAD (Encoded Archival Description), esquema para la información archivística.

eXist Base de datos nativa de XML.

F

FGDC (Federal Geographic Data Committee), iniciativa que se utiliza en TEI para la información de digital geoespacial.

H

HARVARD Norma de descripción formal para documentos.

HTML (Lenguaje de Marcado de Hipertexto), lenguaje de marcado predominante para la construcción de paginas web.

I

IMPLIED Atributo que específica que un dato puede ser opcional, en un documento XML.

ISO (Organización Internacional para la Estandarización), organismo encargado de promover normas internacionales.

J

JAVA Lenguaje de programación orientado a objetos.

JavaScript Lenguaje de programación orientado a objetos, utilizado principalmente para el desarrollo de páginas web.

M

MathML Lenguaje de Marcas para fórmulas matemáticas.

O

OPEN SOURCE (Código Abierto), software distribuido y desarrollado libremente.

ORDBMS Sistema de Gestión de Bases de Datos Objeto-Relacionales.

P

Pascal Lenguaje de programación estructurado.

PCDATA Tipo de Dato especificado en XML.

PDF (Formato Portátil de Documento), formato de almacenamiento de documentos, creado por Adobe Systems.

Perl Lenguaje de programación basado en un estilo de bloques.

PHP Lenguaje de programación, diseñado para la creación de páginas web dinámicas.

PL/pgSQL Lenguaje imperativo utilizado en PostgreSQL.

Postgres Manejador de Bases de Datos relacional orientado a objetos de software libre.

PostgreSQL Manejador de Bases de Datos relacional orientado a objetos de software libre, es la siguiente versión de Postgres.

PostScript Lenguaje de descripción de página utilizado en impresoras y como formato de transporte de archivos gráficos.

Python Lenguaje de programación que permite dividir el programa en módulos reutilizables.

R

RDF (Marco de Descripción de Recursos) Framework para metadatos en la WWW.

REQUIRED Atributo que específica que un dato es obligatorio, en un documento XML.

S

SAX (Simple API para XML), API para utilizar XML en Java.

Schema (Lenguaje de esquema XML), documento que sirve para definir la estructura y restricciones de contenido para documentos XML.

SMBD Sistema Manejador de Bases de Datos.

T

TEI (Text Encoding Initiative), esquema XML que surge como un proyecto para el área de humanidades.

U

URL (Localizador Uniforme de Recursos), secuencia de caracteres, que se usa para localizar recursos en internet.

V

Visual Basic Lenguaje de programación, basado en Basic, con un entorno de desarrollo integrado.

W

W3C (Consorcio World Wide Web), consorcio internacional que produce estándares para la Web.

WAL (Write Ahead Logging), técnica utilizada en PostgreSQL, para el registro de transacciones de registros.

WML Lenguaje de Marcas para equipos inalámbricos.

X

XHTML (Lenguaje eXtensible de Marcado de Hipertexto), lenguaje de marcado pensado para sustituir al lenguaje HTML.

XLink (Lenguaje de vínculos XML), que permite crear relaciones cruzadas entre documentos.

XML (Lenguaje eXtensible de Marcado), lenguaje creado por la W3C, es una simplificación y adaptación del lenguaje SGML.

XPath (XML Path Language), Lenguaje que permite recorrer y procesar documentos XML.

XPointer (Lenguaje de Punteros XML), lenguaje que proporciona una manera de identificar fragmentos de un documento XML.

XQuery (Lenguaje de Consulta XML), lenguaje diseñado para realizar consultas en colecciones de datos XML.

XSL (Lenguaje eXtensible de Hojas de Estilo), lenguaje basado en XML, que permite describir como la información de un documento XML, deberá ser formateada para su presentación.

XSLT (Transformaciones XSL), forma estándar para transformar documentos XML, en otros documentos o a otros formatos.

XUpdate (Lenguaje de Actualización XML), lenguaje de actualización de bases de datos XML.