**UNIVERSIDAD POLITÉCNICA DE PUEBLA**

**Programa Académico de Ingeniería en Informática**

# "Mecanismo de navegación para colecciones de documentos digitales"

Reporte Técnico NUMERO

Miriam Cruz Guzmán

Juan C. Bonilla, Puebla. Diciembre 2010

**UNIVERSIDAD POLITÉCNICA DE PUEBLA**

**Programa Académico de Ingeniería en Informática**

# "Mecanismo de navegación para colecciones de documentos digitales"

Miriam Cruz Guzmán

Reporte Técnico 25-08-10

Comité Evaluador

Dra. María Auxilio Medina Nieto
Asesor

MC. Argelia Berenice Urbina Nájera
Sinodal

MC. Rebeca Rodríguez Huesca
Sinodal

Juan C. Bonilla, Puebla. Diciembre 2010

# Índice

# Resumen

Este proyecto describe un mecanismo para explorar colecciones digitales de documentos representadas en estructuras jerárquicas denominadas *ontologías de registros*. Las ontologías se representan como archivos XML que se construyen mediante el procesamiento de registros del protocolo OAI-PMH y la aplicación de un algoritmo de agrupamiento. El mecanismo está compuesto por una interfaz web y gráficas tipo SVG. A través de la interfaz, los usuarios pueden consultar los datos bibliográficos de los documentos y la organización de la colección a la que pertenecen. Este documento contiene conceptos de la web semántica, así como la metodología, los detalles de implementación, la aplicación del mecanismo para visualizar CORTUPP, la Colección de Reportes Técnicos desarrollados en la carrera de Ingeniería en Informática de la Universidad Politécnica de Puebla, y propone el trabajo a futuro.

# Capítulo 1

# Planteamiento del problema de investigación

Este capítulo contiene las secciones siguientes: introducción, objetivo general, objetivos específicos, justificación, cronograma de actividades, alcances y limitaciones, recursos de hardware y software.

## 1.1. Introducción

El concepto de biblioteca digital ha sido empleado para definir un gran almacén de información digital accesible a través de las computadoras. Al igual que una biblioteca tradicional, una biblioteca digital sirve como un archivo de conocimientos que cubre una gran variedad de temas. De la misma manera que un noticiero, una biblioteca digital proporciona información que cambia rápidamente.

Para construir una biblioteca digital es preciso tomar en cuenta una diversidad de elementos que se interrelacionen para brindar un servicio eficiente y transparente al usuario, con el único objetivo (como el de toda tecnología[1] ) de facilitar la vida del hombre.

[1] `http://es.tldp.org/I+D/donantonio/doc-`*ers¡eee*830*-donantonio-iusuario/*`doc-`*ers¡eee*830*-donantonio-iusuario.pdf*

Actualmente la Universidad Politécnica de Puebla (UPPuebla) aún no cuenta con una biblioteca digital, sin embargo, existe un proyecto paralelo a ese fin. En un futuro, podría contener documentos de diversos tipos organizados en colecciones. Por ello, se propone desarrollar un mecanismo basado en ontologías el cual agrupe documentos. El mecanismo propuesto facilitará la navegación de la ontología de documentos de manera que apoyará al usuario para encontrar y explorar información de su interés.

## 1.2. Objetivo general

Diseñar un mecanismo de navegación por contenido para colecciones de documentos digitales

## 1.3. Objetivos específicos

- Mostrar de forma gráfica la organización de una colección de documentos digitales
- Construir una interfaz que visualize similitud por contenido entre documentos
- Recuperar los datos bibliográficos de cada uno de los documentos representados por el mecanismo.

## 1.4. Justitificación

La UPPuebla es una universidad de reciente creación, en la cual se propone el desarrollo de diversos sistemas que automaticen los procesos que maneja en cada una de sus áreas. Un ejemplo de ello es la propuesta de creación de una biblioteca digital, la cual podría concentrar los proyectos de investigación realizados por alumnos bajo la supervisión de algunos profesores. Estos proyectos se consideran trabajos terminales y

se documentan en reportes técnicos. Los reportes técnicos se editan en una plantilla de Latex.

En general, los servicios que se ofrecen en una biblioteca digital tradicional deben ser eficientes para localizar y acceder a información relevante, generalmente almacenada en documentos digitales. El número de documentos digitales se incrementa con frecuencia, por ello surge la necesidad de integrar mecanismos de navegación por contenido que faciliten la búsqueda de documentos a los usuarios. En una biblioteca digital, los servicios pueden extenderse a través del uso de tecnologías de información y comunicación.

El mecanismo de navegación que se propondrá en este proyecto permitirá que los usuarios puedan adentrarse a cada uno de los grupos de documentos (o clases). Las clases están organizadas jerárquicamente en una ontología, la cual está representada como un documento XML. Los detalles de implementación y representación de la ontología pueden consultarse en [8].

## 1.5. Recursos de hardware y software

Para el desarrollo e implementación de este proyecto, se requieren los recursos siguientes:

### 1.5.1. Recursos de hardware

- Memoria RAM 1G
- Procesador Pentium 4
- Disco duro de 80Gb

### 1.5.2. Recursos de software

- Plugin SVG *ScalableVectorGraphics* versión 1.7
- Lenguaje de programación Java (JDK versión 1.4)

## 1.6. Cronograma de actividades

Las tablas 1.1 y 1.2 muestran las actividades principales del proyecto. Estas actividades están asignadas semanalmente.

Tabla 1.1: Cronograma de actividades para Proyecto de Investigación I

| **Actividades** | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Asignación del proyecto de investigación | * | * | * | | | | | | | | | | | | | |
| Elaboración de la propuesta de investigación | | | | * | * | * | * | | | | | | | | | |
| Revisión de literatura | | | | | | * | * | * | * | | | | | | | |
| Elaboración de resumen de web semántica | | | | | | | | * | * | | | | | | | |
| Análisis del mecanismo de navegación start-fish | | | | | | | | | * | * | | | | | | |
| Extracción de características de navegación de UVA | | | | | | | | | | | * | * | | | | |
| Revisión del método OntoOAIR. | | | | | | | | | | | | * | * | | | |
| Mejoras del protocolo de investigación | | | | | | | | | | | | | | * | * | |
| Elaboracion del marco teórico | | | | | | | | | | | | | | | | |
| Presentación de protocolo de investigación | | | | | | | | | | | | | | | | * |

Tabla 1.2: Cronograma de actividades para Proyecto de Investigación II

| **Actividades** | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Elaboración de diseño | * | | | | | | | | | | | | | | |
| Incorporación de ajustes al diseño del sistema | | * | * | | | | | | | | | | | | |
| Notación del lenguaje SVG. | | | * | * | * | | | | | | | | | | |
| Generación del documento XML. | | | | | | * | | | | | | | | | |
| Codificacion en Java | | | | | | | * | * | * | * | * | | | | |
| Ejecución de pruebas | | | | | | | | | | | | * | | | |
| Elaboración de conclusiones | | | | | | | | | | | | | * | | |
| Presentación de reporte de investigación | | | | | | | | | | | | | | * | |

## 1.7. Alcances y limitaciones

Alcances

- El mecanismo de navegación deberá reducir el tiempo que invertiría un usuario en navegar desde la colección de reportes técnicos (CORTUPP [2] ) para conocer de forma general el tipo de documentos que contiene
- Todo alumno y profesor podrá hacer uso del mecanismo de navegación, dado que éste se almacenará en un servidor web
- La clasificación de los documentos se basa en el algoritmo FIHC [4]. Esta clasificación está representada en la ontología en lenguaje XML.

Limitaciones

- El número de documentos que incluye la ontología influye en la presentación del mecanismo de navegación

[2] CORTUPP está disponible en `http://informatica.uppuebla.edu.mx/~cortupp/index.html`

- El navegador que permite la visualización del mecanismo de navegación deberá soportar imágenes con extensión SVG

# Capítulo 2

# Marco teórico

Para la elaboración del proyecto se consideran diversos temas como web semántica, características de XML, conceptos y aplicaciones de ontologías. Estos temas junto con trabajos relacionados forman el contenido de este capítulo.

## 2.1. Web semántica

El concepto de web semántica nace de la visión de Tim Bernes-Lee quien identifica la necesidad de usar un lenguaje conveniente y legible para expresar los datos web a la computadora, de manera que se automaticen las transacciones. La web semántica propone el empleo de lenguajes como XML y Resource Description Framework (RDF), mapas conceptuales y otras tecnologías de metadatos que puedan hacer a los buscadores más poderosos y exitosos. Esta web permitirá a las personas encontrar lo que necesitan o desean ver [5].

Según el grupo *Web Agreement Group* (SWAG), la web semántica incluye documentos o porciones de documentos que describen explícitamente la relación entre cosas e información semántica intentando automatizar procesos[1] .

[1] Disponible en `http://swag.webns.net/whatIsSW`

[5] identifican los siguientes elementos de la web semántica:

- *XML*. XML es la base de la nueva generación de la web. Maneja URIs, espacios de nombre (*namespaces*) y tecnología en Unicode. Este lenguaje permite definir etiquetas propias
- *Uniform Resource Identifiers (URIs)*.- Referencían recursos arbitrarios en un único camino o ruta
- *Espacios de nombres*.- Identifican elementos únicos en un documento XML
- *Unicode*.- Describe recursos web en algún lenguaje natural
- *RDF schema*.- Provee datos para definir propiedades de dominio específico y clases
- *Mapas conceptuales*.- Documentos XML usados para registrar recursos web con el objetivo de que sean claros para la computadora y basados en temas, asociaciones y ocurrencias

La lógica y XML formarían la web semántica, donde agentes de software realizarían tareas sin intervención humana. Las ontologías son otra tecnología fundamental para implementar la web semántica, éstas se refieren a especificaciones explícitas de una conceptualización [7]. Este proyecto, propone que a partir de una ontología de documentos representada en XML, sea posible implementar el mecanismo de navegación.

### 2.1.1. Características de XML

HTML se creó para presentar la información en la web tomando en cuenta elementos como la distribución del espacio, estilos de fuente y párrafos, integrando objetos multimedia. El crecimiento de áreas como e-commerce propició la identificación de la

necesidad de un lenguaje que pueda ir más allá de las limitaciones de HTML, el cual además de ser entendible para los humanos, lo sea también para las computadoras y lo suficientemente flexible para describir áreas específicas de interés. SGML (de las siglas en inglés de *standard generalyzed markup lenguage*), fue utilizado en la definición HTML, la desventaja es que fue muy complicado y no satisfactorio para internet.

A principios de 1998 se desarrolló un nuevo metalenguaje, XML (*Extensible Markup Lenguage*), el cual se ha difundido hasta nuestros días. Un documento XML puede ser creado en cualquier editor de texto y guardado con extensión `.xml`, en el cual la primera línea incluirá la versión XML, el resto del documento está formado por etiquetas, (palabras o frases de soporte) que describen los datos que contienen.

La web semántica se construye con base en XML y sus tecnologías [5]. Esto no significa que reemplace a los documentos en HTML, sino que se complementan porque XML describe el significado de los datos.

XML permite a una organización o compañía crear sus propios dominios específicos. Es un lenguaje descriptivo de documentos creado para estructurar, almacenar y transportar información, permite definir etiquetas, el autor es quien domina la estructura del documento al definir las propias[2] . Otras ventajas de XML son:

- Se asignan atributos a las etiquetas
- Las etiquetas y atributos se definen de forma exacta mediante un esquema o un DTD (*Data Type Definition*)
- Está basado en texto, (no utiliza el formato binario), por lo que es relativamente fácil de convertir a otro formato
- XML es una industria estándar abierta definida por WWW (W3C), que es un

[2] http://www.w3schools.com/

vendedor de lenguaje independiente utilizado ampliamente por desarrolladores de software

El hecho de que XML permita crear lenguajes propios trae consigo la desventaja de realizar traducciones entre modelos de forma automática. Las ontologías se han utilizado para tratar de resolver este problema. El proceso de creación de ontologías inicia seleccionando las palabras y frases que podrán ser usadas en los nombres de las etiquetas. El siguiente paso es mostrar la relación entre éstas.

Además de las ontologías, se puede utilizar UML para representar objetos, propiedades y relaciones. El modelo UML puede convertirse en un XML schema, el cual en comparación con UML, contiene elementos adicionales no tan importantes para el entendimiento humano, pero sí para las computadoras, lo cual lo hace insuficiente en el contexto de la web semántica.

XML es un lenguaje que también se ha utilizado para describir ontologías, por ejemplo en [8] y [3]. La familia de lenguajes XML consta de cuatro grupos: XML, accesorios XML, traductores XML y aplicaciones XML[3] .

Java trabaja en conjunto con XML, ya que permite integrar XML Parser como una aplicación para generar documentos XML, al igual que Action Script. Existen otras tecnologías para mostrar documentos XML como HTML y Cascading Stile Sheets (CSS), convenientes para formatos simples, o XSL (Extensible Stylesheet Language) para formatos más complejos.

### 2.1.2. Aplicaciones de ontologías

Las ontologías son conceptos teóricos acerca de clases específicas de objetos, propiedades de objetos y relaciones entre objetos [1]. En filosofía, ontología es el estudio de tipos

[3] Disponible en `http://www.cs.jyu.fi/~airi/xmlfamily-20010806.html`

de cosas que existen, es una representación del vocabulario especializado en algún dominio o materia. En sistemas de información, una ontología es un conjunto de términos para representar hechos específicos en una instancia de un dominio. Otra visión sería en definirla como un conjunto general de hechos que son compartidos por alguna comunidad. Otras características de las ontologías son las siguientes:

- Es una representación de lenguaje
- Elimina ambigedades, es decir, los conceptos son únicos
- Aparecen como una taxonomía de árbol conceptual

Las ontologías son importantes porque describen planes y actividades, ya que requieren de especificación de palabras, objetos y relaciones. Son usadas en sistemas para recuperar información, por ejemplo, en bibliotecas digitales se utilizan como mecanismo de integración de fuentes de información heterogéneas en internet o para dirigir el proceso de búsqueda.

El dominio de ontologías llegará a ser tan importante en general en sistemas de software como en muchas áreas de inteligencia artificial [1]. CYC (enciclopedia, proyecto que intenta ensamblar una ontología), Wordnet y Sensus son ejemplos de ontologías compartidas que han sido usadas para entender lenguajes naturales.

Thomas R. Gruber ha propuesto un lenguaje llamado Ontoligua para ayudar a construir ontologías portables [7]. Existen otros lenguajes para definir ontologías que son más intuitivos para el ser humano como XML. Las características de XML se consideran apropiadas para el proyecto, el cual se basará en una ontología que contiene grupos de documentos descritos en XML. Estos documentos forman colecciones de la Iniciativa de Archivos Abiertos, la cual utiliza el formato de metadatos Dublin Core como el mecanismo más simple de interoperabilidad. La sección siguiente describe las ontologías utilizadas en este proyecto.

## 2.2. Ontología de registros

Una ontología de registros propone categorías y relaciones para representar la estructura de una colección de documentos, con el objetivo de agrupar documentos similares y facilitar el acceso a los datos. El DTD de la ontología de registros puede observarse en la Figura 2.2. La ontogía lleva por nombre `Ontologyofrecords`, está construida con base en un algoritmo de agrupamiento que requiere dos parámetros de entrada: `clustersupport` y `globalsupport`. Un grupo (`cluster`) reúne documentos similares, tiene una etiqueta, un nivel, cero o más grupos (`clusters`), cero o más documentos (`records`). Un documento tiene título (`title`), un tema (`subject`), un resumen (`description`), un identificador (`identifier`), un url, pertenece a una colección (`dataprovider`), se describe en un formato de metadatos (`metadataformat`), se libera en una fecha (`datestamp`) y se asocia con un autor (`author`). La Tabla 2.1 describe estos elementos y su aplicación a una colección de reportes de investigación.

Tabla 2.1: Descripción de los metadatos de una ontología de registros

| **Metadato** | **Descripción** |
|---|---|
| `label` | Etiqueta de un grupo |
| `level` | Nivel de un grupo |
| `title` | Título del proyecto de investigación |
| `subject` | Tema principal de la investigación |
| `description` | Resumen de la investigación |
| `identifier` | Identificador del reporte técnico que describe el proyecto de investigación |
| `dataprovider` | Universidad Politécnica de Puebla |
| `metadataformat` | Dublin Core |
| `datestamp` | Fecha de liberación del reporte |
| `autor` | Alumno que realiza la investigación, autor del reporte técnico |

```
<!ELEMENT ontologyofrecords (algorithm, cluster+)>

<!ATTLIST ontologyofrecords
date CDATA #REQUIRED>

<!ELEMENT algorithm EMPTY>
<!ATTLIST algorithm
name CDATA #FIXED "FICH"
globalsupport CDATA #REQUIRED
clustersupport CDATA #REQUIRED>

<!ELEMENT cluster
(label, level, record*, cluster*)>
<!ELEMENT label (#PCDATA)>
<!ELEMENT level (#PCDATA)>
<!ELEMENT cluster (#PCDATA)>

<!ELEMENT record
(title, subject?, description,
identifier, url, dataprovider,
metadataformat, datestamp, author)>
<!ELEMENT  title (#PCDATA)>
<!ELEMENT  subject (#PCDATA)>
<!ELEMENT  description (#PCDATA)>
<!ELEMENT  identifier (#PCDATA)>
<!ELEMENT  url (#PCDATA)>
<!ELEMENT  dataprovider (#PCDATA)>
<!ELEMENT  metadataformat (#PCDATA)>
<!ELEMENT  datestamp (#PCDATA)>
<!ELEMENT  author (#PCDATA)>

<!ENTITY generateBy "OntoSIR 2.1">
```

Figura 2.1: Ontología de registros

## 2.3. Trabajos relacionados

Esta sección describe trabajos relacionados en los cuales se realiza la visualización de documentos con base en alguna estructura representada en ontologías. Los autores de [6] presentan un modelo para la visualización de metadatos llamado modelo de objetos de documentos generalizado (*Generalizad Document Object Model tree Interface, G-DOM-Tree Interface*). Este modelo está basado en visualización XML DOM, representa los componentes lógicos y estructurales como nodos. Su implementación es parecida a la visualización de un XML Schema o al modelo UML.

En G-DOM-Tree Interface, la ontología de un documento XML es más compacta que el modelo UML o XML schema; su forma basada en identación periódicamente iguala su estructura a la de un documento XML por el uso de etiquetas en éste. Desde el punto de vista tecnológico, la visualización se implementa como una película flash o un applet de Java principalmente para explotar las técnicas para generar menús dinámicos.

En [2] se describe el sistema Spectacle, el cual provee dos formas para encontrar y explorar la información. La primera crea interfaces de hipertexto, la segunda emplea visualizaciones. El usuario puede seleccionar la forma que mejor le convenga. La visualización en Spectacle utiliza mapas de grupos, los cuales visualizan los objetos de un número de clases seleccionadas ofreciendo herramientas que permiten sobresaltar cada clase en pantalla. El usuario puede navegar por mapa de grupos ya sea posicionando el cursor sobre alguna de sus clases, dando doble clic o cambiando la barra de estado. Las ontologías se emplean para formar los mapas de grupo.

UVA es un sistema cuya interfaz se basa en representaciones de árboles tridimensionales, los cuales visualizan una colección de documentos. En la interfaz del usuario se pueden observar los primeros niveles del árbol, los cuales están agrupados en un máximo de 7 subconjuntos. Partiendo del hecho de que la Clasificación del Congreso

tiene 21 categorías básicas, al inicio se presentan 7 grupos los cuales están formados por 3 subgrupos. Cuando el usuario selecciona uno de los nodos, éste se expande y muestra sus elementos agrupando a todos los demás conjuntos en uno sólo, el cual se diferencía con otro color. La política se aplicará mientras el número de elementos sea más grande que el factor de agrupación, posteriormente llega a elementos no agrupados por los cuales podrá descender al siguiente nivel. Al llegar al final de cada nodo, el usuario encuentra la ficha bibliográfica. La interfaz presenta al usuario un mirador con controles de navegación para rotar, acercar, agrandar o trasladar. Las etiquetas muestran la siguiente información:

- Usa colores en nodos para evitar desorientación al usuario, ayudando a diferenciar categorías, agrupación de categorías, documentos y nodos ya visitados
- Se usan sonidos para alertar y retroalimentar al usuario acerca de las acciones que realiza sobre los controles de navegación
- Incluye el uso de 4 idiomas: español, inglés, francés y alemán
- Cuenta con ayuda en línea

El sistema de UVA presenta una interfaz muy completa, el inconveniente desde nuestro punto de vista, es que sólo puede ser aplicado a una biblioteca y entendido por un bibliotecario, dada la nomenclatura que maneja en la información. En caso de que existiese una biblioteca digital en la UPPuebla, este sistema no podría ser aplicable, ya que la información que un alumno busca generalmente es por contenido, no por la clasificación del congreso.

# Capítulo 3

# Metodología

La metodología es un proceso y modelado a la vez, donde se describe una idea completa de los datos y módulos que comprenderá el mecanismo de navegación a construir. Esta fase se utilizará como guía para la construcción del código, pruebas y mantenimiento del sistema.

## 3.1. Especificación de requerimientos

La Tabla 3.1 muestra los requerimientos funcionales de visualización de la colección. Se considera como entrada una ontología en XML. Cada grupo está descrito por una etiqueta.

Como requerimientos no funcionales se consideran los siguientes:

1. El mecanismo de navegación se encontrará disponible en la web, se espera que

Tabla 3.1: Descripción del requerimiento visualizar colección

| **Identificador** | RF-1 |
|---|---|
| Nombre de requerimiento: | Visualizar colección |
| Descripción corta: | Consultar elementos de la colección de datos |
| Descripción detallada: | 1.- El alumno da clic sobre la estrella que representa un documento y visualiza su información |

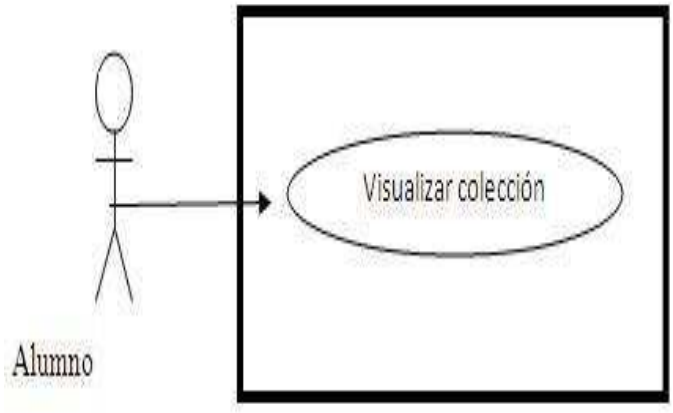



Figura 3.1: Caso de uso para visualizar colección

Tabla 3.2: Módulo para visualizar la colección de documentos

| | |
|---|---|
| Descripción: | El alumno podrá visualizar toda la colección de documentos |
| Datos de entrada: | |
| Datos de salida: | - Visualización de la colección de documentos |

sea consultado principalmente por usuarios de la comunidad universitaria con el objetivo de conocer de forma general la Colección de Reportes Técnicos CORTUPP[1] . Este mecanismo se implementará de manera independiente del sistema operativo

2. La visualización podría verse afectada por la configuración de los navegadores

## 3.2. Casos de uso

Las Figuras 3.1 y 3.2 representan las tareas que realizará el alumno, éstas se implementarán en los módulos descritos en las tablas 3.2 y 3.3, respectivamente.

La Tabla 3.4 describe el actor del mecanismo de visualización.

[1] http://informatica.uppuebla.edu.mx/tildecortupp/index.html

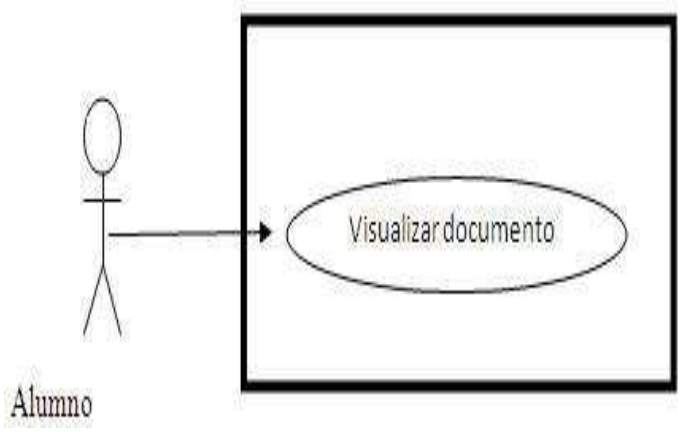



Figura 3.2: Caso de uso para visualizar documento

Tabla 3.3: Módulo para consultar un documento

| | |
|---|---|
| Descripción: | El alumno podrá consultar los datos de un documento de la colección |
| Datos de entrada: | |
| Datos de salida: | - Visualización de los datos bibliográficos del documento que desea consultar o de aquellos ubicados en el mismo nivel de la ontología |

Tabla 3.4: Especificación del actor alumno

| | |
|---|---|
| Descripción: | Persona que ingresa y consulta el mecanismo de navegación |
| Características: | Alumno regular inscrito en la UPPuebla |
| Relaciones: | Interactúa con el sistema para navegar y encontrar documentos de interés. |

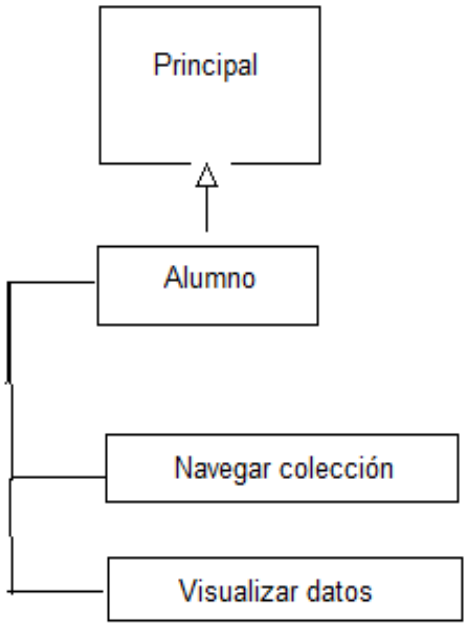



Figura 3.3: Diseño conceptual

## 3.3. Diagrama conceptual

La Figura 3.3 muestra el diagrama conceptual del sistema, el cual indica que desde la página principal el alumno observará la colección de documentos sobre la cual podrá navegar para visualizar los datos bibliográficos pertenecientes a un proyecto de investigación.

La Figura 3.4 muestra la pantalla de visualización del sistema. La Figura 3.5 muestra la pantalla de consulta del documento.

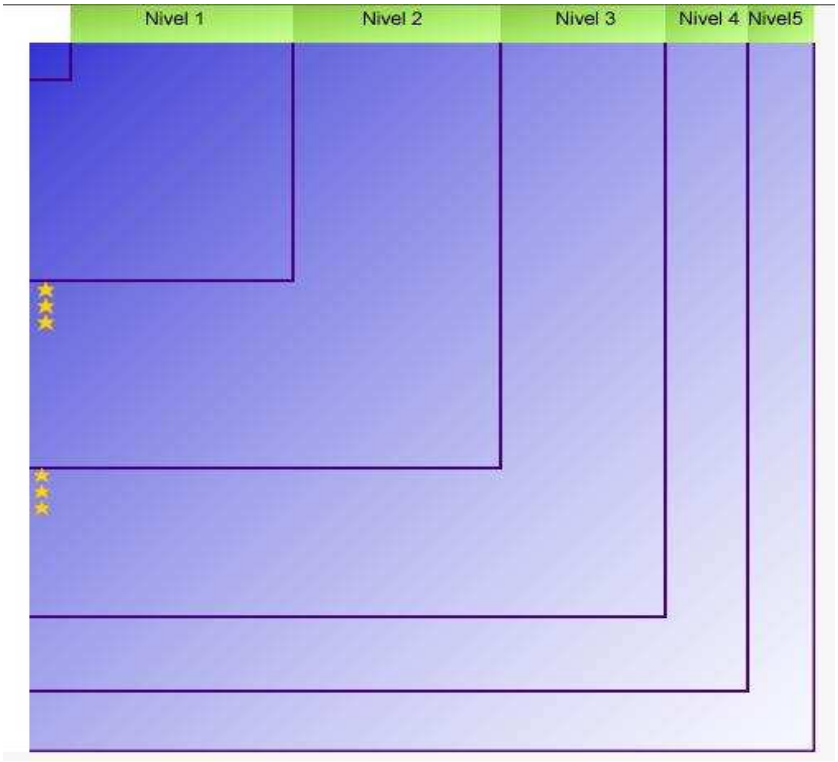
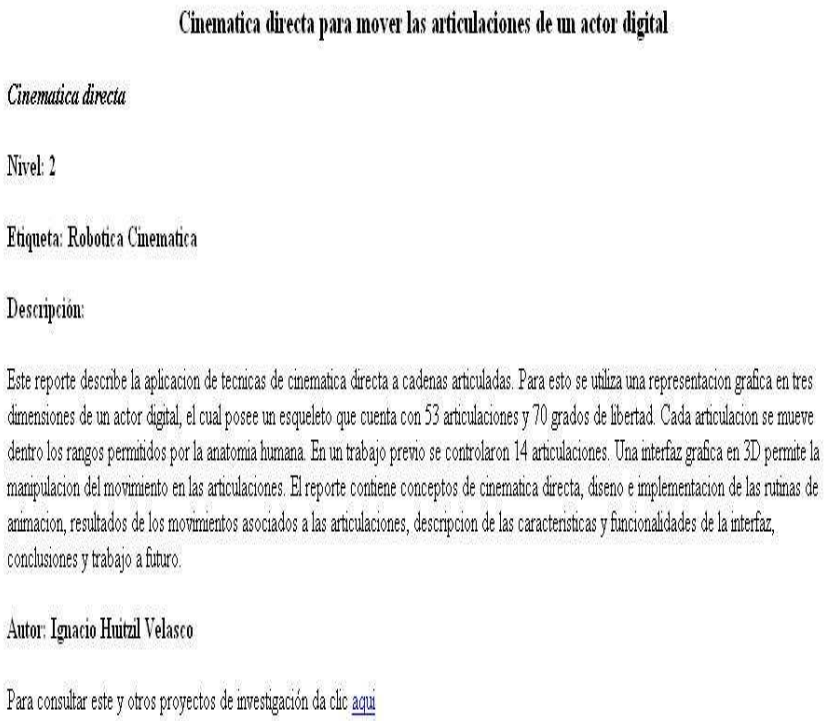



Figura 3.4: Visualización de la ontología



**Cinematica directa para mover las articulaciones de un actor digital**

*Cinematica directa*

Nivel: 2

Etiqueta: Robotica Cinematica

Descripción:

Este reporte describe la aplicacion de tecnicas de cinematica directa a cadenas articuladas. Para esto se utiliza una representacion grafica en tres dimensiones de un actor digital, el cual posee un esqueleto que cuenta con 53 articulaciones y 70 grados de libertad. Cada articulacion se mueve dentro los rangos permitidos por la anatomia humana. En un trabajo previo se controlaron 14 articulaciones. Una interfaz grafica en 3D permite la manipulacion del movimiento en las articulaciones. El reporte contiene conceptos de cinematica directa, diseno e implementacion de las rutinas de animacion, resultados de los movimientos asociados a las articulaciones, descripcion de las caracteristicas y funcionalidades de la interfaz, conclusiones y trabajo a futuro.

Autor: Ignacio Huitzil Velasco

Para consultar este y otros proyectos de investigación da clic aqui

Figura 3.5: Consulta de datos bibliográficos

## 3.4. SVG

SVG es una aplicación de XML para describir gráficas vectoriales bidimensionales los cuales pueden ser estáticas o animadas. Las gráficas pueden ser agrupados y transformados sin perder propiedades o alteraciones en su diseño. El código siguiente muestra parte del código de una gráfica en SVG.

```
<rect x=20 y=10 width=120 height=40 >
<animate attributeName=width from=120 to=40 begin=0s dur=8s fill=freeze/>
< animate attributeName=height from=40 to=82 begin=6s dur=7s fill=freeze />
< /rect>
< /svg>
```

El documento inicia con la primitiva $< svg>$ la cual indica el inicio del documento SVG con un alto y ancho del 100 % de la pantalla especificado en sus atributos `width` y `height`. Posteriormente utiliza el atributo `rect` para dibujar un rectángulo partiendo del punto `x` e `y` con un ancho de 120 y 40 pixeles de alto. El atributo `animate` se utiliza para dar animación al rectángulo donde el ancho se reducirá durante 8 segundos de 120 a 40 y el alto incrementará de 40 a 82 desde el segunto 6 hasta el segundo 7. Para finalizar el documento, se escribe $< /svg>$.

# Capítulo 4

# Implementación

Las secciones de este capítulo describen las fases principales de la implementación del mecanismo de visualización.

## 4.1. Instalación de Xerces

Xerces es un API que contiene una serie de librerías escritas en Java, cuya funcionalidad es la de un *parser*, es decir, un analizador sintáctico que convierte su entrada en una forma estructural de árbol y cuya utilidad es la de validar y editar XML.

Para validar el archivo `ontologyofrecords.xml` que contiene una ontología de registros, es necesario descomprimir Xerces (en este proyecto se utilizó la versión 1.4.4) en c:/xerces/ y añadir la ruta respectiva en variables de entorno como muestra la Figura 4.1.

Después de agregar las variables de entorno, se requiere incluir las librerías que se muestran seleccionadas en la Figura 4.2, pasar el nombre del documento (Figura 4.3) y finalmente el nombre de la etiqueta XML de la cual se desea recuperar su contenido (Figura 4.4).


Figura 4.1: Variables de entorno y del sistema

DomParserExample.java

```
1  import java.util.*;
2  import java.util.StringTokenizer;
3  import java.io.*;
4  import java.io.IOException;
5  import java.util.ArrayList;
6  import java.util.Iterator;
7  import java.util.List;
8  import java.util.Vector;
9  import javax.xml.parsers.DocumentBuilder;
10 import javax.xml.parsers.DocumentBuilderFactory;
11 import javax.xml.parsers.ParserConfigurationException;
12 import org.w3c.dom.Document;
13 import org.w3c.dom.Element;
14 import org.w3c.dom.NodeList;
15 import org.xml.sax.SAXException;
16
17 public class DomParserExample {
18
19     //No generics
20     List myOntology;
21     List myOntology2;
22     Document dom;
```

Figura 4.2: Importación de las librerías de Xerces en Java

```
//Using factory get an instance of document builder
DocumentBuilder db = dbf.newDocumentBuilder();

//parse using builder to get DOM representation of the XML file
dom = db.parse("ontologyofrecords.xml");
```

Figura 4.3: Nombre del documento XML a procesar

```
Element docEle = dom.getDocumentElement();

//get a nodelist of <record> elements
NodeList nl = docEle.getElementsByTagName("record");
if(nl != null && nl.getLength() > 0) {
    for(int i = 0 ; i < nl.getLength();i++) {

        //get the record element
        Element el = (Element)nl.item(i);

        //get the Record object
        record e = getRecords(el);
```

Figura 4.4: Nombre del elemento XML a recuperar

## 4.2. Estructura de archivos

Los archivos principales del código implementado son: `ontologyofrecords.xml`, `DomParserExample.java`, `record.java` y `cluster.java`, los cuales se relacionan entre sí como se muestra en la Figura 4.5. Estos archivos se describen a continuación:

- **ontologyofrecords.xml**: documento xml que contiene la ontología de los proyectos de investigación, generado a través de un documento DTD el cual se procesa utilizando el parser Xerces 1.4.4[1] .

- **DomParserExample.java**: este archivo es ejecutado en JCreator LE 3.50, y contiene métodos que pueden observarse en la Tabla 4.1.

- **record.java**: posee los atributos de tipo `String` `title`, `subject`, `description`, `identifier`, `url`, `dataprovider`, `metadataformat`, `datestamp` y `author`. Estos atributos se manipulan a través de métodos `get` y `set`. Esta clase sobre escribe el método `toString()`, el cual almacena en un buffer los datos que se depositaron en los atributos de `record`, obtenidos por el método `get` que cada uno posee.

[1] Disponible en http://xerces.apache.org/xerces-j/

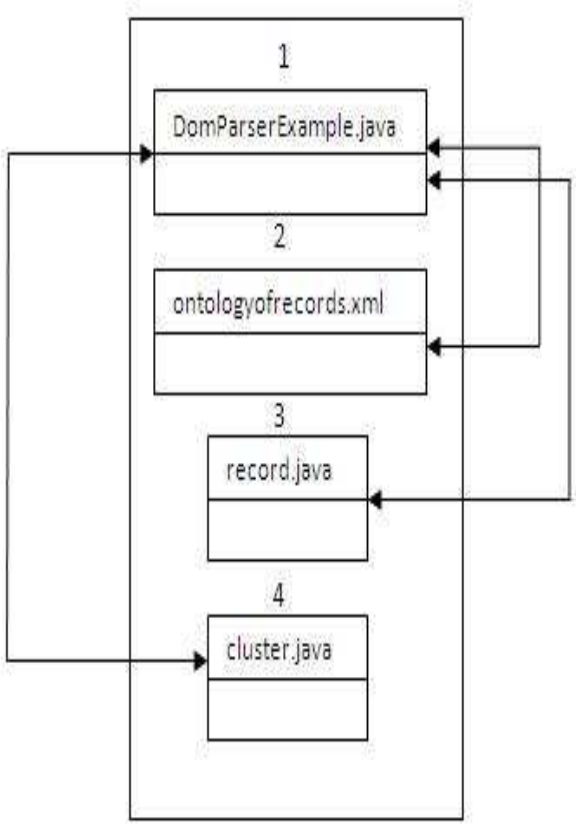



Figura 4.5: Diagrama de clases

Tabla 4.1: Métodos de la clase DomParserExample.java

| **Método** | **Entrada** | **Salida** |
|---|---|---|
| `parseXmlFile`() | - | Lee la información del documento XML y la almacena en un objeto de tipo `Document` |
| `parseDocument`() | - | Obtine un `arrayList` que contiene objetos de tipo `record` y `cluster` |
| `getRecords`() | Elemento extraido del archivo XML con etiqueta `record` | Un objeto tipo documento |
| `getCluster`() | Elemento leído del archivo XML con etiqueta `cluster` | Un objeto tipo grupo |
| `crearFichero`() | - | Crea el documento con extensión SVG |
| `dEstrella`() | Coordenadas a dibujar dentro del archivo SVG, el nivel al que pertenece el documento y la escala que difiere según el nivel | Genera el dibujo de estrella con las coordenadas correspondientes al nivel del documento para almacenar en el archivo SVG |
| crearHTML() | Recibe un vector que contiene la información de los documentos | Genera los documentos HTML que se visualizan al dar clic sobre una estrella en el mecanismo de navegación |

- **cluster.java**: este archivo posee los atributos `label` (`String`), `level`(`String`) y `record` (`Vector`) y los métodos `get`, `set` y `toString`() para almacenar los componenetes de un grupo.

## 4.3. Generación del archivo SVG

La representación gráfica del archivo `ontologyofrecords.xml` lleva por nombre `SVGarchivo` con extensión `SVG` y se encuentra almacenado en `C:/archivosg`. Este

documento contiene el resultado de cada uno de los métodos que fueron llamados en `DomParserExample.java`. La primera instrucción es:

< *svg* width=”502”height = ”502”>

Esta instrucción indica el espacio de trabajo sobre el cual se dibujará el mecanismo de navegación, en este caso corresponde a 502 puntos en pantalla de ancho por 502 puntos en pantalla de altura. Posteriormente se encuentra las líneas siguientes:

```
<linearGradient id=marino-cielo x1=0 % y1=0 % x2=100 % y2=100 %>
<stop offset=0 % style=stop-color:mediumblue;stop-opacity:0.8 / >
<stop offset=100 % style=stop-color:deepskyblue;stop-opacity:0.4/>
< /linearGradient>
```

La etiqueta `linearGradient` sirve para especificar un gradiente que será llamado con el nombre de marinocielo como lo indica el `id`, el cual comenzará desde un punto inicial (x1=0 % y1=0 %) hasta un punto final (x1=0 % y1=0 %) con el color mediumblue con opacidad de color de 0.8 indicado en la propiedad `style` para finalizar en color `deepskyblue` con opacidad del 0.4. Es preciso mencionar que este degradado corresponde al color de fondo del mecanismo. Similarmente se aprecia en las instrucciones siguientes las cuales correponden al color de fondo para las etiquetas de los niveles:

```
<linearGradient id=verdeolivo x1=0 % y1=0 % x2=100 % y2=100 % >
<stop offset=0 % style=stop-color:blueviolet;stop-opacity:0.96/”>
<stop offset=100 % style=stop-color:fuchsia;stop-opacity:0.4/”>
< /linearGradient>
```

Otra de las instrucciones del archivo para crear la estrella son:

```
<symbol id=estrella>
< polygon points=186,104,233,104,250,60,
267,104 314,104,280,138 304,190 250,
158 196,190 220,138
```

`style=fill:gold;stroke:orange; stroke-width:4/>`

</symbol>

En las instrucciones anteriores la etiqueta <`symbol id=estrella`> contiene las coordenadas del polígono que representa la estrella así como sus características: color de relleno (`fill:gold`), color de borde (`stroke:orange`) y el grosor del borde (`stroke-width:4`); establecidas estas característica solo bastará con hacer referencia al `id` para dibujar la estrella en el mecanismo de navegación cuantas veces sea necesario como se muestra en las siguientes instrucciones:

<`use x=0 y=0 xlink:href=#estrella`

`transform=translate(15.0,203.0) scale(0.08)` />

La etiqueta <`use`> recibe como parámetros las coordenadas donde se dibujará la estrella `translate(15.0,203.0)` y la escala scale(0.08), cabe mencionar que ésta depende del nivel en el que se represente, para el nivel 1 escala=0.09, nivel 2 escala=0.08, nivel 3 escala=0.07, nivel 4 escala=0.06 y para el nivel 5 escala=0.05. Esta etiqueta se encuentra almacenada dentro de <a> como se muestra a continuación:

<a `xlink:href=C: archivosg PII-10-04-09.html` >

< `use x=0 y=0 xlink:href=#estrella`

`transform=translate(15.0,307.0)`

`scale(0.07)/>` </a>

La etiqueta <a> recibe como parámetros `xlink` que contiene una referencia al documento que almacena los datos bibliográficos que se visualizarán al dar clic sobre la estrella, en este caso `PII-10-04-09.html`, el nombre del documento corresponde al nombre del identificador contenido en la ontología. Para finalizar la etiqueta se usa la etiqueta </a>.


Figura 4.6: Mecanismo de navegación

## 4.4. Implementación del mecanismo de navegación

La implementación del mecanismo de navegación lleva consigo una serie de requisitos que incluyen el uso del componente de Java J2SDK, el editor JCretor LE 3.50 y posteriormente el *plug in* para visualizar archivos SVG [2] .

En el directorio C:/ se crea una carpeta con el nombre de `archivosg` donde se almacenarán los archivos que se generan al ejecutar `DomParserExample.java`, una vez ejecutado se abre el archivo `frame.html` y muestra una pantalla como se ilustra en la Figura 4.6.

El mecanismo de navegación ilustra 5 regiones que pertenecen a los niveles dentro de la ontología y que son representados por líneas que forman ángulos de 90 grados. Otro

[2] Disponible en http://www.adobe.com/svg/viewer/install/


Figura 4.7: Representación de 6 documentos de una ontología de registros

elemento del mecanismo son las estrellas que simbolizan cada uno de los documentos que se encuentran dentro de la ontología, colocadas dentro del nivel al que pertenecen. Para conocer los datos bibliográficos, basta con dar clic sobre la estrella para visualizarlos en el costado derecho de la pantalla. A manera de ejemplo, la Figura 5.3 muestra la representación de una ontología de registros formada por 6 documentos.

Los datos bibliográficos se cambian si se de clic sobre otra estrella. El mecanismo se despliega utilizando los colores oficiales de la Universidad Politécnica de Puebla, como se aprecia en la Figura 5.3, aunque éstos pueden modificarse en el archivo `colores.txt` ubicado en `c:/archivosg`. La Figura 4.8 representa el mecanismo de navegación con otros colores.

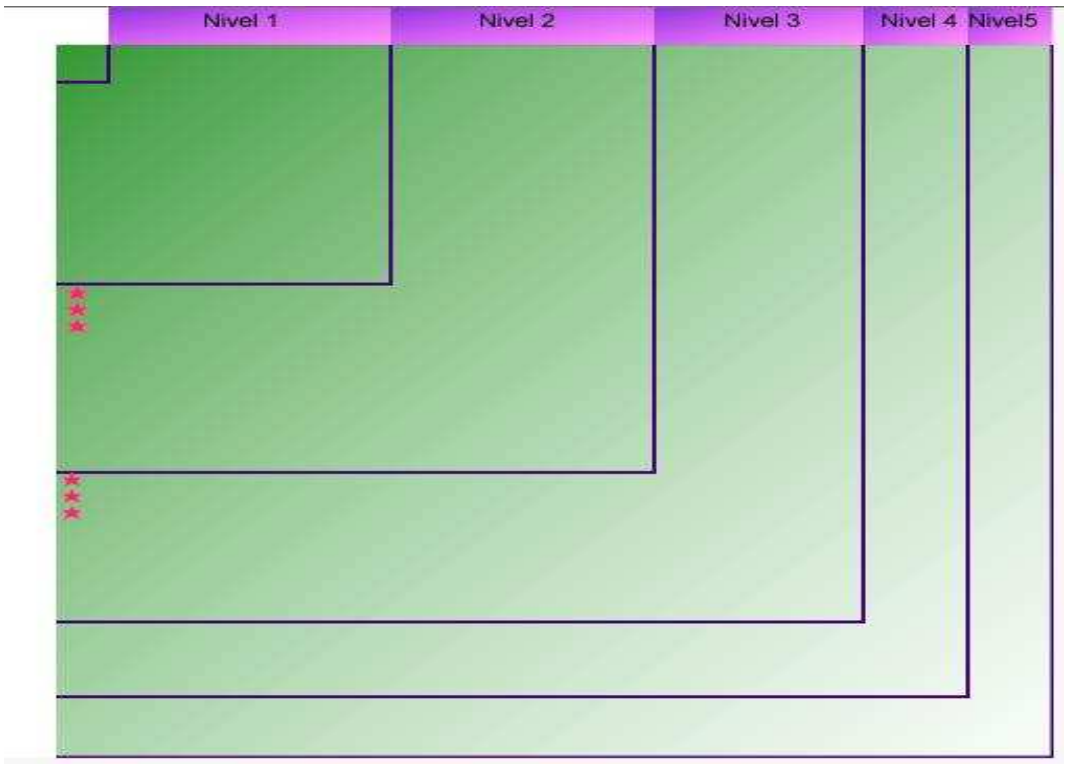



Figura 4.8: Ejemplo de personalización de colores

# Capítulo 5

# Resultados

En este capítulo se describen las pruebas que se realizaron al mecanismo de navegación, así como los resultados obtenidos en cada una de éstas.

## 5.1. Pruebas por unidad

Se analizó cada una de las funciones de la clase `DomParserExample.java`, en particular las que corresponden a la lectura y escritura de documentos. Para probar el método de lectura, se incrementaron 6 documentos en los niveles 1,2 y 3 con el objetivo de verificar que se extrajera y almacenara la información correctamente. Se verificó la obtención de resultados a través de mensajes en consola.

El primer dato corresponde al número de registros leídos en la ontología, en seguida se muestran los datos pertenecientes a los elementos `ontologyofrecords`, `title`, `subject`, `description`, `identifier`, `url`, `dataprovider`, `metadataformat`, `datestamp` y `author`. En cuanto al método de escritura, se comprobó que el contenido corresponda a cada uno de los documentos digitales representados y se analizó la representación por niveles. Se obtuvieron resultados favorables como se observa en la Figura 5.2.

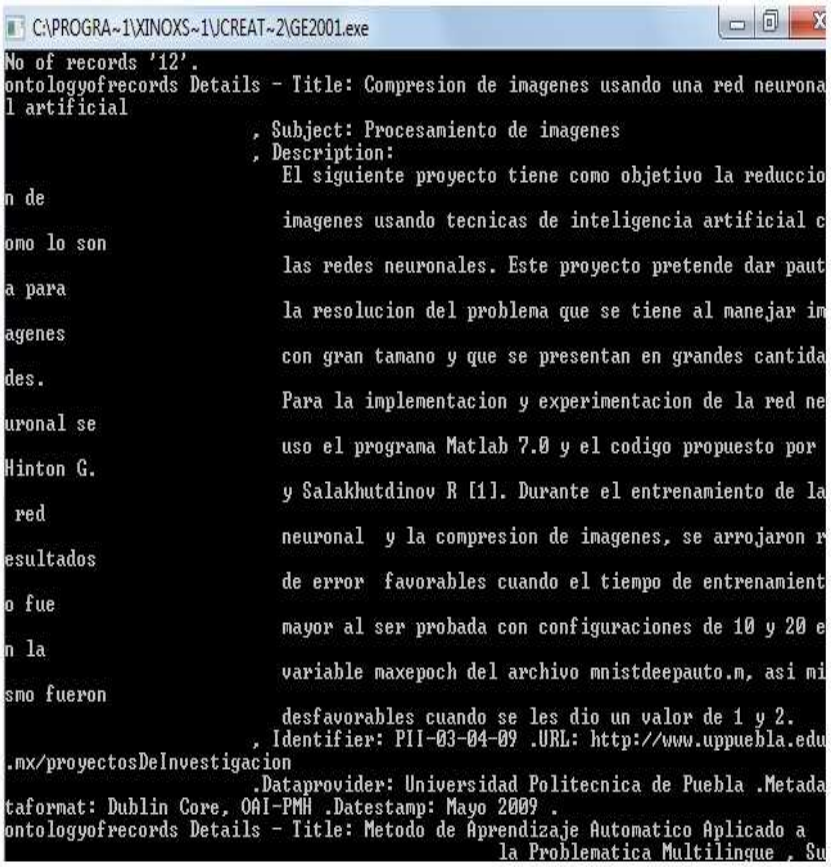



Figura 5.1: Salida en consola de la lectura del archivo ontologyofrecords.xml

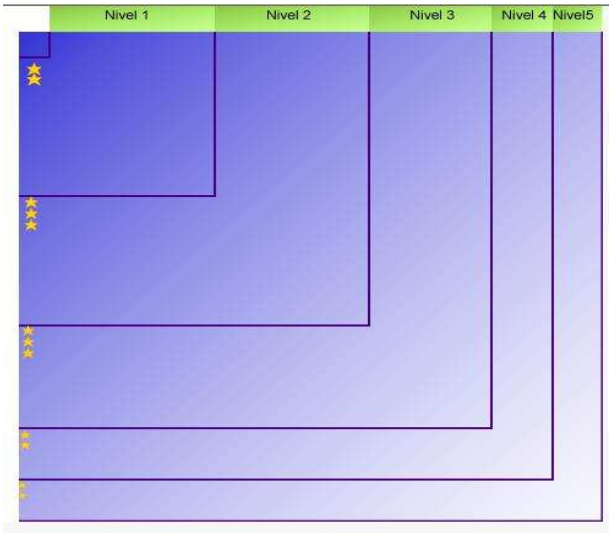



Figura 5.2: Incrementación de documentos en la visualización

## 5.2. Pruebas de integración

En esta prueba el sistema se analizó como un todo para descubrir errores asociados con la interfaz. La Figura 5.3 muestra la integración de tres páneles en una página web que despliega el mecanismo de navegación.

## 5.3. Pruebas de visualización en diferentes navegadores

Se realizaron pruebas de visualización para el mecanismo de navegación en diferentes navegadores: Firefox 1.5, Firefox 3.6, Firefox Beta, Internet Explorer 8, Google Chrome 8.0.552.215 y Opera 10.6; de los cuales se obtuvieron resultados favorables en los últimos tres, donde se mantuvo constante la representación y la funcionalidad. Las figuras 5.4 y 5.5 son ejemplo de ello. Sin embargo, no se obtuvieron los mismos resultados en las diferentes versiones de Firefox, esto debido a que SVG es una aplicación de reciente

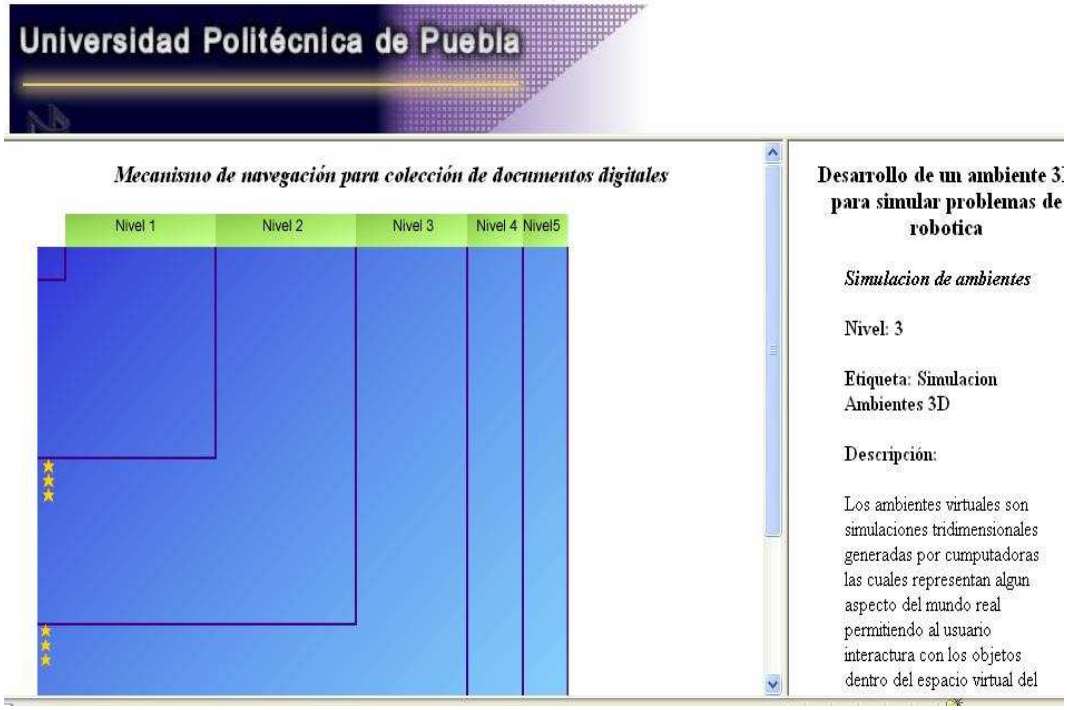



Figura 5.3: Correcto Funbcionamiento del mecanismo de visualización

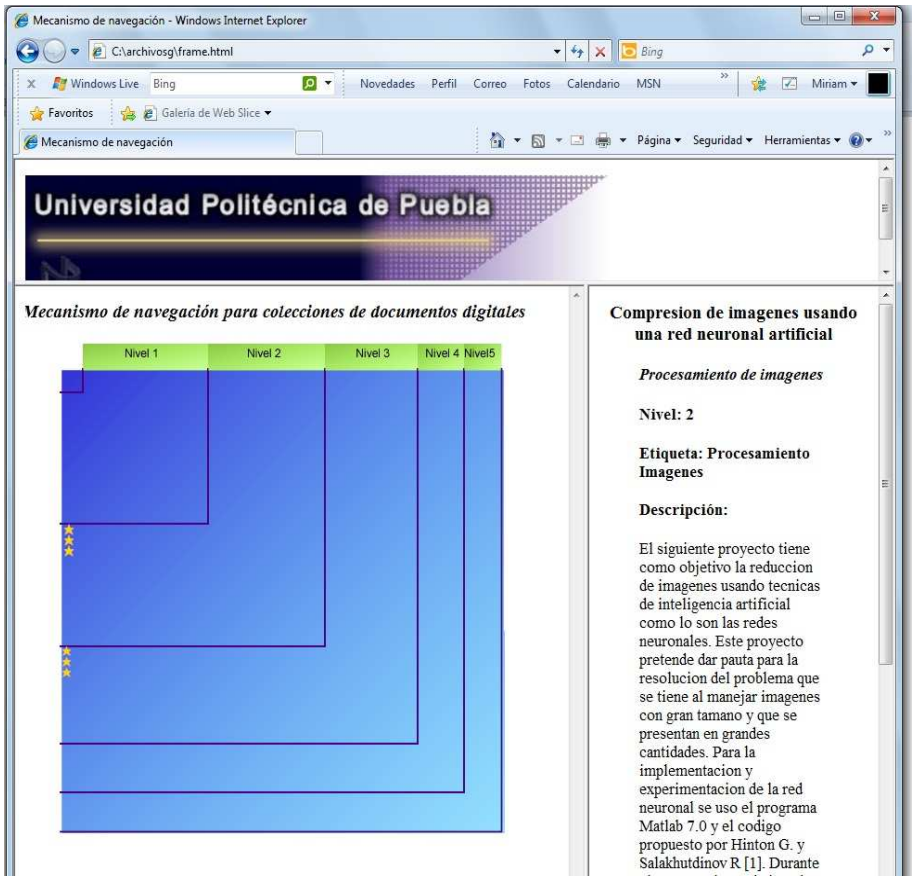



Figura 5.4: Visualización en Internet Explorer 8

creación y en ese navegador no soporta las características de animación[1] .

[1] Sustentado en http://www.mozilla.org/projects/svg/status.html

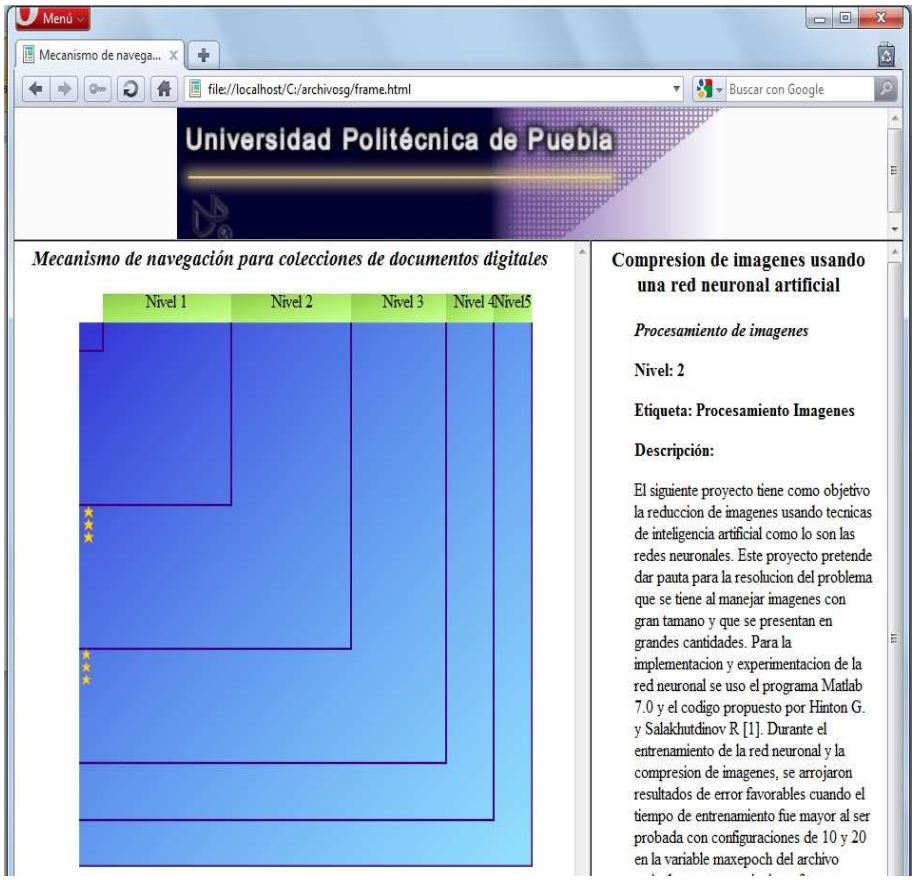



Figura 5.5: Visualización en Opera 10.6

# Capítulo 6

# Conclusiones

Una vez alcanzados los objetivos y con base en los resultados de las pruebas realizadas al mecanismo de navegación, se puede concluir lo siguiente:

SVG es una aplicación que utiliza un lenguaje sencillo. Desde el punto de vista de los desarrolladores, sintácticamente es similar a HTML. En el proyecto se utilizó este lenguaje para la construcción del archivo `SVGarchivo.svg`. SVG soporta los cambios de escala, lo cual se puede observar en las estrellas que a pesar de pertenecer a niveles diferentes, y por tanto representadas en diferentes escalas, no deja de apreciarse su forma ni se pierde el nivel de detalle.

Para la elaboración del mecanismo se identificaron 4 etapas principales: 1) análisis de `SVG`, 2) revisión de la estructura de una ontología de registros para construir las clases `record` y `cluster`, 3) elaboración de la clase `DomParserExample.java` que procesa el archivo `XML` y recupera los datos de los elementos y 4) creación del archivo `SVGarchivo.svg` que representa el mecanismo de navegación, incluye degradados, colores, coordenadas de las líneas utilizadas en los niveles, coordenadas pertenecientes a cada una de las estrellas y enlaces a los documentos y sus datos bibliográficos.

El mecanismo desarrollado ofrece una interfaz alternativa para acceder a los datos de un documento en específico. Está disponible en:

http://informatica.uppuebla.edu.mx/~visualizacionPI/index.html

Los principales obstáculos en la implementación del mecanismo fueron: 1) la inclusión de enlaces hacia los documentos que contienen los datos bibliográficos, debido a que el nombre de dichos archivos corresponde al idetificador del documento y éste contenía retornos de carro que no se aprecian a simple vista, pero que fungen como delimitadores en el parser de Xerces. Este problema se solucionó con la reedición del documento `ontologyofrecords.xml` y 2) las pruebas en diferentes versiones de Firefox, dado que en las líneas que corresponden a la animación de una estrella en la página principal, este navegador las identificaba como errores de sintaxis. En Firefox, la implementación de SVG se encuentra aún en desarrollo.

El mecanismo se concluyó con éxito al aprobar cada una de las pruebas y obtener resultados favorables. Como trabajo a futuro, se plantean las siguientes actividades:

- Visualizar los grupos y sus datos descriptivos
- Probar el mecanismo para visualizar colecciones de cientos de documentos

# Bibliografía


[1] Chandrasekaran B. and Josephson John R. What are ontologies, and why do we need them? *IEEE INTELLIGENT SYSTEMS*, pages 20–26, 1994.

[2] Fluit C., Sabou N., and Harmelen van F. *Ontology-based information visualization in Visualizing the Semantic Web. XML-based Internet and Information Visualization.*, volume 1. Springer, Wokingham, England, segunda edition, Julio 2003.

[3] Proal C. Sistema uva: interfaces para visualización de grandes colecciones digitales. Tesis de maestría, Universidad de las Américas Puebla, Santa Catarina Mártir S/N, San Andrés Cholula, Puebla, México., Mayo 2002.

[4] B.C.M. Fung, K. Wang, and M. Ester. Hierarchical document clustering using frequent itemsets. In *Proceedings of the Third SIAM International Conference on Data Mining, (SDM'03, San Francisco, California, May),*, pages 59–70, San Francisco, CA, USA, May 2003. SIAM.

[5] Chen C. Geroimenko V. *Visualizing the Semantic Web. XML-based Internet and Information Visualization.*, volume 1. Springer, Wokingham, England, segunda edition, Julio 2003.

[6] Geroimenko L. Geroimenko V. *Interactive interfaces for mapping e-commerce ontologies in Visualizing the Semantic Web. XML-based Internet and Information Visualization.*, volume 1. Springer, Wokingham, England, segunda edition, Julio 2003.

[7] T.R. Gruber. A translation approach to portable ontology specification. *Knowledge Adquisition*, 5(2):199–220, 1993.

[8] M.A. Medina and J.A. Sánchez. Ontoair: a method to construct lightweight ontologies from document collections. In *Proceedings of the Ninth Mexican International Conference on Computer Science 2008, (ENC 08, Baja Californiaí, México, October)*, page 12. IEEE Computer Society, October 2008.