Centro de Investigación y de Estudios
Avanzados del Instituto Politécnico Nacional
Departamento de Computación

Modelado de datos para base de datos espaciales. Caso de estudio: sistemas de información geográfica.

Tesis que presenta

Flor Radilla López

para obtener el Grado de

Maestro en Ciencias

en la Especialidad de

Ingeniería Eléctrica

Director de la Tesis:

Dr. Sergio Víctor Chapa Vergara

México, D. F. Mayo 2008

Agradecimientos

Gracias al Consejo Nacional de Ciencia y Tecnología por haberme proporcionado el apoyo económico que sirvió para sustentar mi estancia durante el tiempo que realicé mis estudios de maestría.

Gracias al Consejo Nacional de Ciencia y Tecnología por financiar, a través del Convenio-Colima-2005-C01-28, el proyecto: "Sistema de información geográfica para el manejo sustentable de cuencas hidrológicas".

Gracias a Dios por brindarme vida y salud para seguir luchando en este camino de dudas, incertidumbre y sentimientos encontrados que al final nos enseña a mejorar como persona.

Gracias al Dr. Sergio Víctor Chapa Vergara por la gran oportunidad que me brindó para regresar al Cinvestav porque a través de ésta he aprendido mucho no sólo en conocimiento técnico sino a nivel personal. Por el apoyo brindado para poder concluir mis estudios de maestría, por haberme dado la oportunidad y confiar en mi para realizar este trabajo de tesis.

Gracias a los Drs. Sonia Guadalupe Mendoza Chapa y Matías Alvarado Mentado, por sus valiosos comentarios durante la revisión del documento de tesis.

Gracias a mis padres por haberme apoyado durante esta etapa de mi vida.

Gracias al personal administrativo del Departamento de Computación por su atención brindada en los trámites que fueron requeridos durante mi estancia en la maestría.

Resumen

En el proceso de desarrollo de software existen diversas tareas como la especificación de requerimientos, el diseño, la programación y depuración. La falta de organización e integración de estas fases y los constantes cambios de los requerimientos afectan el proceso de software que conlleva a incosistencias entre la transición de una fase a otra y retrasos en la finalización de un producto de sofware. Por otro lado, la falta de una metodología para el diseño de sistemas que consideren el manejo de datos complejos (e.g. *sistemas de información geográfica, sistemas de información ambiental, biología*) aumenta el dominio del problema.

Para reducir la problemática mencionada, en esta tesis desarrollamos una herramienta de modelado de datos con el objetivo de diseñar sistemas e integrar las fases de desarrollo de los mismos, de tal manera que sea posible agilizar y automatizar las tareas que engloban la creación de sistemas. Se considera como caso de estudio el diseño de un modelo de datos para un *Sistema de Información Geográfica* (*SIG*). La herramienta permite crear tres tipos de modelos (*base de datos, interfaz de usuario, comportamiento*) de manera gráfica y son traducidos en documentos *XML* para su posterior uso.

La principal contribución de nuestro trabajo consiste en proporcionar una metodología integrada en una herramienta para el desarrollo de modelos de datos de sistemas que manejen datos complejos (e.g. SIGs). Un aspecto importante en el desarrollo de este trabajo es el uso de tecnologías y herramientas de software de distribución gratuita para el desarrollo de aplicaciones con el propósito de aprovechar los beneficios que ofrece el uso de este tipo de software como: reducción del costo, calidad, desempeño y seguridad. Finalmente, la aplicación de estándares para datos geográficos permite seguir una norma respecto a la representación de la información geográfica para poder interoperar con otros sistemas que manejen datos geográficos.

Abstract

In the process of software deveplopment there are a number of tasks such as specification requirements, design, programming and debugging but, the lack of organization and integration of these phases and the constantly changing of requirements affect the software process that leads to inconsistencies between the transition from one phase to another and delays in the completion of a software product. Therefore the absence of a methodology for the design of systems that consider the management of complex data (e.g. *geographic information systems, environmental information systems, biology*) increases the problem domain.

To reduce the problematic mentioned, in this thesis we developed a data modeling tool with the goal of designing systems and integrating the phases of development of the same in such a manner as possible to speed up and automate tasks that include building systems. It is considered as a case study the design of a data model for *Geographic Information System* (*GIS*). The tool allows you to create three types of models (*database, user interface and behavior*) in graphical form and are translate into *XML* documents for later use.

The main contribution of this thesis work is to provide an integrated approach in a tool for the development of data models of systems that handle complex data (e.g. SIGs). An important aspect in the development of this work is the use of technology and free software tools application development, the purpose, reap the benefits offered by the use of such software as: cost reduction, quality, performance and safety. Finally, the implementation of standards for geographic data allow to follow a standard regarding the representation of geographic information to be able to interoperate with other systems to handle geographical data.

ÍNDICE GENERAL

Resumen vi

Abstract viii

Índice de figuras xiii

Índice de tablas xvi

1. Introducción 1
- 1.1. Panorama general . . . 1
- 1.2. Antecedentes y motivación . . . 3
  - 1.2.1. CASE: Ingeniería de Software Asistida por Computadora . . . 4
  - 1.2.2. Programación visual en sistema CASE . . . 4
  - 1.2.3. Sistemas abiertos . . . 5
- 1.3. Planteamiento del problema . . . 6
- 1.4. Objetivo general . . . 6
- 1.5. Estructura de la tesis . . . 6

2. Fundamentación y modelos 9
- 2.1. Modelo de desarrollo de software . . . 9
  - 2.1.1. Ingeniería dirigida por modelos . . . 11
  - 2.1.2. Ingeniería inversa dirigida por modelos . . . 13
  - 2.1.3. Desarrollo dirigido por modelos . . . 15
  - 2.1.4. Arquitectura dirigida por modelos . . . 15
- 2.2. Modelo de datos . . . 16
  - 2.2.1. Metodología de diseño de bases de datos . . . 17
  - 2.2.2. Tipos de modelos de datos . . . 18
  - 2.2.3. Problemas abiertos . . . 19
- 2.3. Sistemas de información geográfica . . . 23
  - 2.3.1. Fases de un proyecto SIG . . . 25

2.3.2. Bases de datos geográficas 25
2.3.3. Modelo de datos en SIG 27
2.3.4. Semántica de la información espacial 31

3. Tecnologías y herramientas **33**
3.1. API Java Swing 33
3.2. Tecnología XML 36
3.2.1. Estructura básica de un documento XML 38
3.3. Transformación de documentos con XSLT 41
3.4. Procesador dom4j 42
3.5. Manejador de base de datos PostgreSQL 43
3.5.1. PostGIS 44
3.6. Entorno de desarrollo integrado NetBeans 45

4. Estándares para datos geográficos **47**
4.1. Seguimiento de estándares 47
4.2. La problematica del geoprocesamiento 48
4.3. OGIS 50
4.3.1. Funcionamiento de OGIS 51
4.3.2. Beneficios 52
4.3.3. Ámbito 53
4.4. Modelo de datos geográficos abierto 53
4.4.1. Representación de los elementos geográficos 54
4.4.2. Ubicación: lugar y tiempo 55
4.4.3. Propiedades y coberturas 55
4.4.4. Semántica y metadatos 57
4.4.5. Geometrías en elementos OGIS 57
4.5. ISO/TC 211 58
4.6. Esquema conceptual GeoFrame 60

5. Diseño de la solución **63**
5.1. Descripción del problema 63
5.2. Arquitectura propuesta 65
5.2.1. Proceso de creación del modelo de datos 66
5.3. Componentes del entorno 67
5.4. Caso de estudio 69
5.4.1. Modelo de la base de datos 70
5.4.2. Modelo de interfaz de usuario 70
5.4.3. Modelo de comportamiento 72
5.5. Diagramas UML 72
5.5.1. Diagrama de paquetes 72
5.5.2. Diagramas de clases 75

6. Implementación **79**
6.1. Descripción general 79
6.2. Implementación del editor 79
6.3. Implementación del script SQL de la base de datos 81
6.4. Implementación de descripciones XML 81
6.4.1. Descripción de base de datos 82
6.4.2. Descripciones de interfaz de usuario 83
6.4.3. Descripción de comportamiento 102

7. Conclusiones y trabajo a futuro **105**
7.1. Conclusiones 105
7.2. Contribuciones 107
7.3. Trabajo a futuro 108

A. Diagrama de clases **111**

Bibliografía **112**

ÍNDICE DE FIGURAS

1.1. Estructura del documento 7

2.1. Modelos de desarrollo de software y tipos de modelos de datos 10
2.2. Componentes de un SIG 24

3.1. Herramientas de software utilizadas en esta tesis 34
3.2. Características de las clases base de Java 35
3.3. Componentes gráficos de Swing 36
3.4. Descripción de algunos componentes de Swing 37
3.5. Partes que componen un documento XML 39
3.6. Proceso de transformación de documentos XML 41

4.1. Ubicación y representación geométrica del modelo esencial 55
4.2. Característica del modelo esencial OGIS 56
4.3. Estándares ISO/TC 211 59
4.4. Diagrama de clases del esquema conceptual GeoFrame 61
4.5. Iconos para la representación de entidades geográficas 62

5.1. Arquitectura de la solución 65
5.2. Proceso de creción del modelo de datos 67
5.3. Estructura de directorios 68
5.4. Componentes del entorno 69
5.5. Entidades geográficas del modelo de hidrología 70
5.6. Entidades no geográficas del modelo de hidrología 71
5.7. Modelo de datos de la base de datos 72
5.8. Entidades del modelo de datos de interfaz de usuario 73
5.9. Entidades del modelo de datos de comportamiento 73
5.10. Diagrama de paquetes 74
5.11. Diagrama de clases para el editor de diagramas 76

5.12. Diagrama de clases para la generación de la estructura de la base de datos 77
5.13. Diagrama de clases para la generación de descripciones XML 78

6.1. Métodos para crear el editor de modelos 85
6.2. Métodos para crear el editor de modelos 86
6.3. Métodos para crear el editor de modelos 87
6.4. Procedimiento para la creación de modelos en el contexto gráfico. . . 88
6.5. Procedimiento para la creación de una entidad. 89
6.6. Menú contextual del panel de dibujo 90
6.7. Menú contextual de la entidad 90
6.8. Proceso de generación de documentos XML 91
6.9. Pantalla de captura para los datos de la entidad 91
6.10. Descripción XML del modelo de base de datos 92
6.11. Hoja de estilo para la creación de la estructura de la base de datos . . 92
6.12. Definición de la hoja de estilo para aplicar las transformaciones. . . . 93
6.13. Procedimiento para la creación de la descripción xml para conectarse a una base de datos. 93
6.14. Procedimiento para la creación de la descripción xml de la interfaz de usuario. 94
6.15. Procedimiento para la creación de las descripciones xml de menús. . . 95
6.16. Procedimiento para la creación de las descripciones xml de submenús. 96
6.17. Procedimiento para la creación de la descripción xml de páginas jsp. . 97
6.18. Descripción para la conexión de base datos 98
6.19. Descripción de interfaz de usuario 99
6.20. Descripción de menú 100
6.21. Descripción de submenú 101
6.22. Descripción de página jsp 101
6.23. Modelo de interfaz de usuario 102

INDICE DE TABLAS

3.1. Tecnologías XML 40

6.1. Métodos de la clase BuildXMLEntities 82
6.2. Métodos para la generación de descripciones de interfaz 84

CAPÍTULO 1

Introducción

Este capítulo aborda el panorama general que engloba el desarrollo de esta tesis. Se presentan los antecedentes y la motivación del trabajo desarrollado en donde se describen desde las herramientas *CASE* hasta la importancia de los sistemas abiertos. Además se define el planteamiento de la solución de nuestro trabajo, el objetivo general y finalmente la estructura del documento.

1.1. Panorama general

El proceso de diseño y desarrollo de software es un conjunto de tareas complejas que implica diversas actividades; para controlar la complejidad inherente en los sistemas de software existen técnicas y procesos que reducen el esfuerzo requerido para producir sistemas grandes y complejos.

En el ámbito de la ingeniería de software existe un concepto importante denominado *proceso del software* y se describe como: *un conjunto de actividades y resultados asociados que conducen a la creación de un producto de software* [38]. Tal proceso puede considerar desarrollar software desde una etapa inicial que contemple los requerimientos, sin embargo cada vez es mas común el desarrollo de nuevo software ampliando y modificando los sistemas existentes.

Con el propósito de agilizar estos procesos y disminuir la complejidad de la creación de sistemas, se ha intentado la automatización de tales procesos pero el éxito obtenido ha sido limitado. Las herramientas de *ingeniería de software asistida por computadora* (*CASE, Computer Aided Software Engineering*) pueden ayudar a automatizar algunas actividades del proceso, sin embargo no existe posibilidad de una mayor automatización en el diseño creativo del software realizado por los ingenieros. Una razón del éxito limitado de la automatización de los procesos es la inmensa diversidad de procesos del software [38].

Debido a que no existe un proceso ideal, diversas organizaciones han creado enfoques completamente diferentes para desarrollar software. Aunque los procesos de software son diferentes, tienen actividades fundamentales que son comunes para todos ellos. Éstas son:

1. **especificación del software**. Consiste en definir la funcionalidad del software y las restricciones en sus operaciones;

2. **diseño e implementación del software**. Comprende la creación de software que cumpla con su especificación;

3. **validación del software**. Implica validar el software para asegurar que hace lo que el cliente desea;

4. **evolución del software**. Contempla que el software debe evolucionar para satisfacer los cambios en las necesidades del usuario.

La mejora del proceso de creación del software se puede lograr mediante diversas formas. Puede provenir del proceso de estandarización que permite reducir la diversidad en los procesos del software dentro de una organización. Esto permite mejorar la comunicación involucrada, reducir el tiempo de capacitación y disminuir los costos de mantenimiento del proceso automatizado. La estandarización también es un medio para introducir nuevos métodos, técnicas y buenas prácticas de ingeniería de software.

Un *sistema de información* es un conjunto de elementos (información, bases de datos, personas, equipo de cómputo) que interactúan entre sí para procesar datos e información con el objetivo de apoyar las actividades que se realizan dentro de una organización. Las bases de datos son componentes esenciales de los sistemas de información porque en ellas se organiza y almacena el contenido de las aplicaciones que forman parte del sistema. El diseño de las bases de datos es el proceso por el que se determina la organización de una base de datos, considerando su estructura, contenido y las aplicaciones que se han de construir. Conforme la tecnología de las bases de datos ha avanzado, se han desarrollado metodologías y técnicas de diseño. Además, se ha logrado un acuerdo en relación a la descomposición del proceso de diseño en fases, los objetivos principales de cada fase y las técnicas para conseguir estos objetivos. La aplicación de una metodología de diseño es un factor significativo para obtener una base de datos eficiente [38].

Formalmente, el diseño de una base de datos se descompone en diseño conceptual, diseño lógico y diseño físico (sección 2.2). Un mecanismo formal matemático/computacional para representar los diseños mencionados consiste en utilizar un modelo de datos que se define como "*una serie de conceptos que puede emplearse para describir un conjunto de datos y las operaciones para manipularlos*" [3].

Los sistemas de información geográfica (*SIG*) han sido propuestos desde hace más de treinta años como una disciplina para representar, modelar, analizar y planificar el territorio, usando herramientas computacionales. Su relación con la computación partió de la simple representación gráfica de aspectos espaciales y de la generación de las estructuras de datos adecuadas para el almacenamiento de este tipo de datos hasta expandirse hoy en día en toda una gama de posibilidades que incluye la aplicación en diversas ramas como: la minería de datos aplicada para el descubrimiento de conocimiento a partir de información vectorial e imágenes de satélite, la extensión de arquitecturas de bases de datos para modelar información espacial, la aplicación de técnicas de inteligencia artificial para realizar búsquedas heurísticas en grandes cantidades de información, las técnicas de transmisión de atributos espaciales a través de redes y la ingeniería de software aplicada a proyectos de *SIG*.

Es justamente la última aplicación mencionada la que ha presentado las mayores dificultades pues el carácter de los *SIG* como ciencia multidisciplinaria ha definido la implementación de estos sistemas como una tarea problemática, tanto por la diversidad de los miembros de los equipos presentes en el dominio de aplicación, como por el amplio espectro de posibles usuarios de información espacial. Sin embargo, la falta de metodologías bien definidas para la captura de requerimientos del sistema y la existencia de arquitecturas cerradas siguen manifestándose como obstáculos en el funcionamiento de las aplicaciones de los *SIGs*.

1.2. Antecedentes y motivación

El desarrollo de nuestro trabajo de tesis surge a partir de la concepción y diseño del proyecto dirigido por el Dr. Sergio Víctor Chapa Vergara, profesor investigador del departamento de computación del CINVESTAV. Tal proyecto se denomina *diseño de base de datos asistido por computadora* (*CADBD, Computer Aided Database Design*) y comprende una metodología y un sistema innovador que ayudan en el diseño y la manufactura automática de bases de datos, así como en el análisis exploratorio de datos. La metodología tiene por objetivo crear desarrollos basados en ingeniería de modelos con conceptos comunes e instancias de distintos dominios. Por otro lado el sistema pretende automatizar los modelos con lenguajes visuales estándar. El proceso para realizar esto contempla la ingeniería de requerimientos con descripciones-representaciones, la construcción de modelos conceptuales, metadatos y modelos lógicos así como la implementación física de las bases de datos. Los enfoques de *CADBD* son: estructural/semántico, operacional/procedimental y de comportamiento/eventos discretos [45].

La motivación para desarrollar este proyecto tiene sus antecedentes en las herramientas *CASE* y en los proyectos realizados por el Dr. Sergio Víctor Chapa en las áreas de bases de datos, ingeniería de software y sistemas de información (secciones 1.2.1 y 1.2.2).

1.2.1. CASE: Ingeniería de Software Asistida por Computadora

Algunos trabajos se han desarrollado para elevar el nivel de abstracción usado en el desarrollo de software. En la década de 1980, el área de *CASE* se dedicó a crear métodos de desarrollo de software y herramientas de las que dispusieron los desarrolladores para expresar sus diseños en términos de representaciones de programación gráfica con propósitos generales, tales como máquinas de estados finitos, diagramas de estructura y diagramas de flujo de datos. Una meta de la *CASE* fue aprovechar un análisis más completo de las representaciones gráficas para sustituir la complejidad que se tiene en los lenguajes de propósito general.

Automatizar el proceso de desarrollo de software implica que la computadora participe en la tarea de elaboración del mismo, transfiriendo a la máquina el proceso de transformación de alguna o todas las etapas del proceso. La tecnología *CASE* desarrollada durante 1980 y 1990, no cumplió de manera significante con la automatización como una de las principales metas, y sólo se limitó a desarrollar software que aprovecharon los diseñadores para dibujar diagramas de arquitecturas de sistemas y documentos de diseño, los cuales fueron usados por los programadores como guía para la creación y manufactura de software.

Sin embargo, a principio de la década de 1990, con el desarrollo de la tecnología gráfica en computación, el surgimiento de la programación y lenguajes visuales vino a ser un pilar importante para la abstracción y automatización del software. Los lenguajes y ambientes visuales integran un nuevo paradigma de programación basado en esquemas y diagramas en el que la interacción hombre-máquina se basa en una comunicación mediante: iconos, diagramas, gráficas e imágenes. El interés que se ha tenido en el área ha originado el desarrollo de algunos lenguajes y ambientes visuales: como lenguajes visuales de consulta en base de datos [44] y lenguajes visuales de propósito general [34].

1.2.2. Programación visual en sistema CASE

El inicio del proyecto *CADBD* fue el proyecto denominado"*Programación automática a partir de descriptores de flujo de información*", donde se creó el lenguaje *LIDA* como lenguaje visual de alto nivel y aplicaciones en bases de datos, dentro de un paradigma de flujo de datos con potencial de ejecutar procesos en paralelo [44]. A partir de este resultado se diseñó un proyecto para desarrollar ambientes y lenguajes visuales que abarcaran desde el diseño conceptual de una base de datos hasta la visualización de resultados obtenidos por una aplicación creada por *LIDA*.

El proyecto "*Programación visual en sistema CASE*" apoyado por Conacyt 1399-E9206, planteó el desarrollo de todo un ambiente visual con capacidades de: diseñar conceptualmente la base de datos con diagramas, crear un modelo lógico relacional en

tercera forma normal con transformación automática a partir del diseño conceptual y establecer un esquema físico en un sistema manejador de base de datos.

El primer resultado de una herramienta *CASE* para la metodología entidad vínculo extendido, fue el sistema *EVE* (*Entidad Vínculo Extendido*) desarrollado en plataforma *MSDOS* (*Microsoft Disk Operating System*) y con lenguaje C [44, 40]. A partir de éste, la propuesta fue llevar la herramienta a un ambiente gráfico con mayores capacidades en el sistema *DEC 3000/300X*[1], el entorno gráfico basado en ventanas *XWindows* proveía las primitivas de graficación de bajo nivel *Xlib*[2], las bibliotecas de alto nivel *Xtoolkits*[3] y *OSF/Motif*[4] dentro del sistema operativo *UNIX*. El sistema *EVE* se constituye como un ambiente visual donde se diseñan diagramas conceptuales entidad-vínculo-extendido, se generan dependencias funcionales, se normaliza con el método de Berstein y se obtiene como resultado un modelo lógico de base de datos [10].

1.2.3. Sistemas abiertos

En este contexto, parte de la estrategia es aprovechar al máximo los "sistemas abiertos", tanto para la tecnología de "hardware" y "software", como para los esquemas de organización de los grupos de trabajo. Las telecomunicaciones en la actualidad hacen innecesarias muchas de las grandes infraestructuras de los países desarrollados, que se vuelven obsoletas a la velocidad con que nuevas tecnologías "abiertas" bajan considerablemente de precio o se obtienen gratuitamente, siendo más accesibles a países en vías de desarrollo [37].

La comunidad actual aunque todavía no está familiarizada con el "software gratuito", ya tiene nociones de lo que hace la *Free Software Foundation* o que son los sistemas operativos abiertos como *Linux* y *USENET* los cuales permiten recibir soporte de primera calidad sin tener que desembolsar un centavo a diferencia del software propietario por el cual se tiene que pagar una gran cantidad de dinero para hacer uso de él y no aportan la robustez ni la potencia que tienen las herramientas de software de distribución gratuita. Consideramos que los sistemas abiertos pueden romper los anteriores círculos de dependencia porque el desarrollo de aplicaciones en una plataforma abierta funciona en las de otros fabricantes. La academia puede participar en desarrollos compartidos y distrutar de los mismos en aplicaciones más complejas. El problema de costo de capital que se presentaba anteriormente puede ser superado en

[1]DEC 3000/300X es una serie de computadoras personales fabricadas por *DEC (Digital Equipment Corporation)*; 3000 indica la serie y 300X es el modelo.

[2]Es una colección de funciones escritas en lenguaje C que permite desarrollar aplicaciones para el sistema *XWindows*.

[3]Es un grupo de bibliotecas de funciones para desarrollar interfaces gráficas; están construidas sobre Xlib.

[4]Consiste de un conjunto de bibliotecas de funciones para la construcción de interfaces de usuario, propuesto por *OSF (Open Software Foundation)*.

primera instancia por la diversificación y participación de la comunidad académica nacional, además del óptimo aprovechamiento de los desarrollos del dominio público a nivel internacional [37].

La integración de sistemas abiertos en este trabajo de tesis es un aspecto importante y se dió a través del uso de herramientas de software de distribución gratuita para el diseño de sistemas y especificaciones para la representación de la información geográfica, las cuales van a permitir el funcionamiento en múltiples plataformas y el seguimiento de estándares para la representación de la información.

1.3. Planteamiento del problema

El diseño de una base de datos es un problema principalmente de índole conceptual el cual se traduce en procedimientos pragmáticos de implementación de software. La metodología inicia con la etapa de recopilación y análisis de requerimientos en donde se establece el modelo de empresa, pasando posteriormente por etapas de construcción de codiseño y refinamiento a través de los distintos niveles y diferentes orientaciones.

Se pretende una solución consistente en un buen diseño que produzca la manufactura de un software (base de datos y sistema) el cual cumpla con las siguientes expectativas:

i) correcto sintácticamente y semánticamente válido;

ii) suficientemente flexible de implantaciones y facilidades de reingeniería;

iii) robusto;

iv) con un buen desempeño para responder eficientemente a poblemas que se plantean al sistema.

1.4. Objetivo general

El objetivo general es desarrollar una metodología y un sistema para automatizar el diseño conceptual de bases de datos y la creación de sistemas a partir de descripciones *XML*. Con base en modelos de desarrollo de software, estándares de información geográfica y técnicas de modelos de datos, se desarrollarán modelos conceptuales de bases de datos y sistemas con los siguientes componentes: *estructural/semántica, interfaz de usuario, operacional/comportamiento.*

1.5. Estructura de la tesis

Esta tesis ha sido estructurada de la siguiente manera (figura 1.1). En el capítulo 1, se ha descrito el panorama general sobre el tema tratado en este trabajo de tesis. En el capítulo 2, se aborda el marco teórico, el cual contiene los conceptos que

fundamentan la propuesta de tesis desarrollada. En el capítulo 3, se describen las tecnologías y herramientas de desarrollo empleadas para diseñar la solución que provee el trabajo propuesto. Por otra parte en el capítulo 4, se explican los estándares para la representación y la organización de los datos geográficos (*OpenGIS, ISO/TC 211*). En el capítulo 5, se plantea el diseño del sistema que implica la arquitectura de solución, la definición de los componentes del sistema y se describe el caso de estudio enfocado hacia *sistemas de información geográfica*. En el capítulo 6, se presenta la descripción de la implementación del diseño propuesto. Finalmente, en el capítulo 7, se explican las conclusiones del trabajo, los resultados obtenidos y el trabajo a futuro.

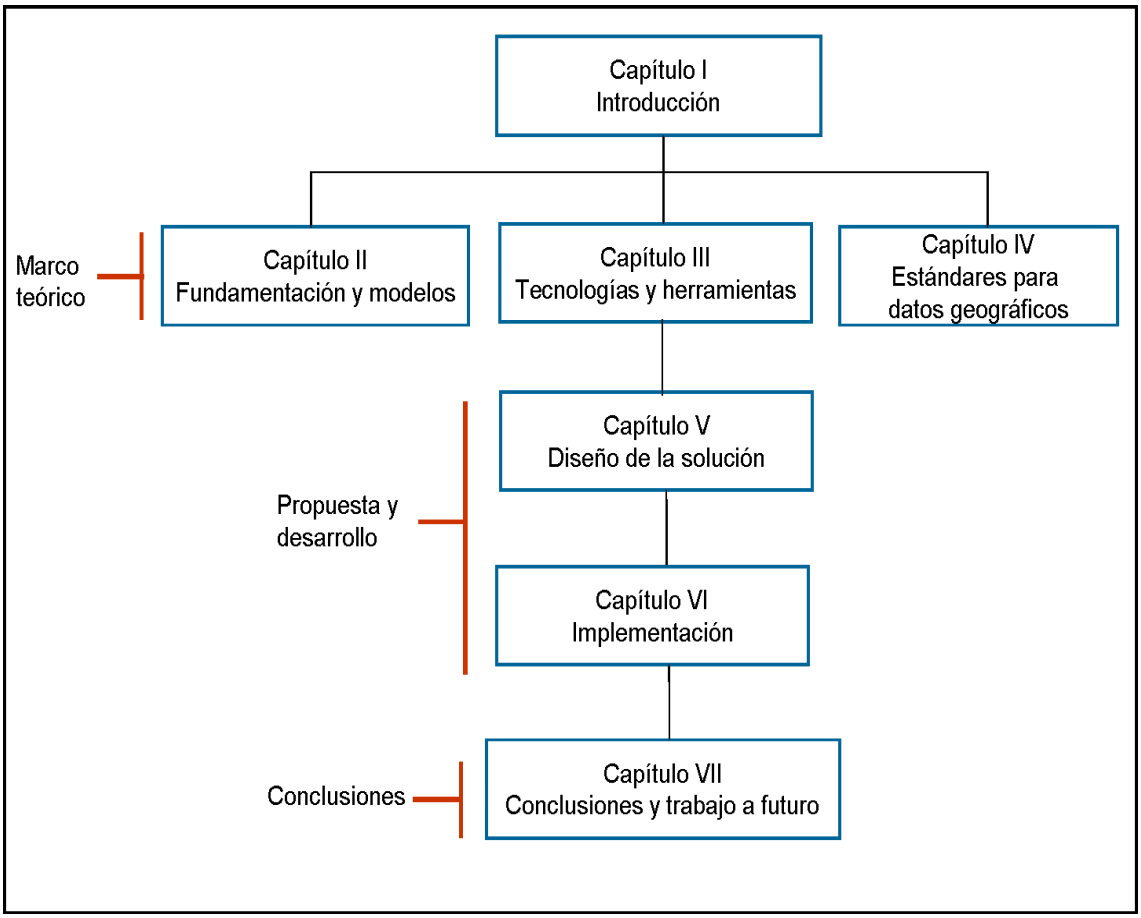


Figura 1.1: Estructura del documento

CAPÍTULO 2

Fundamentación y modelos

En este capítulo exponemos el marco teórico que engloba los conceptos relacionados con el trabajo de tesis desarrollado. Iniciamos con la explicación acerca de los modelos de desarrollo de software e indicamos cómo estos ayudan al proceso de diseño y desarrollo de software (figura 2.1). Después presentamos la definición del modelo de datos, describimos las etapas que integran el diseño de una base de datos, y explicamos también los modelos de datos más utilizados (figura 2.1). Mencionamos los problemas abiertos existentes en ciertas áreas relacionadas con el diseño de bases de datos, la creación de bases de datos objeto relacional y flujos de trabajo, la ingeniería de codiseño continuo, los metadatos y la interoperabilidad de información geográfica. Luego, abordamos la definición y el propósito de un *sistema de información geográfica*, describimos las fases que integran un proyecto de *SIG*, se indica la importancia del uso de bases de datos geográficas y se describe el modelo de datos para la representación de la información geográfica en *SIGs*.

2.1. Modelo de desarrollo de software

En las últimas cinco décadas, la investigación relacionada al desarrollo de software ha buscado la forma de definir abstracciones para crear programas con un mayor enfoque en sus diseños, contemplando en menor grado los ambientes de cómputo subyacentes. En los inicios de la computación, estas "*abstracciones*" comprendían lenguajes y tecnologías de plataforma que exigían al programador pensar en términos de la arquitectura de la computadora más que en la del problema en sí a resolver. Los primeros lenguajes de programación (ensamblador y Fortran) protegían a los desarrolladores de las complejidades de programación con código máquina. También, las primeras plataformas de los sistemas operativos (*OS/360 y Linux*) proporcionaban a los desarrolladores un conjunto de funciones que facilitaban la implementación de sus aplicaciones, de esta manera los programadores no tenía que preocuparse por tener

un amplio conocimiento del *hardware*[1] [36].

Un destacado esfuerzo para la definición de abstracciones inició en el año de 1980 con la invención de las herramientas *CASE*, las cuales se enfocaban en la creación de herramientas y métodos de desarrollo de software que permitían a los desarrolladores expresar sus diseños en términos de representaciones de programación gráfica con propósitos generales, tales como máquinas de estados finitos, diagramas de estructura y diagramas de flujo de datos [36].

Una característica importante de la *CASE* es su orientación para permitir un análisis minucioso de manera gráfica de los programas, que implican una menor complejidad comparada con la de los lenguajes de programación convencionales de propósito general. En estos lenguajes tal complejidad se presenta a través de la corrupción y pérdida de memoria (en lenguajes como C). Otra característica se define por la intención de simplificar mecanismos de implementación de las representaciones gráficas con el objetivo de reducir el esfuerzo manual de codificación, depuración y migración de programas [36].

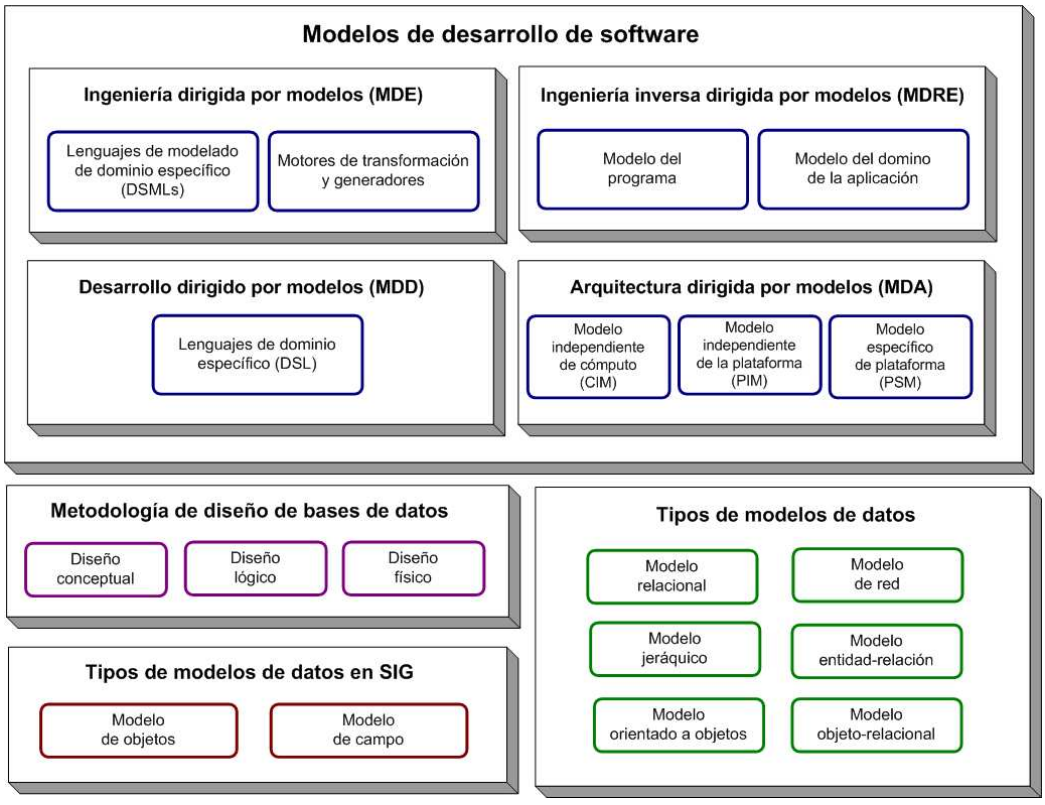


Figura 2.1: Modelos de desarrollo de software y tipos de modelos de datos

[1]Conjunto de los componentes que integran la parte material de una computadora e.g. tarjetas, placas, unidades de disco, monitor, teclado, ratón.

2.1.1. Ingeniería dirigida por modelos

Los avances en lenguajes de programación y plataformas entre 1980 y 1990 han aumentado el nivel de abstracción de software disponible para los desarrolladores, y esto ha permitido avanzar en la investigación relacionada con la creación de herramientas *CASE*. Una muestra de esto es que actualmente los desarrolladores usan leguajes orientados a objetos (*C++*, *Java* o *C#* más que *Fortran* o *C*) para implementar diversas aplicaciones. Además, actualmente el uso de bibliotecas de clases reutilizables y la aplicación de *marcos de trabajo (frameworks)*[2] minimizan la necesidad de reinventar servicios *middleware*[3] de dominio específico. Debido a la maduración de los lenguajes de tercera generación y plataformas reutilizables, los desarrolladores están mejor equipados para protegerse de las complejidades asociadas con la creación de aplicaciones que utilizan tecnologías anteriores [36].

Sin embargo, a pesar de estos avances, varios problemas permanecen. Uno de los principales problemas es el crecimiento de la complejidad de las plataformas que han evolucionado más rápido que la capacidad de los lenguajes de propósito general. Actualmente, plataformas de *middleware* populares, como *J2EE*, *.NET* y *CORBA*, contienen miles de clases y métodos con muchas dependencias complicadas, que requieren de un considerable esfuerzo para programar y armonizar adecuadamente los componentes. Además, estas plataformas suelen evolucionar rápidamente, lo cual conlleva a que los desarrolladores requieran de un esfuerzo considerable para migrar manualmente el código de las aplicaciones a diferentes plataformas o a nuevas versiones de la misma plataforma.

Otro problema relacionado es que el código de la mayoría de las aplicaciones y plataformas aún es escrito y mantenido manualmente utilizando lenguajes de tercera generación, lo cual implica exceso de tiempo y esfuerzo particularmente para la integración de actividades claves, como el despliegue del sistema, la configuración y la garantía de calidad. Hoy en día, es difícil escribir código *Java* o *C#* que despliegue correctamente y de manera óptima sistemas distribuídos a gran escala con cientos o miles de componentes de software interconectados. Gran parte de esta complejidad se deriva del espacio semántico entre el intento de diseño y la expresión de esta intención en miles de líneas de código o descripciones *XML* (*eXtensible Markup Language*) cuya sintaxis no transmite la semántica de dominio ni el propósito del diseño [36].

Debido a estos problemas, la industria del software está llegando a un límite donde la complejidad de las tecnologías de plataforma de la próxima generación, de servicios Web y de arquitecturas de productos en línea han crecido considerablemente, de tal manera que los desarrolladores pasan años tratando de dominar la *interfaz de*

[2]Son el conjunto de bibliotecas de funciones, cuyo objetivo es ayudar en la construcción de aplicaciones proporcionando una clase o un conjunto de clases [11].

[3]Son componentes de software que ofrecen un conjunto de servicios que hacen posible el funcionamiento de aplicaciones distribuidas sobre plataformas heterogéneas.

programación de aplicaciones (*API* [4]*, Application Programming Interface*) y el uso de patrones (es frecuente que sólo se conozca un subconjunto de las bibliotecas de funciones debido a la frecuencia de su uso).

La carencia de una visión integrada, a menudo obliga a los desarrolladores a implementar soluciones subóptimas que innecesariamente duplican código, violan los principios fundamentales de la arquitectura, complican el desarrollo del sistema y afectan la garantía de la calidad del software [36].

Un enfoque prometedor para abordar tal complejidad y la incapacidad de los lenguajes de tercera generación consiste en expresar conceptos de dominio eficazmente, a través del desarrollo de tecnologías de *ingeniería dirigida por modelos* (*MDE, Model-Driven Engineering*) que combinen los siguientes aspectos [36]:

1. **lenguajes de modelado de dominio específico (DSMLs, Domain-Specific Modeling Languages)**. Estos lenguajes son descritos utilizando metamodelos[5] que definen la relación entre los conceptos de un dominio y especifican correctamente la clave semántica y restricciones relacionadas con estos conceptos de dominio. Los desarrolladores utilizan *DSMLs* para crear aplicaciones empleando elementos de cierto tipo de sistema capturados por metamodelos. El tipo de sistemas definidos con estos lenguajes formaliza la estructura de la aplicación, el comportamiento y los requerimientos dentro de dominios particulares (e.g. radios definidos por software, computación de misiones aviónicas, servicios financieros en línea y administración de almacenes de datos) e incluso dentro del dominio de las plataformas de *middleware*;

2. **motores de transformación y generadores**. Analizan algunos aspectos de los modelos y luego sintetizan diversos tipos de artefactos[6] como código fuente, entradas de simulación, despliegue de descriptores basados en *XML* o representaciones alternativas del modelo. La capacidad de sintetizar los artefactos de los modelos ayuda a garantizar la coherencia entre las implementaciones de la aplicación y el análisis de la información relacionada con la funcionalidad y los requerimientos capturados por esos modelos.

Este es el enfoque de desarrollo de software que usará la herramienta propuesta en el trabajo de tesis. El propósito de la herramienta es la creación de modelos de datos (sección 2.2) y por lo tanto, dentro del esquema de *MDE* corresponde con la parte de *lenguajes de modelado de dominio específico*. El uso del enfoque *MDE* resulta apropiado por las características que a continuación se mencionan:

[4]Es el conjunto de funciones y procedimientos que ofrece cierta biblioteca de funciones para ser utilizado por otro software como una capa de abstracción.

[5]Son descripciones de un modelo que contiene la estructura y el comportamiento necesario para representar un sistema.

[6]Cualquier elemento que resulte del proceso de desarrollo de software.

- representa de manera parcial los sistemas que hacen uso de una notación y una semántica claramente definidas para describir ciertos aspectos de los mismos;
- utiliza modelos que permiten aumentar el nivel de abstracción con que se realizan los diseños, así como el nivel de reutilización de los mismos;
- facilita la comunicación de ideas, ya que éstas se pueden expresar de manera explícita (utilizando una notación gráfica asociada a los conceptos que se modelan);
- disminuye la complejidad en la definición y verificación de sistemas;
- reduce el costo de corrección de errores de los sistemas;
- es posible la generación de código de manera automática a partir del desarrollo de modelos.

2.1.2. Ingeniería inversa dirigida por modelos

La administración de proyectos relacionados con el mantenimiento de software es una actividad difícil. Esta dificultad implica que un administrador de proyectos debe enfrentar los problemas relacionados con el trabajo atrasado de tareas pendientes y los grupos de trabajo deben discutir diversos problemas que se presentan al mismo tiempo durante el proceso de desarrolo de software. Cuando un proyecto incluye programas escritos por un grupo diferente de personas o posiblemente otra empresa, existe el problema adicional de intentar comprender código externo.

Ingeniería inversa es el proceso de comprender el diseño y la producción de un modelo de software de alto nivel de abstracción, con el objetivo de realizar la documentación, el mantenimiento, o la reingeniería. Los administradores de proyecto pueden aplicar ingeniería inversa para manejar mejor el código externo. En muchos proyectos, la tecnología de ingeniería inversa ha sido útil para el equipo de mantenimiento de sistemas de manera que permite conocer mejor la estructura y función de un sistema de software. Los líderes de proyecto opinan que existen dos problemas principales [39]:

1. es difícil o imposible predecir cuánto tiempo requerirá el proceso de ingeniería inversa;
2. no existen normas para evaluar la calidad de la ingeniería inversa que el personal de mantenimiento realiza.

La **ingeniería inversa dirigida por modelos (MDRE, Model-Driven Reverse Engineering)** puede superar estas dificultades. Un modelo es una representación de alto nivel de algún aspecto de un sistema de software. Los ingenieros de software usan modelos para especificar con precisión los diseños de los sistemas antes

de construirlos. En algunos casos, las herramientas de modelado pueden incluso generar parte o todo el código sin programación explícita, propensa a errores. *MDRE* aprovecha estas características de la tecnología de modelado, pero las aplica de manera diferente para tratar los problemas de la administración del mantenimiento de software [39].

En la administración del mantenimiento existe el problema de la incertidumbre que se origina por la falta de conocimiento acerca del momento adecuado en el que se debe aplicar el esfuerzo de la ingeniería inversa. Tal momento se decide en base a las pruebas que se realizan al software. Por lo que el conjunto de pruebas adecuado es aquel definido por pruebas que se ejecutan con éxito, indica que la fase de prueba de desarrollo de software está completa. Entonces, si existen criterios que permitan definir momentos adecuados para aplicar ingeniería inversa, los ingenieros de software pueden recopilar informes anteriores y construir bases de datos de proyectos estadísticos para ayudar a predecir el tiempo y esfuerzo de la ingeniería inversa.

Por lo tanto, dos características idóneas para un modelo de ingeniería inversa son:

- **minuciosidad.** Es la medida en que el esfuerzo de ingeniería inversa abarca todo el sistema de estudio;

- **claridad.** Es la medida en que la ingeniería inversa se enfoca sobre los propósitos del sistema y la forma en que el código cumple con ese propósito.

Como se aprecia, los modelos pueden contribuir a medir los esfuerzos de ingeniería inversa relacionados con minuciosidad y claridad, invirtiendo el proceso de ingeniería inversa. En otra palabras, podemos usar el resultado de la ingeniería inversa para producir una segunda versión del sistema original. Así, la ingeniería inversa produce un modelo de alto nivel del sistema en estudio. Si ese modelo es expresado mediante un lenguaje de especificación formal apoyado por una herramienta de generación de código, podemos producir otra versión del sistema original. *MDRE* maneja dos tipos de modelos, un modelo del programa y un modelo del dominio de la aplicación.

El *modelo del programa* proporciona una traducción de alto nivel de las funciones que el programa calcula. Por lo tanto, proporciona una descripción exacta de los valores del programa, pero a un nivel de abstracción más alto que el código fuente del programa [39].

Un *modelo del dominio de la aplicación* expresa dominios de conceptos, sus relaciones y su significado independientemente del programa. Un dominio de la aplicación es un conjunto de problemas relacionados para los que se han generado soluciones de software.

2.1.3. Desarrollo dirigido por modelos

El **desarrollo dirigido por modelos (MDD, Model-Driven Development)** es un nuevo paradigma que soluciona numerosos problemas relacionados con la composición e integración de sistemas a gran escala a la vez que incrementa los avances en el desarrollo de software (como las tecnologías basadas en componentes *middleware*). *MDD* eleva el desarrollo de software a un nivel más alto de abstracción lo cual es posible con lenguajes de programación de tercera generación [4].

MDD emplea modelos para representar los elementos de un sistema y sus relaciones. Los modelos sirven como entrada y salida en todas las etapas de desarrollo de los sistemas, hasta que estos son generados. Un modelo es escrito en un *lenguaje de dominio específico* (*DSL, Domain Specific Language*) que precisa los detalles del diseño de un programa. Un modelo particular capta información limitada, y un programa es a menudo determinado por varios modelos particulares diferentes. Un modelo puede derivarse de otros modelos por medio de transformaciones, y en este caso la síntesis del programa es el proceso de transformación de los modelos de alto nivel a un programa ejecutable [9].

2.1.4. Arquitectura dirigida por modelos

La **arquitectura dirigida por modelos (MDA, Model-Driven Architecture)** es un estándar de la *OMG* (*Object Management Group*) que aborda el ciclo de vida completo de desarrollo de software que comprende el diseño, el despliegue, la integración y la gestión de aplicaciones mediante el uso de modelos [18]. La *MDA* tiene como objetivo la generación de sistemas de alto nivel, de modelos de sistemas y de requerimientos de modelos evitando gran parte de los cambios manuales concurrentes de los artefactos de software en las diferentes etapas del desarrollo de software [28].

El propósito inherente de la *MDA* es separar la implementación de las especificaciones y, de la funcionalidad del sistema. La propuesta de la *MDA* es crear el código de la aplicación a partir de los requerimientos del modelo, en lugar de escribir el código manualmente, de esta manera se logran evitar las fuentes de errores comunes, y en consecuencia mejorar la calidad de la aplicación resultante.

Las fases que componen el proceso de la *MDA* son [28]:

- **modelo independiente de cómputo (CIM, Computation Independent Model)**. Define los requerimientos del sistema futuro, a través de la construcción de modelos de manera formal;
- **modelo independiente de la plataforma (PIM, Platform Independent Model)**. Precisa la separación de la funcionalidad del sistema de las especificaciones de la implementación. La implementación es modelada en esta fase;

- **modelo específico de plataforma (PSM, Platform Specific Model)**. Determina la especificación de la funcionalidad descrita en *PIM* sobre una tecnología de plataforma.

2.2. Modelo de datos

La gestión de los datos implica tanto la definición de estructuras para almacenar la información como la provisión de mecanismos para manipular la información. La importancia de la información en la mayoría de las organizaciones, que determina el valor de las bases de datos, ha conducido al desarrollo de una gran cantidad de conceptos y técnicas para la gestión eficiente de los datos.

Las bases de datos evolucionan a lo largo del tiempo conforme la información se inserta y borra. La colección de información almacenada en la base de datos en un momento particular se llama un *ejemplar* de la base de datos. El diseño completo de la base de datos se denomina el *esquema* de la base de datos. Un *esquema* de base de datos corresponde a una definición de tipo en un lenguaje de programación [15].

Un *modelo* es una representación simplificada de un objeto de investigación para propósitos de: descripción, explicación, pronóstico o planeación [43].

Un *modelo de datos* es una colección de herramientas conceptuales para describir los datos, las relaciones de los datos, la semántica de los datos y las relaciones de consistencia [15].

Un *modelo de datos* consiste de:

- **objetos**. Son entidades que existen y se manipulan;
- **atributos**. Son las características de los objetos;
- **relaciones**. Forma en que se vinculan entre sí los distintos objetos.

Los objetivos de un modelo de datos son [15]:

- **formalización**. Consiste en definir las estructuras permitidas y las restricciones a fin de presentar los datos en un sistema de información;
- **diseño**. El modelo resultante es un elemento básico para el desarrollo de la metodología de diseño de la base de datos.

2.2.1. Metodología de diseño de bases de datos

Algunos de los sistemas de software utilizan bases de datos de información. En algunos casos, la base de datos existe de forma independiente del sistema de software y en otros se crea para el sistema que se está desarrollando. Una parte importante del modelado de sistemas es definir la forma lógica de la relación entre los datos procesados por el sistema. Por lo tanto, es necesario describir la estructura de una base de datos para definir su modelo de datos; esto implica integrar un grupo de herramientas conceptuales para describir los datos, sus relaciones, su semántica y sus limitantes.

Así, el diseño de una base de datos se descompone en *diseño conceptual, diseño lógico y diseño físico* y forman parte del desarrollo de nuestra solución [3]:

- **diseño conceptual,** parte de las especificaciones de requerimientos de usuario y su resultado es el esquema conceptual de la base de datos. *Un esquema conceptual* es una descripción de alto nivel de la estructura de la base de datos, independiente del *DBMS* (*DataBase Management System*) que se vaya a emplear para manipularla. *Un modelo conceptual* es un lenguaje que se utiliza para describir esquemas conceptuales. El objetivo del diseño conceptual es describir el contenido de la información de la base de datos y no las estructuras de almacenamiento que se necesitarán para manejar esta información;

- **diseño lógico,** proviene del esquema conceptual y da como resultado un esquema lógico el cual es una descripción de la estructura de la base de datos en términos de las estructuras de datos que puede procesar un tipo de *DBMS*. Un *esquema lógico* es una descripción de la estructura de la base de datos, independientemente del *DBMS* que se vaya utilizar para manejarla. Un *modelo lógico* es el lenguaje usado para especificar esquemas lógicos (e.g. modelo relacional, modelo de red y modelo jerárquico). El diseño lógico depende del tipo de *DBMS* que se vaya a utilizar, y no del tipo de base de datos a diseñar;

- **diseño físico,** es parte del esquema lógico y da como resultado un esquema físico. Un *esquema físico* es una descripción de la implementación de una base de datos en memoria secundaria: las estructuras de almacenamiento y los métodos utilizados para tener un acceso eficiente a los datos.

Los modelos conceptuales deben ser buenas herramientas para representar la realidad, por lo que deben poseer las siguientes cualidades [3]:

- **expresividad**: deben tener suficientes conceptos para expresar perfectamente la realidad;

- **simplicidad**: deben ser simples para que los esquemas sean fáciles de entender;

- **minimalidad**: cada concepto debe tener un significado distinto;

- **formalidad**: todos los conceptos deben tener un interpretación única, precisa y bien definida.

2.2.2. Tipos de modelos de datos

Los modelos de datos más utilizados son el modelo relacional, el modelo de red y el modelo jerárquico, ya que han sido capaces de satisfacer con éxito las necesidades (en cuanto al diseño de bases de datos se refiere) de las aplicaciones de administración tradicionales.

- **Modelo relacional**. Este modelo fue propuesto por el Dr. Edgar Frank Codd quien consideraba que los sistemas de base de datos deberían presentarse con los datos organizados en estructuras llamadas relaciones. Una relación es una tabla bidimensional integrada por filas (tuplas) y columnas (atributos). Este modelo alcanza un alto grado de independencia de datos, pero puede perder cierta información semántica sobre el mundo real [31].

- **Modelo de red**. Representa datos y sus relaciones a través de diagramas que contienen registros y vínculos que constituyen las relaciones. Un registro contiene campos que se utilizan para guardar valores individuales que representan la información de la entidad del mundo real [15].

- **Modelo jerárquico**. Es un modelo similar al modelo de red ya que los datos y sus relaciones se representan por medio de registros y vínculos, respectivamente; la diferencia radica en que los registros están organizados como un conjunto de árboles [15].

Las técnicas descritas anteriormente presentan algunas deficiencias cuando se trata de aplicaciones más complejas o sofisticadas como, por ejemplo, el diseño y la fabricación de ingeniería (CAD (*Computer-Aided Design*) / CAM (*Computer-Aided Manufacturing*), CIM *(Computer Integrated Manufacturing)*), los experimentos científicos, los sistemas de información geográfica o los sistemas multimedia. Los requerimientos y características de estas nuevas aplicaciones difieren en gran medida de las típicas aplicaciones de administración: la estructura de los objetos es más compleja y las transacciones son de larga duración; se necesitan nuevos tipos de datos para almacenar imágenes y textos. Por ello, se crearon nuevos modelos que permitieran la representación y el manejo de datos de sistemas con mayor grado de complejidad que las aplicaciones tradicionales. Los modelos creados son explicados en los siguientes puntos.

- **Modelo entidad-relación**. En 1976, Peter Chen propuso el modelo entidad-relación, el cual adopta una representación más natural del mundo real que consiste de entidades, los atributos de éstas y las relaciones entre esas entidades, incorporando la semántica de los datos en el modelo de datos [21].

  Una entidad puede ser identificada con precisión (una persona específica, compañía o evento). Una relación es una asociación entre estas entidades, (e.g. "padre-hijo").

- **Modelo objeto-relacional**. A finales de la década de 1990, surgió el modelo "objeto-relacional" como una extensión del modelo entidad-relación. Considera los principios de herencia, abstracción, encapsulación, modularidad y constituye la base para la creación de bases de datos objeto-relacional [1].

- **Modelo orientado a objetos**. El modelo de datos orientado a objetos es una extensión del paradigma de programación orientado a objetos. Los objetos de un programa orientado a objetos son análogos a las entidades que se utilizan en las bases de datos orientadas a objetos con la diferencia de que los objetos desaparecen cuando el programa termina su ejecución, mientras que los objetos de la base de datos permanecen. A esto se le denomina *persistencia* [22].

Se eligió el modelo entidad-relación para el desarrollo de nuestra solución porque su fundamento teórico y funcionamiento es el que mejor se adecúa a la estructura de los datos obtenidos de las cartas del INEGI (*Instituto Nacional de Estadística Geografía e Informática*) para realizar el diseño lógico de la base de datos que será nuestro caso de estudio (ver sección 5.4).

2.2.3. Problemas abiertos

SAT (Satisfiability)

Uno de los problemas en las ciencias de la computación, específicamente en el campo de la complejidad computacional es el problema de *SAT*. El problema de *SAT* consiste en saber si una fórmula booleana dada, es verdadera para alguna asignación de sus variables. Este problema fue introducido por S.A. Cook, y desde entonces se ha analizado en diversas áreas y se ha relacionado con una amplia cantidad de aplicaciones computacionales [24].

Los problemas *SAT* son problemas centrales de la lógica matemática y de la teoría de la computación. Estos son fundamentales para la solución de varios problemas en razonamiento automático, diseño y manufactura asistida por computadora, planificación, visión computacional, bases de datos, robótica, diseño de circuitos integrados, arquitectura y redes de computadoras, entre otros [24].

La solución del problema de *SAT* sería aplicable en el diseño de bases de datos para:

- comprobar la consistencia de una base de datos;

- determinar si un conjunto de restricciones de integridad se cumplen bajo determinadas condiciones o no es un problema indecidible respecto a la coherencia de un conjunto de fórmulas lógicas;

- organizar el conocimiento.

Procesos para el diseño de lenguajes

Las primeras aplicaciones de bases de datos tenían una funcionalidad sencilla. Las transacciones eran simples secuencias de acciones de lectura y escritura sin condiciones. La funcionalidad de las aplicaciones de bases de datos se basó en funciones genéricas. Con la aplicación de bases de datos en diversas áreas, los procesos se volvieron más complejos y se apreció que los datos y los procesos son elementos dependientes. Así, los objetos se introdujeron con el fin de especificar los datos en combinación con los procesos. Un objeto encapsula datos y procesos, para representar y manipular la información estructurada con estos elementos se han desarrollado sistemas de bases de datos orientadas a objetos.

Así, un objeto encapsula datos y procesos, este concepto que ha sido útil para el desarrollo de sistemas de bases de datos.

Se pueden distinguir dos puntos para la comprensión de la tecnología de bases de datos orientadas a objetos [42]:

- las bases de datos contienen un gran número de objetos y proporciona al usuario una funcionalidad para hacer frente a la gestión de una gran cantidad de objetos. Los objetos se pueden definir en base a un tipo de sistema y se agrupan en clases, éstos se pueden manipular de una manera eficiente;

- las clases y los objetos son heterogéneos. Las clases contienen objetos que tienen características comunes. Los objetos en una clase tienen diferentes estructuras. Un sistema administrador de base de datos es necesario para proporcionar todos los mecanismos para una gestión eficiente.

Los objetos son muy complejos. La base de datos almacena un número reducido de objetos complejos. La funcionalidad definida para un objeto es diferente para casi cualquier objeto, y por lo tanto no puede ser definida de manera generalizada. En este caso necesitamos un lenguaje de especificación que permita el mapeo de especificaciones de objetos a eficientes estructuras de implementación. Los manejadores de bases de datos objeto-relacional comerciales muestran que este enfoque puede basarse en una tecnología de base de datos que no es menos eficaz que la tecnología de base de datos relacional. De esta manera, las tareas complejas se especifican en base a flujos de trabajo (*workflows*). Los flujos de trabajo son actividades relacionadas con el área de coordinación de la ejecución de múltiples tareas realizadas por diferentes procesos. Aún con esta tecnología, todavía existen varias cuestiones que es necesario resolver en el proceso de diseño de lenguajes como son [42]:

- se requiere un lenguaje de especifiación de interacción;

- los procesos de lenguaje deben cubrir la especificación de procesos, así como la especificación de flujos de trabajo;
- los diferentes lenguajes deben estar bien integrados;
- los lenguajes necesitan un simple mapeo a conceptos de implementación eficientes.

Ingeniería de codiseño continuo

El modelo conceptual se entiende como el modelado de las estructuras, los procesos y la interacción. Este codiseño de requerimientos estructurales, operacionales y de interfaz proporcionan un medio más adecuado para un modelado completamente integrado. El modelado de flujos de trabajo permite la formalización de las actividades relacionadas con la coordinación de la ejecución de múltiples tareas realizadas por diferentes agentes de procesos o sistemas de software. Una tarea define algún trabajo por hacer. Los usuarios realizan tareas las cuales interactúan con el sistema a través de interfaces [42].

El proceso de diseño de base de datos se limita actualmente a un proceso de diseño simplificado. Cuando el esquema de base de datos es diseñado e implementado, los cambios aparecen después de que el sistema de base de datos está en funcionamiento. Así, el esquema cambia frecuentemente, el proceso se repite varias veces y el esquema es cada vez más complejo, por ello es necesario brindarle mantenimiento. La ingeniería de codiseño de bases de datos permite, en el proceso de explotación de una base de datos, que se incorporen cambios en el diseño de ésta. Se puede realizar cualquier cambio de las estructuras y los procesos e interacciones se pueden realizar a través del modelado conceptual. Este enfoque permite a la empresa mantener el sistema en funcionamiento con las nuevas y las anteriores aplicaciones al mismo tiempo [42].

Problemas combinatorios

La complejidad combinatoria sólo es conocida por algunas clases de dependencias y sólo desarrollada para el modelo relacional. Sin embargo, los métodos y resultados podrían aplicarse también a otras bases de datos y modelos de bases de conocimientos. La complejidad de las restricciones, las relaciones y los modelos no se considera al momento de desarrollar bases de datos, sin embargo sería útil atenderla en los casos prácticos para evitar problemas de funcionalidad [42].

Metadatos

Los metadatos constituyen información estructurada que describe, explica, localiza o de alguna manera facilita la obtención, el uso o la administración de un recurso de información. Los metadatos en el contexto de sistemas de información espacial digital

se definen como, la información de fondo, la cual describe el contenido, la calidad, la condición y otras características apropiadas de los datos [33].

A los metadatos podemos clasificarlos en tres tipos principales:

- **metadatos descriptivos**. Esta clase de metadatos ayuda a los usuarios, en la búsqueda de un recurso, a distinguir un recurso de otro y a entender el contenido del recurso. En otras palabras, describen un recurso para propósitos tales como descubrimiento e identificación. En esta clase se incluyen elementos como son: título, autor, palabras clave y resumen;
- **metadatos estructurales**. Indican como agrupar objetos compuestos, e.g. la relación entre los artículos, el número y el volumen de una publicación serial o la manera en la cual se ordenan las páginas de un libro para formar secciones, capítulos;
- **metadatos administrativos**. Esta clase de metadatos provee información que ayuda en la administración de un recurso, e.g. la fecha de creación e identidad del creador, el tipo de archivo del que se trata, quienes pueden acceder a la información.

Una razón importante para crear metadatos descriptivos es facilitar el descubrimiento de información relevante, además de ayudar a organizar los recursos electrónicos, facilitar la interoperabilidad y la integración de recursos, proveer identificación digital y apoyar a la preservación de la información [33].

Las funciones de los metadatos en el descubrimiento de recursos son:

- permitir la búsqueda de recursos a partir de criterios de importancia;
- identificar los recursos;
- reunir recursos similares y distinguir los que no son;
- proporcionar información de localización.

Interoperabilidad de datos geográficos abiertos (OGIS)

El advenimiento de Internet y la posibilidad de los *SIG* de ofrecer contenidos a través de la red han catapultado el uso de la información georeferenciada como herramienta fundamental en la toma de decisiones y en la participación comunitaria. Sin embargo, la falta de metodologías bien definidas para la captura de requerimientos del sistema y las arquitecturas cerradas siguen manifestándose como obstáculos en el funcionamiento de aplicaciones *SIG*.

Los *SIG* son sistemas de uso intensivo de datos, multidisciplinarios, dinámicos y complejos que basan su representación de la información en un modelo ambiental de la realidad. Estos modelos distan de ser completamente precisos por no obedecer a una jerarquía única de conocimiento. La proliferación de datos geográficos y la preocupación por su control de calidad hacen deseable la implementación de estándares para el almacenamiento de la información y construcción de metadatos [5]. Existe una organización enfocada al desarrollo de estándares en el manejo de información geográfica denominada *OGIS*.

El *Open GIS Consortium, OGIS* es un consorcio formado por la industria, academia y agencias gubernamentales que intentan construir especificaciones que vayan más allá de la simple normalización de los datos procurando emitir todo un conjunto de recomendaciones que engloben las partes conceptual y técnicas requeridas para el funcionamiento de las aplicaciones [5].

2.3. Sistemas de información geográfica

Los *Sistemas de Información Geográfica* (*SIG*) datan del año de 1970 [43]. Actualmente son vigentes por los problemas teóricos fundamentales relacionados a ellos, por las nuevas tecnologías de la información y por una variedad considerablemente grande de aplicaciones importantes en las que son usados.

Un *SIG* es una tecnología basada en computadoras de propósito general y es útil para almacenar, manejar y explotar datos geográficos en forma digital. Se integra por un conjunto de subsistemas enfocados en la captura, el almacenamiento, el análisis, la visualización y la graficación de diversos conjuntos de datos espaciales o georeferenciados[7]. Tales subsistemas en conjunto obtienen información territorial para resolver problemas de planificación, gestión y toma de decisiones apoyándose en la cartografía[8]. Un *SIG* es un sistema geográfico porque permite el modelado espacial: la creación de mapas y el análisis espacial. Es un sistema de información porque orienta en la gestión de la información, procesa datos almacenados previamente y permite eficaces consultas que son útiles para añadir valor a la información gestionada y es un sistema informático con hardware y software especializados que procesan los datos obtenidos de diversas fuentes (e.g. fotografía aérea, metadatos, imágenes de satélite, *INEGI*, datos de procedencia comercial y gratuita) [12]. Su característica principal es el manejo de datos complejos basados en datos geométricos (coordenadas e información topólogica) y datos de atributos (información nominal) el cual describe las propiedades de los objetos geométricos tales como puntos, líneas y polígonos. La codificación de la información en datos requiere de estructuras y formatos adecuados para su almacenamieto en una base de datos, la cual podrá tener una descripción en

[7]Se refiere a datos acerca de los fenómenos geográficos relacionados con su ubicación espacial en la superfice terrestre [30].

[8]Disciplina que trata del trazado de mapas.

un nivel de abstracción más alto si se usa una base de metadatos [43].

El núcleo de software de los *SIG* es el *DBMS*. Tal sistema debe tener la capacidad de almacenar y gestionar las entidades asociando su representación geométrica con su representación nomimal constituida por atributos. La base para tener una independencia con la plataforma de implantación es el modelo lógico de datos el cual es una interfaz entre el modelo conceptual entidad-relación y el modelo físico donde se mantienen las estructuras de datos y almacenamiento. La figura 2.2 presenta los componentes mencionados anteriormente que integran un *SIG*.

Las aplicaciones de un *SIG* aumentan día a día ya que sirve para la elaboración de mapas temáticos y composiciones cartográficas al añadir gráficos con tablas enlazadas a los mapas. La creación de mapas activos con posibilidades de agregar componentes multimedia (e.g. video, imágenes, animaciones) y combinado con la tecnología de la Web posibilita la generación de escenarios y realidad virtual, dibujos en perspectiva realista y vuelos virtuales, trazado de rutas (e.g. comerciales, de emergencia, red de alcantarillado). Un *SIG* permite crear inventarios de recursos naturales y humanos (catastros), hacer investigación de los cambios producidos en el medio ambiente, realizar cartografía de usos de suelo y prevenir de incendios.

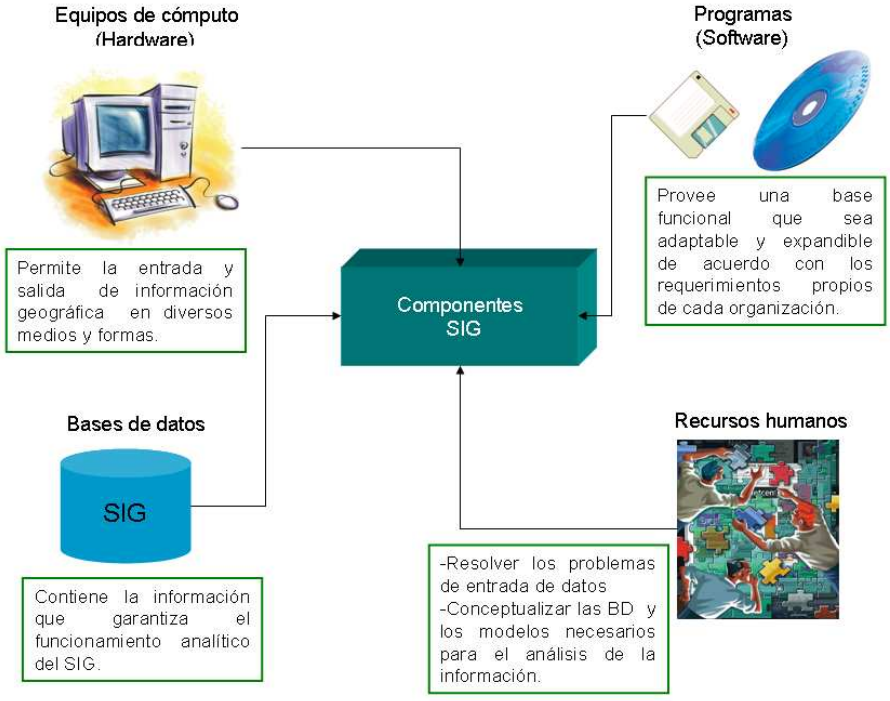


Figura 2.2: Componentes de un SIG

2.3.1. Fases de un proyecto SIG

Por su carácter multidisciplinario, es difícil dividir en fases un proyecto *SIG* obedeciendo a un campo del conocimiento específico. Algunos enfoques combinan elementos del método de las ciencias y de la gestión de proyectos de ingeniería junto con el esquema clásico de desarrollo de aplicaciones de software. Sin embargo, se ha notado que las fases de un proyecto *SIG* obedece al cumplimiento de las unidades funcionales que se presentan a continuación [5]:

- **entrada de datos**. Los *SIG* se alimentan de datos tomados por sensores remotos (e.g. radares e imágenes de satélite), sistemas de posicionamiento global (*GPS, Global Positioning System*) y digitalizaciones sobre información analógica existente. Mediante este proceso, muchas veces manual, se prepara la información para alimentar un sistema específico, se discretizan los datos continuos y se valida la información para que cumpla con relaciones topológicas (e.g. vecindad, contenido e intersección). También se efectúa una valoración de la calidad de la información y se inicia la construcción de sus metadatos;

- **modelado de datos**. Se construye el modelo conceptual de la información dandole sentido lógico a los datos recolectados. La información es almacenada en capas o temas, los cuales han priorizado un atributo particular del territorio estableciendo particiones sobre la continuidad de los fenómenos espaciales de acuerdo a un valor o rango de valores. Este modelado organiza la información preparándola para ser almacenada en una base de datos;

- **manipulación de datos**. Es en esta fase donde toda la ingeniería *SIG* entra en acción a través de los modelos en álgebra de mapa raster, generalizaciones, intersecciones, cruces, análisis de adyacencias, necesarios según alguna metodología específica para obtener los resultados deseados que propiciarán que el proyecto cumpla con sus objetivos;

- **presentación de resultados**. Finalmente, los resultados deben llegar a los usuarios interesados, para alimentar así procesos posteriores. En esta etapa se han ubicado generalmente las ciencias relacionadas con la tecnología informática construyendo aplicaciones que permitan la visualización, la consulta y la organización de la información resultado de un proyecto para caracterizar un territorio.

2.3.2. Bases de datos geográficas

La construcción de una base de datos geográfica implica un proceso de abstracción para pasar de la complejidad del mundo real a una representación simplificada que pueda ser interpretada por las computadoras actuales. Este proceso de abstracción tiene diversos niveles y generalmente comienza con la concepción de la estructura de

la base de datos, generalmente en capas temáticas[9]; en esta fase y dependiendo de la utilidad que se vaya a dar a la información a procesar, se seleccionan las capas temáticas a incluir.

Al separar la información en diferentes capas temáticas, ésta es almacenada independientemente, permitiendo trabajar con ellas de manera rápida y sencilla, facilitando al programador la posibilidad de relacionar la información existente a través de la topología de los objetos. Las *bases de datos geográficas* consisten de un conjunto de datos que se agrupan en capas, de manera que cada capa representa un tipo de información geográfica. Los *SIG* obtienen y procesan esta información obtenida de la base de datos para combinar esas capas en una sola imagen mostrando que las capas están relacionadas entre sí. Una base de datos geográfica puede incluir un gran número de capas. También, se pueden generar imágenes de un área en dos o tres dimensiones, representando elementos naturales (e.g. colinas o ríos) junto a elementos artificiales (e.g. carreteras, tendidos eléctricos, núcleos urbanos). En bases de datos geográfica la relación entre un objeto geográfico (topología) a un elemento de la realidad (e.g. ríos, carreteras, montañas) se reduce a cuestiones más sencillas como el saber cuáles líneas forman una determinada carretera.

Convertir los productos de información geográfica (e.g. cartas, reportes, estudios) de forma analógica a forma digital, implica la necesidad de considerar que los mecanismos de percepción y análisis de información digital difieren de los tradicionales. Los productos convertidos serán procesados por computadoras y aunque pueden ser visualizados en monitores, su análisis se realiza fundamentalmente por la combinación de métodos de análisis geométrico, métodos estadísticos y consultas de bases de datos.

Los datos que integran esta información se clasifican, de acuerdo con su naturaleza, en tres tipos: *vectorial, raster y alfanuméricos*. El tipo *vectorial* contiene los datos provenientes de las cartas que en diferentes escalas y temas se han producido; el tipo *raster* contiene la información de tipo imagen (e.g. imágenes tomadas por satélites y modelos digitales de elevación). El tipo alfanumérico comprende los datos tabulares y textuales (e.g. reportes de campo) [16].

Los datos geográficos tienen cuatro características principales [30]:

1. posición geográfica (coordenadas);
2. atributos (valores de los datos);
3. relaciones topológicas;
4. componentes de tiempo.

[9]Es una colección de datos que describe cierto tema (e.g. vegetación, hidrología, suelos, pendientes).

Las tres primeras características forman parte del contenido de la base de datos geográfica de nuestro caso de estudio, sin incluir componentes de tiempo. Una vez almacenadas en un *SIG*, estos datos se pueden clasificar en tres categorías principales [30]:

- **datos convencionales**: atributos alfanuméricos tradicionales, manipulados por *DBMS* convencionales;
- **datos espaciales**: atributos que describen la geometría de fenómenos geográficos;
- **datos gráficos**: atributos que almacenan imágenes (e.g. fotos).

El rápido crecimiento de los *SIG* se ha traducido en un gran número de sistemas, cada uno de los cuales tiene su propio manejo de datos y almacenamiento de características. La mayoría de los *SIG* siguen basados en manipuladores de datos espaciales e imágenes usando un gestor de archivos, sin ningún tipo de servicio de bases de datos.

El acoplamiento del *DBMS* a los requerimientos del procesamiento de datos en un *SIG* se ha hecho en función de tres arquitecturas [30]:

- **sistemas propietarios**. Una base de datos de propósito especial está estrechamente unida a los módulos de procesamiento de datos espaciales. Los usuarios no pueden acceder a la base de datos directamente y los datos no pueden ser migrados a otro *DBMS*;
- **sistemas de capas**. Un *DBMS* común se utiliza como base para las funciones de acceso de datos espaciales. Los usuarios pueden acceder a la base de datos directamente y pueden ser portados a otros sistemas;
- **sistemas extensibles**. Usan las facilidades proporcionadas por el *DBMS* relacional extendido u orientado a objetos introduciendo la dimensión espacial en el sistema.

2.3.3. Modelo de datos en SIG

Información y datos es una dicotomía que se presenta entre la concepción del mundo real y su representación simbólica codificada. La complejidad de la realidad geográfica da lugar a diversas forma de adquirir la información asociada a los eventos espacio-temporales. Su codificación a representaciones finitas simbólicas (e.g. atributos de valores de suelo), se conforma de estructuras más complejas (e.g. matriz de variables del suelo), las cuales son asociadas a una localización espacial que varía con el tiempo. Las bases de datos geográficas consideran dos clases principales de datos: *espaciales* (*localización o gráficos*) y *nominales* (*atributos no gráficos*). La componente espacial de un *SIG* se describe con frecuencia como una serie de capas o coberturas,

donde cada una contiene una serie de rasgos que son relacionados funcionalmente con un tema. Los objetos pueden ser puntos, líneas o áreas se describen por atributos que toman valores dentro de un dominio que tiene asociado escalas, operaciones y predicados [43].

En el contexto de un *SIG*, un *modelo de datos* es la abstracción y la representación de los fenómenos del mundo real de acuerdo a un esquema conceptual formalizado que es aplicado generalmente usando las primitivas geográficas (e.g. líneas, puntos y polígonos) [47].

Para poder describir la fisionomía espacial geográfica, la realidad debe ser concebida mediante un modelo espacial de datos, el cual puede ser descrito en dos niveles: *modelo de objetos* y *modelo de campo*.

El *modelo de objetos* representa fenómenos geográficos que pueden ser individualizados (poseen una identidad y características que pueden ser descritas a través de un conjunto de atributos) y está constituido por:

- **punto**. Un objeto cero-dimensional que especifica la localización geométrica por medio de un conjunto de coordenadas;
- **línea**. Un objeto uni-dimensionl que es un segmento de linea determinado entre dos puntos;
- **área**. Un objeto bi-dimensional que es continuo, acotado y puede incluir su frontera;
- **celda**. Es un objeto bidimensional que representa un simple elemento en un espacio discreto referenciado a una superficie continua;
- **píxel**. Es un objeto gráfico bidimensional que se define como el elemento más pequeño indivisible de una imagen;
- **símbolo**. Elemento gráfico que representa alguna característica de los puntos sobre un mapa.

El modelo de campo representa la variación espacial de una simple variable mediante una colección de objetos discretos. Un campo puede ser asociado con una variable medida sobre una escala continua o discreta. Una base de datos geográfica puede comprender varios tipos de modelos de campo de los cuales seis son comúnmente manejados en un *SIG*.

- **Muestreo irregular de puntos**. La base de datos contiene un conjunto de tripletas (x, y, z) que representan valores de puntos de una variable en un conjunto finito de localizaciones irregularmente distribuidas.

- **Muestreo regular de puntos**. El muestreo de puntos es un arreglo que se mapea sobre un enrejado regular. Este campo tiene similitud con el anterior excepto por el espaciamiento regular en donde se tienen localizadas las mediciones

- **Contornos**. La base de datos contiene un conjunto de líneas, cada una representada por un conjunto de parejas ordenadas (x, y) que están asociadas a un valor z. Los puntos en cada conjunto están linealmente conectados como en las líneas de contorno cartográficas de una misma elevación.

- **Polígonos**. El área es dividida en un conjunto de elementos polígonales, de tal forma que cualquier localización que cae dentro de un polígono tiene asociado un valor y las fronteras son definidas por un conjunto de parejas ordenadas (x, y) de puntos (e.g. uso de suelo, área básica de simulación).

- **Enrejado de celdas**. El área es dividida en un enrejado regular de celdas. Cada celda tiene un valor y una variable que se supone tiene un valor para todas las localizaciones dentro de la celda.

- **Red triangular**. En este caso el área es dividida en triángulos. El valor de la variable es especificado en cada vértice del triángulo y se considera que varía linealmente sobre el triángulo (e.g. el modelo de elevación *TIN*, *Triangular Irregular Network*).

Los modelo de objetos y de campo son aplicados en el desarrollo de nuestra solución para describir las características de la información geográfica de acuerdo a las propiedades geográficas que engloben las entidades que integran la base de datos.

Los aspectos que deben considerarse en relación a los objetos geográficos en los modelos de datos geográficos son [2]:

- **localización y extensión**. Es la ubicación y alcance de las coordenadas (x, y, z) en el sistema de referencia específico. Estos aspectos están representados por cualquiera de los puntos, líneas y polígonos;

- **medida temporal**. Es necesario mantener registro de la existencia y cambio de un objeto a lo largo del tiempo. La existencia y la operación de tiempo deben ser válidos;

- **medida espacial compleja**. Debe ser posible asociar medidas espaciales complejas con objetos. De esta manera medidas de cero, uno, dos y tres dimensiones consiste de e.g. puntos (y multi-puntos), líneas (y multi-líneas), polígonos (y multi-polígonos) y raster debería ser posible. Esto permite a los objetos que estén constituidos de más elementos;

- **valores temáticos**. Un objeto tiene varios atributos que definen sus valores;

- **objetos difusos**. La representación de un objeto difuso está relacionada con su ubicación y valores temáticos. En apoyo a los objetos geográficos, debería ser posible modelar objetos que tengan una extensión y medida, las cuales son características que hasta cierto punto pueden ser referidas por medio de coordenadas (x, y, z). Los objetos pueden poseer valores temáticos asociados o no con una determinada clase;

- **entidades vs. datos basados en campo**. El mundo real puede representarse por entidades totalmente definibles (e.g. carreteras y edificios). En el enfoque basado en campo, el mundo real consiste de atributos los cuales se supone que varían en el espacio como una función continua. Este último sería el enfoque preferido si se representan a fenómenos que impliquen altura y profundidad;

- **generalización**. Se refiere a aspectos de un objeto como la escala y el propósito. Un sólo objeto puede derivarse de varios objetos en una escala mayor y la información sobre todos los objetos que se agregan o no deben ser accesibles;

- **restricciones**. Un objeto puede estar definido por atributos cuyos valores requieran de restricciones en cierto intervalo;

- **funciones**. La función de un objeto está estrechamente relacionada con su representación. Un objeto geográfico puede ser definido de manera diferente dependiendo de la aplicación;

- **identificación del objeto**. Un identificador de objeto único es necesario para el intercambio de datos; si es necesario hacerlo entre diversos repositorios tal identificación es de gran ayuda;

- **calidad de la información**. La calidad describe el valor de los datos. Esta información es importante al evaluar la credibilidad de los datos y debería ser obligatoria.

Los elementos a considerar en las relaciones entre los objetos que definen los modelos geográficos son [2]:

- **relaciones topológicas**. Se refieren a las relaciones y conexiones entre los objetos. Ejemplos de relaciones topológicas binarias son: *contiene, superposición*;

- **relaciones métricas**. Implica la distancia y depende de la posición absoluta de los objetos en relación con un determinado sistema de referencia;

- **relaciones semánticas**. Son relaciones entre los objetos, pertinentes en el plano conceptual, que no son ni topólogicas ni métricas;

- **relaciones parte-de**. Indica que un objeto puede constar de otros objetos;

- **restricciones de la relación**. Son muy importantes y dependen en gran medida del tipo de relación entre los objetos.

Dos aspectos importantes afectan a los objetos:

- **visualización**. La visualización de objetos es fundamental, de tal manera que estos se puedan visualizar en diferentes escalas;

- **cambio**. Define la manera en la que un objeto puede cambiar. La información indica las circunstancias externas que pueden producir un cambio importante.

2.3.4. Semántica de la información espacial

La necesidad del ser humano por el uso de sistemas y aplicaciones que manejen y procesen información georeferenciada ha originado la creación de grupos de investigación en el área de la ciencia de la información geográfica. El trabajo de esta área consiste en obtener representaciones acerca de cómo el ser humano percibe el mundo real por medio de una conceptualización basada en procesos de abstracción y representación de los datos geográficos. La humanidad inició representando el mundo real a través de bosquejos y dibujos, posteriormente con mapas, con la llegada de la era digital, los mapas son la base para el desarrollo de *SIG* (Sistemas de Información Geográfica), los cuales proporcionan mecanismos para almacenar manipular y analizar la información geográfica.

En la actualidad, se ha reconocido ampliamente que la semántica de la información geográfica es crítica para el desarrollo de bases de datos geoespaciales y aplicaciones interoperables. En adición a esto, los *SIG*; así como la tecnología de desarrollo de éstos, deben ser interoperables con otros sistemas y bases de datos. Con la semántica espacial de objetos geográficos, es posible analizar la interacción de diversos fenómenos de un área común, además de poderlos representar adecuadamente a diferentes niveles de detalle de conocimiento, dependiendo del propósito o caso de estudio.

Las bases de datos geográficas son herramientas muy poderosas y utilizadas para manejar, desplegar y procesar la información geoespacial. Estas bases de datos integran *SIG*, los cuales están diseñados para almacenar y procesar los datos geoespaciales que son muy complejos y mixtos. Por tal motivo para evitar cualquier tipo de ambigüedad en el procesamiento e interpretación de estos datos, se debe contar con una buena calidad desde el proceso de entrada hasta su representación.

Las aplicaciones *SIG* son utilizadas para analizar las características de diferentes ambientes geográficos. Sin embargo, estas aplicaciones utilizan diferentes fuentes de datos geoespaciales para lograr sus objetivo; así como diversos formatos, los cuales pueden encontrarse o en diferentes bases de datos geográficas; motivo por el cual la integración de información geoespacial es un problema complejo para llevar a cabo tareas específicas. Por tal razón, el objetivo de realizar una conceptualización del dominio geográfico es solucionar problemas de heterogeneidad e interoperabilidad

semántica debido a la existencia de diferentes fuentes de información.

En la actualidad existen diversos trabajos de investigación enfocados a representación semántica relacionada relacionada con datos geoespaciales. De manera particular existen dos trabajos de doctorado desarrollados en el CIC (Centro de Investigación en Computación) y se mencionan a continuación:

- **representación ontólogica basada en descriptores semánticos aplicacada a objetos geográficos**. El trabajo consiste en una metodología para la representación, recuperación e integración de objetos geográficos por medio de su semántica espacial, haciendo uso de una conceptualización del dominio geográfico que permite generar ontologías de apliación, las cuales pueden ser utilizadas para describir regiones espaciales basadas en descriptores semánticos obtenidos de un esquema conceptual [35];

- **representación semántica de datos espaciales raster**. El desarrollo del trabajo esta enfocado a definir un concepto de semántica espacial en el contexto de los datos espaciales raster, así como una metodología para obtener una representación semántica de un conjunto de datos [41].

De esta manera, el trabajo desarrollado esta enfocado hacia la creación de una herramienta que ayude a la creación de modelos de sistemas y bases de datos, las áreas que enbloban este trabajo son la ingeniería de software y el diseño de bases de datos, pero se utilizó como caso de estudio los *SIG* por lo tanto se tomo como fundamento la misma base teorica y conceptual que los trabajos desarrollados respecto a la descripción y representación semántica de la información espacial, aunque el propósito de la investigación no sea el mismo.

En este sentido, nuestro trabajo utiliza la semántica espacial de objetos como un mecanismo para la correcta representación y descripción de objetos espaciales en una base de datos geoespacial, en donde se puedan distinguir estos objetos por medio de sus características (propiedades) y también a través de un valor de atributo o una primitiva de representación espacial (punto, línea, polígono, etc), sin realizar algún tipo de procesamiento, clasificación o manipulación de información georeferenciada.

CAPÍTULO 3

Tecnologías y herramientas

En este capítulo presentamos de manera general las herramientas de software que se utilizaron para el desarrollo de nuestra solución. Primeramente, se describe la *API Java Swing*, la cual proporciona un conjunto de componentes para el diseño e implementación de aplicaciones de escritorio con interfaces gráficas amigables para el usuario. Posteriormente, se aborda la tecnología *XML* que brinda la funcionalidad de establecer la estructura de los contenidos de los documentos, lo que permite crear archivos de descripciones *XML* estructurados. La transformación de documentos *XML* a otro formato de archivo (en cuanto a presentación) es por medio de estilos y formatos establecidos en hojas de estilo *XSLT* sobre el archivo *XML*. Brevemente se describe el uso de la biblioteca *dom4j* para acceder y manipular documentos *XML*. Exponemos las características primordiales de *PostgreSQL* y la funcionalidad de *PostGIS*. Finalmente, se proporciona información del entorno de desarrollo *Netbeans* para el diseño de aplicaciones con tecnología Java de manera sencilla y rápida. En la figura 3.1 se presenta un diagrama de las herramientas de software utilizadas.

3.1. API Java Swing

Para diseñar aplicaciones que hagan uso de interfaces gráficas (e.g. ventanas con controles, etiquetas, cajas de texto, botones, barras de desplazamiento), Java proporciona una biblioteca de clases denominada *JFC* (*Java Foundation Classes - clases base de Java*) que comprende un grupo de interfaces para programación de aplicaciones que engloban a *AWT* (*Abstract Window Toolkit*), *Swing*, *Java 2D* y permite añadir gráficos ricos en funcionalidad e interactividad a los programas desarrollados con Java [6]. La figura 3.2 describe las características de la biblioteca de clases *JFC* [27].

Una *interfaz gráfica de usuario* (*GUI, Graphical User Interface*) presenta un mecanismo amigable para interactuar con un programa, ya que al usuario le proporciona

una "apariencia visual" única. El *API* de Java Swing provee distintas clases que permiten desarrollar programas cuya interfaz de usuario contenga componentes que sean consistentes e intuitivos, de manera que los usuarios puedan familiarizarse con un programa incluso antes de utilizarlo. Esto permite reducir el tiempo que se requiere para aprender a usar un programa y permite incrementar la habilidad del usuario para utilizarlo en una manera productiva [14]. Las *GUIs* se crean a partir de componentes denominados *controles o widgets* (*accesorios de ventana*). Un componente de la *GUI* es un objeto con el cual interactúa el usuario mediante el ratón, teclado u otra forma de entrada, e.g. reconocimiento de voz.

Las clases que crean los componentes de la *GUI* se indican en la figura 3.4 y se describen algunos de los componentes que integran la *GUI de Swing* del paquete **javax.swing**. La mayoría de los componentes de Swing están escritos (también denominados componentes *puros de Java*), se manipulan y muestran completamente en Java. Los componentes de Swing son parte de la *JFC* y para obtener información completa de la *JFC*, ver **http://java.sun.com/docs/books/tutorial/uiswing/index.html** [14]. En la figura 3.3 se presentan todos los componentes *Swing* que son utilizados en el desarrollo de *GUIs*; algunos de ellos se usaron para la implementación de la interfaz gráfica de nuestra solución. La biblioteca de clases *Swing* resulta conveniente para el diseño e implementación de *GUIs* por las siguientes características:

- basa sus componentes en la arquitectura *MVC* (*Modelo-Vista-Controlador*) [6], esta arquitectura permite una clara separación entre los componentes de un programa y ofrece un enfoque de desarrollo muy apegado a los entornos gráficos de usuario diseñados con el paradigma orientado a objetos;

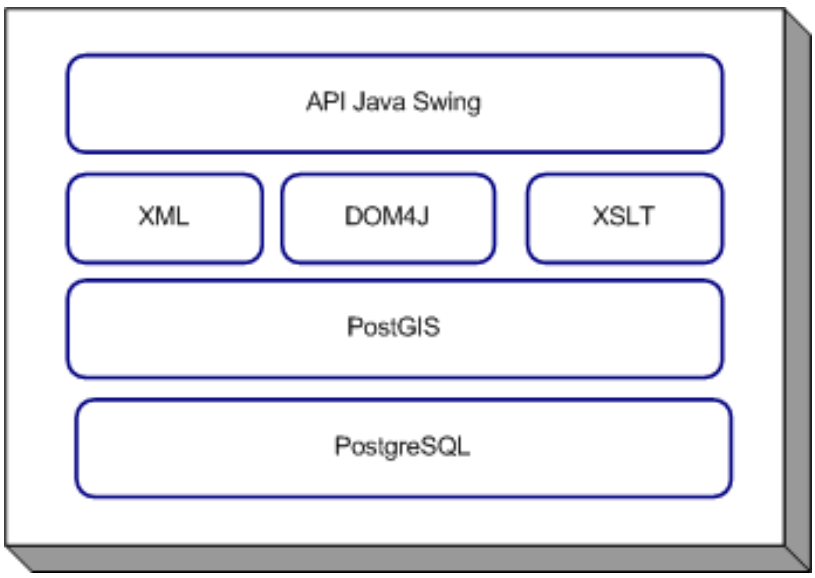


Figura 3.1: Herramientas de software utilizadas en esta tesis

Característica	**Descripción**
Componentes GUI Swing	Incluye botones, paneles de división y tablas entre otros. Ofrece muchos componentes útiles para la clasificación, impresión y descripción.
Soporte de aspecto acoplable	El diseño de aplicaciones Swing es acoplable ya que permite la opción de cambiar el aspecto de la interfaz gráfica a través de paquetes que están disponibles en la *API* de Java.
Accesibilidad de la *API*	Permite que las tecnologías de asistencia (e.g. lectores de pantalla y pantallas Braile) puedan obtener información de la interfaz de usuario.
API Java 2D	Permite a los desarrolladores incorporar fácilmente gráficos en 2D de alta calidad, textos, imágenes en aplicaciones y *applets*. Java 2D incluye extensas *APIs* para la generación y el envío de alta calidad de salida a los dispositivos de impresión.
Internacionalización	Permite a los desarrolladores crear aplicaciones que puedan interactuar con los usuarios en todo el mundo en sus propios idiomas y convenciones culturales.

Figura 3.2: Características de las clases base de Java

- ofrece un conjunto de componentes escritos en Java con una mayor funcionalidad y lógicamente independientes de la plataforma;
- cuenta con una amplia variedad de componentes (e.g. botones, cajas de texto, tablas, diálogos);
- proporciona un aspecto modificable *look and feel*[1] para personalizar el aspecto de los componentes (e.g. botones, cajas de texto, listas, etiquetas) de las interfaces de usuario.

[1]Es la apariencia visual que se establece a una *GUI* basada en componentes Swing.

3.2. Tecnología XML

La tecnología *XML* provee una definición y estructura de documentos útil para el desarrollo de nuestra solución. La descripción de información en documentos *XML* representa una forma práctica y fácil de manipular los datos de manera estructurada, por lo que en esta sección examinamos esta tecnología.

Publicar un documento en Internet consiste en aplicar a determinado contenido (e.g. texto, imágenes) una serie de formatos para que pueda ser visualizado desde la aplicación generalmente conocida como navegador o explorador (*browser*) de Internet (e.g. Explorer, Firefox, Netscape).

La aplicación de los formatos para la correcta visualización del documento se efectúa por medio de determinadas etiquetas (*tags*) definidas en los denominados lenguajes de marcas o de marcado. *HTML* (*HyperText Markup Language*) es el lenguaje de marcado utilizado para dar formato a los documentos publicados en Internet, popularmente conocidos como páginas Web. *XML* es también un lenguaje de marcas, pero su finalidad no es dar formato a los documentos para su visualización, sino establecer la estructura de los contenidos (*datos*) del documento [46].

A continuación se describen algunas características de la tecnología *XML* [46]:

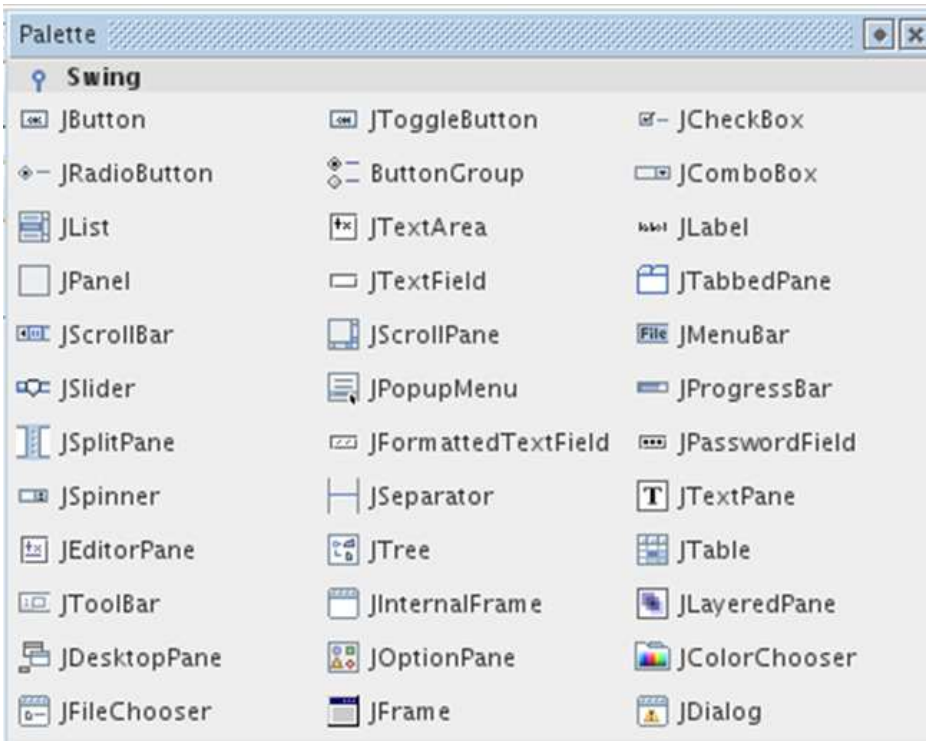


Figura 3.3: Componentes gráficos de Swing

Componente	**Descripción**
JLabel	Espacio en donde pueden mostrarse íconos o texto no editable.
JTextField	Superficie en la que el usuario introduce datos desde el teclado. En esta área se puede mostrar información.
JButton	Espacio que, por medio del ratón puede recibir un clic sobre él, lo que desencadena un evento.
JCheckBox	Componente de la *GUI* que puede o no estar seleccionado.
JComboBox	Lista desplegable de elementos, de los cuales el usuario puede seleccionar cualquiera haciendo clic sobre él o posiblemente escribiendo dentro del cuadro.
JList	Espacio que contiene una lista de elementos, de los cuales el usuario puede seleccionar cualquiera haciendo clic sobre él.
JPanel	Contenedor en el cual pueden colocarse y organizarse componentes.

Figura 3.4: Descripción de algunos componentes de Swing

1. **XML es un lenguaje de marcado para documentos**. Al igual que otros lenguajes como *SGML* y *HTML*. *XML* se basa en el uso de marcas o etiquetas para diferenciar los diversos elementos que pueden existir en un documento. Al contrario de *HTML*, que se utiliza para establecer cómo se han de presentar o de visualizar dichos elementos, con *XML* lo que se define es la estructura del documento, i.e. la perfecta organización de los contenidos existentes en el documento.

2. **XML no utiliza etiquetas predefinidas**. El nombre de las etiquetas utilizadas en la creación de un documento *XML* se define de manera personalizada. Ésta es otra diferencia con *HTML*, el cual es un lenguaje con etiquetas ya definidas (de ahí la X de *eXtensible* de *XML*) no es un lenguaje con marcas ya establecidas, sino que deja absoluta libertad al programador para que defina las que crea convenientes.

3. **XML es un lenguaje muy estricto**. Otra de las principales características de *XML* es el ser estricto; un documento *XML* ha de cumplir las normas de sintaxis y estructuración para que sea reconocido como tal; para ello, el documento ha de estar, al menos, bien formado.

XML es una versión reducida de *SGML*, el cual está basado y especialmente

diseñado para la definición de estructuras de documento y el almacenamiento de datos. Por lo tanto, podemos decir que *XML* se puede utilizar para el desarrollo de dos tipos de aplicaciones:

- **aplicaciones de datos:** los documentos *XML* por definición son "unidades de almacenamiento de datos" y el principal objetivo de este lenguaje consiste en definir la estructura lógica de dichos datos. *XML* es utilizado, por tanto, para la manipulación de información y está más cercano a los mecanismos usados en bases de datos que los utilizados en la publicación de documentos. *XML* también proporciona mecanismos para el intercambio de datos entre aplicaciones;
- **aplicaciones de documento:** las aplicaciones de documento representan la fase final de la creación de una aplicación *XML* y permiten la publicación de documentos *XML* utilizando técnicas de creación de estilos (*CSS o XSL:FO*) y técnicas de aplicación de formatos (*XSL, XSLT*) al contenido de documentos tipo *XML*. Las aplicaciones de documento posibilitan la publicación en prácticamente cualquier tipo de formato conocido e.g. *XHTML o PDF* dependiendo del medio en el cual se vaya a publicar.

XML se convirtió en un lenguaje estándar con la publicación del documento titulado "*eXtensible Markup Language* 1.0. Recomendación *W3C* (*World Wide Web Consortium*)[2]" el 10 de febrero de 1998. Además, proporciona un mecanismo para imponer restricciones al almacenamiento de los datos y a la estructura lógica del documento. Una unidad de almacenamiento de tipo *XML* se puede considerar como un documento *XML* si está bien formado. Un documento *XML* bien formado puede además ser válido si cumple una serie de restricciones definidas en una *DTD* o *XML Schema* (ver tabla 3.1).

XML es, por lo tanto, un lenguaje diseñado para trabajar con datos y estructuras; considerando lo visto anteriormente se puede definir, a los documentos *XML* como **"objetos de datos cuya estructura está bien formada"** [46]. Las tecnologías *XML* son un conjunto de módulos que ofrecen servicios útiles a las demandas más frecuentes por parte de los usuarios. En la tabla 3.1 se muestran las tecnologías relacionadas con *XML*.

3.2.1. Estructura básica de un documento XML

La estructura lógica de un documento *XML* está definida por declaraciones, elementos, comentarios, referencias e instrucciones de procesamiento. Todo documento *XML* consta de dos partes: el prólogo y el cuerpo del documento. La figura 3.5 presenta el contenido de un documento *XML*.

[2]Es un consorcio internacional donde, organizaciones y público en general trabajan conjuntamente para desarrollar estándares Web.

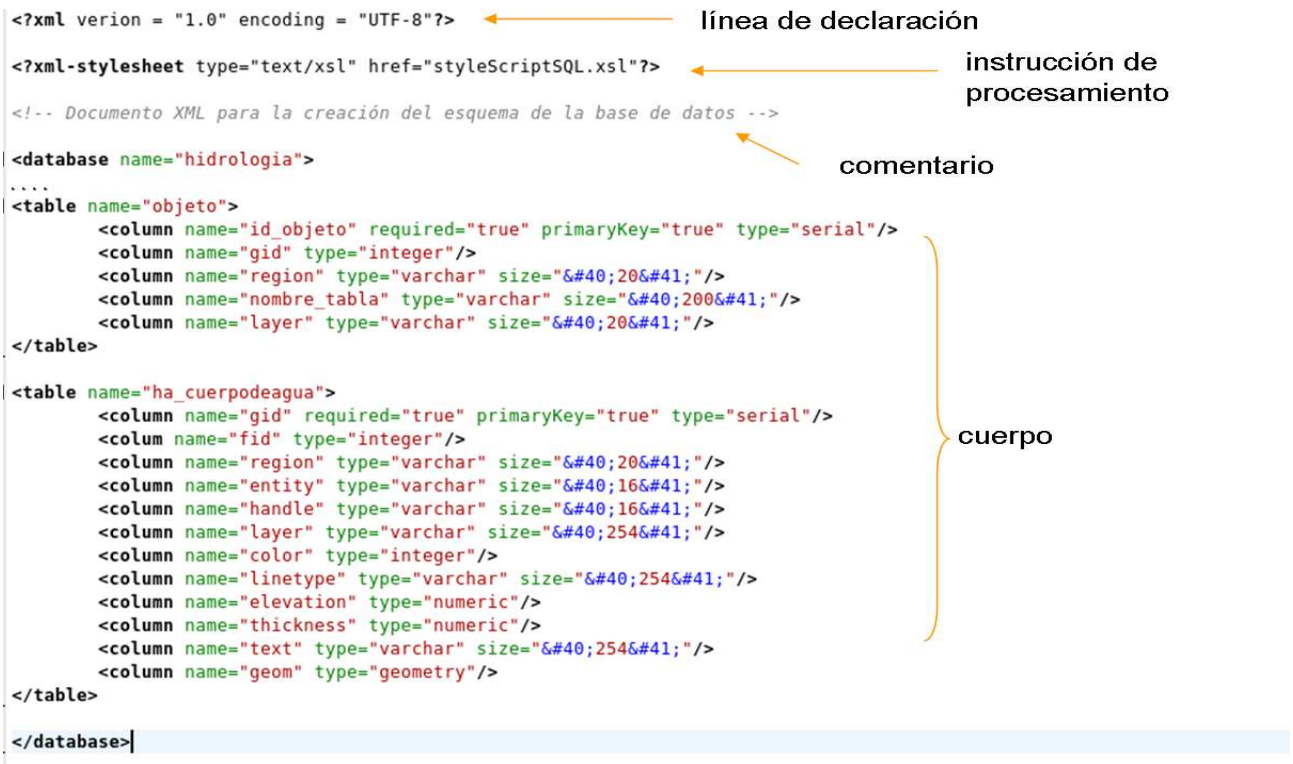
Prólogo

Contiene datos con metainformación utilizada por el procesador *XML* y, generalmente, está formado por:

- **línea de declaración de XML**: es una instrucción que marca el inicio de un documento XML, los parámetros especificados son la versión XML y el tipo de codificación utilizado por los caracteres del documento;
- **instrucciones de procesamiento**: sirven para ligar el documento XML con documentos de hojas de estilo o cualquier otro documento que proporcione formato;
- **comentarios**: son descripciones breves acerca del funcionamiento o significado de algún elemento en el documento XML.

Cuerpo del documento

Representado por el elemento raíz (objeto o entidad documento), el resto de los elementos y contenido.




```
<?xml verion = "1.0" encoding = "UTF-8"?>

<?xml-stylesheet type="text/xsl" href="styleScriptSQL.xsl"?>

<!-- Documento XML para la creación del esquema de la base de datos -->

<database name="hidrologia">
....
<table name="objeto">
        <column name="id_objeto" required="true" primaryKey="true" type="serial"/>
        <column name="gid" type="integer"/>
        <column name="region" type="varchar" size="&#40;20&#41;"/>
        <column name="nombre_tabla" type="varchar" size="&#40;200&#41;"/>
        <column name="layer" type="varchar" size="&#40;20&#41;"/>
</table>

<table name="ha_cuerpodeagua">
        <column name="gid" required="true" primaryKey="true" type="serial"/>
        <colum name="fid" type="integer"/>
        <column name="region" type="varchar" size="&#40;20&#41;"/>
        <column name="entity" type="varchar" size="&#40;16&#41;"/>
        <column name="handle" type="varchar" size="&#40;16&#41;"/>
        <column name="layer" type="varchar" size="&#40;254&#41;"/>
        <column name="color" type="integer"/>
        <column name="linetype" type="varchar" size="&#40;254&#41;"/>
        <column name="elevation" type="numeric"/>
        <column name="thickness" type="numeric"/>
        <column name="text" type="varchar" size="&#40;254&#41;"/>
        <column name="geom" type="geometry"/>
</table>

</database>
```

Figura 3.5: Partes que componen un documento XML

Tecnología	Descripción
XSLT	*(Extensible Stylesheet Language Transformations)* establece cómo efectuar las transformaciones necesarias para aplicar los estilos a documentos *XML* para su publicación.
XSL-FO	*(XSL-Formatting Object)* se usa para definir estilos utilizando los objetos de formato. Éstos objetos de formato definen una tecnología más relacionada con hojas de estilo en cascada *CCS (Cascading Style Sheets)*, básicamente establecen el formato de las páginas y definen los estilos que van a ser utilizados por los diversos elementos que componen el documento resultante.
XPath	*(XML Path)* es utilizado para establecer el camino que se ha de seguir para llegar a un determinado nodo en la jerarquía del árbol de nodos de un documento *XML*.
XLink	*(XML Linking Language)* es un lenguaje que permite insertar elementos en documentos *XML* para crear enlaces entre recursos *XML*.
XPointer	*(XML Pointer Language)* es un lenguaje que permite el acceso a la estructura interna de un documento *XML*, i.e. a sus elementos, atributos y contenido.
XQL	*(XML Query)* es un lenguaje que facilita la extracción de datos desde documentos *XML*. También ofrece la posibilidad de crear consultas flexibles desde documentos *XML*.
XML Schema Definition	*(XML Schema Definition)* es un mecanismo de descripción de documentos que sirve para declarar los diversos objetos que pueden ser utilizados en un documento *XML* y permite definir los elementos con su contenido y atributos para establecer cómo se han de usar para crear la estructura del documento *XML* que utilice determinado esquema; así se podrá efectuar su validación.
DTD	*(Document Type Definition)* es un mecanismo a través del cual se declaran las entidades, notaciones y elementos permitidos en aquellos documentos *XML* que los utilicen. Para los elementos, se definen las etiquetas estableciendo al mismo tiempo cuáles serán sus contenidos y la lista de atributos que deben utilizar.

Tabla 3.1: Tecnologías XML

3.3. Transformación de documentos con XSLT

XSL es un lenguaje creado por el *W3C* cuya finalidad es la creación de estilos y la aplicación de formato a los documentos *XML* para que así puedan ser publicados en cualquier medio. *XSLT* (*Extensible Stylesheet Language Transformation*) es una ampliación de *XSL* y es el lenguaje utilizado para efectuar las transformaciones por medio de las cuales se aplicarán los estilos y formatos establecidos en hojas de estilo, ya sean *CSS* o *XSL* (*XSL-FO*), sobre el documento *XML* para su publicación.

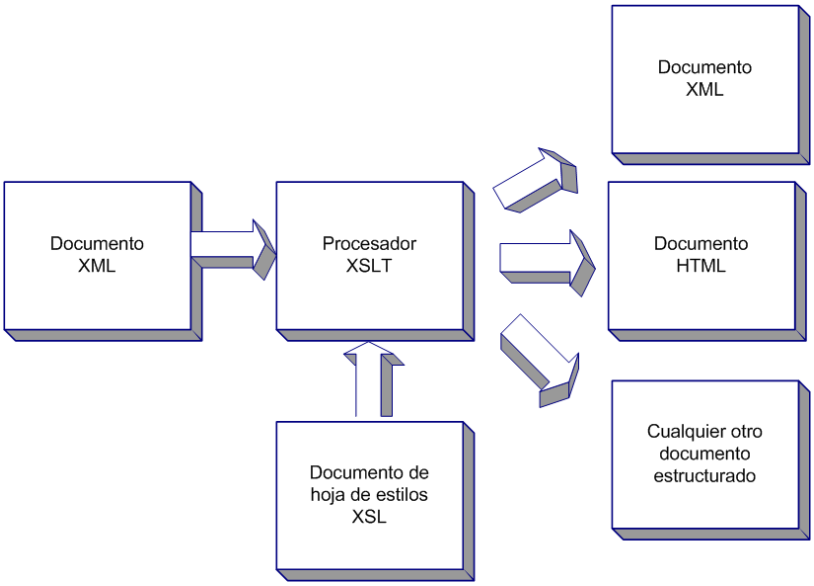


Figura 3.6: Proceso de transformación de documentos XML

El proceso de transformación de un documento *XML* a cualquier otro formato de publicación (texto plano, *HTML*, *XHTML*, *WML*, *PDF*) se realiza a través de un documento *XML* el cual tiene definido una hoja de estilo. Ésta se basa en la definición de plantillas (*templates*) y cada una de las plantillas definidas ha de estar relacionada con algún elemento existente en el documento *XML*. Los elementos relacionados con alguna plantilla serán los objetivos sobre los cuáles se aplicarán las transformaciones (*procesador XSLT*) y formatos definidos en la plantilla para generar el documento final, la figura 3.6 muestra este proceso.

Algunas características del procesador *XSLT* son:

- permite transformar una única fuente de información en varios formatos;
- por ser una aplicación *XML*, su descripción se basa en las mismas reglas que los documentos *XML* solo que la extensión del archivo es *.xsl*;
- es posible separar el contenido de la presentación, permitiendo modificar aspectos visuales fácilmente sin que los contenidos se vean afectados en el proceso;

- permite utilizar las funciones de *XPath* para acceder fácilmente a un determinado elemento del documento *XML*.

3.4. Procesador dom4j

Los procesadores *XML* son módulos de software utilizados para **leer documentos XML y proporcionar acceso a su contenido y estructura**. Deben ser implementados siguiendo las especificaciones establecidas por la *W3C* en las recomendaciones sobre dicho lenguaje. La implementación mínima de todo procesador *XML* está representada por el *analizador* (*parser*) de *XML* encargado de comprobar que el documento que se quiere examinar cumpla las normas establecidas para ser considerado un **documento XML bien formado** [46].

Los procesadores *XML* pueden implementar dos tipos de analizadores:

- **analizadores de buena formación:** comprueban que la estructura del documento sea correcta;
- **analizadores con validación:** en caso de que el documento utilice algún mecanismo de definición de documento, *DTD* o *XML Schema*, comprueban que se utilizan los elementos, sintaxis y estructuras allí definidos;

dom4j es un marco de trabajo *XML* de código abierto para Java que permite leer, escribir, navegar, crear y modificar documentos *XML*, y está integrado por *DOM* y *SAX* [20]. Este marco de trabajo se utilizó para la generación, el acceso y la consulta de documentos *XML* en el desarrollo de nuestra solución.

Algunas características son:

- está diseñado para la plataforma Java con soporte completo para el marco de trabajo de Java;
- implica amplio soporte para *JAXP*, *SAX*, *DOM*, y *XSLT*;
- integra un extenso soporte de *XPath* para facilitar la navegación en documentos *XML*;
- está basado en interfaces Java para implementaciones *plug and play*[3] flexibles.

[3]Tecnología que permite la autodetección de dispositivos en una computadora, con la finalidad de facilitar su instalación.

3.5. Manejador de base de datos PostgreSQL

Postgresql es un sistema administrador de bases de datos objeto-relacional (*ORDBMS, Object-Relational Database Management Systems*) basado en *POSTGRES*, version 4.2, desarrollado en la Universidad de California en Berkeley, en el Departamento de Ciencias de la Computación. *POSTGRES* fue pionero de muchos conceptos que sólo estuvieron disponibles en algunos sistemas de bases de datos comerciales mucho más tarde [13].

PostgreSQL es un proyecto desarrollado con código abierto que soporta el estándar *SQL* y ofrece muchas características interesantes:

- consultas complejas;
- claves foráneas;
- disparadores (*triggers*);
- vistas;
- integridad transaccional;
- control de concurrencia.

Además *PostgreSQL* puede ser ampliado por el usuario de muchas formas, e.g. mediante la adición de nuevas definiciones de:

- tipos de datos;
- funciones;
- operadores;
- funciones de agregado;
- métodos indexados;
- lenguajes procedurales.

Debido a que es de licencia libre, *PostgreSQL* puede ser usado, modificado y distribuido por todo el mundo de forma gratuita para cualquier fin, ya sea privado, comercial o académico. Es considerado como uno de los mejores gestores de bases de datos de software libre. Muchas empresas han iniciado el uso de esta herramienta beneficiándose en la reducción de los costos y en el aumento de la fiabilidad [43]. Dentro de sus características están:

- cumple completamente con la propiedad *ACID* (*Atomicity, Consistency, Isolation and Durability*).

- compatible con ANSI SQL;
- permite integridad referencial;
- implica replicación (soluciones comerciales y no comerciales) lo que permite la duplicación de bases de datos maestras en múltiples sitios de réplica;
- intefaces nativas para ODBC, JDBC, C, C++, PHP, Perl, TCL, ECPG, Python y Ruby, Reglas, Vistas, Triggers;
- define procedimientos almacenados;
- implica soporte nativo SSL (*Secure Socket Layer*);
- incluye lenguajes procedurales;
- autenticación Kerberos nativa;
- soporte para consultas con UNION, UNION ALL y EXCEPT;
- extensiones para SHA1, MD5, XML y otras funcionalidades;
- comprende herramientas para generar SQL portable para compartir con otros sistemas compatibles con SQL;
- funciones de compatibilidad para ayudar a la transición desde otros sistemas menos compatibles con SQL.

Las características mencionadas contribuyeron a la elección de *PostgreSQL* como gestor de base de datos para la generación de la descripción del esquema de la base de datos (*script SQL*), asi como también para el manejo y administración de la información que será almacenada en ella.

3.5.1. PostGIS

PostGIS es una extensión del sistema de bases de datos objeto-relacional *PostgreSQL*, que permite el uso de objetos geográficos. Fue creado por *Refractions Research Inc*, como un proyecto de investigación de tecnología de bases de datos espaciales. Está públicado bajo licencia *GNU*. Con *PostGIS* se pueden usar todos los objetos que aparecen en la especificación *OpenGIS* (*Open Geospatial Consortium*)[4], como puntos, líneas, polígonos, multilíneas, multipuntos y colecciones geográficas [32].

PostGIS es una módulo de apoyo que permite el manejo de información georeferenciada a través de *PostgreSQL*. Al integrarse estas dos partes es posible el manejo de objetos geográficos de forma sencilla y amigable por medio de consultas *SQL* que

[4]Es una organización internacional sin fines de lucro que está liderando el desarrollo de normas para datos geoespaciales [7].

proporcionan el resultado en *coordenadas de 2D y 3D*.

Para integrar *PostGIS* con *PostgreSQL* es necesario instalar el módulo para soporte de objetos geográficos, el cual es un archivo con el siguiente esquema de nombramiento postgis-version.tar.gz (e.g. *postgis-1.2.1.tar.gz*). Para obtener más detalle acerca de la instalación y funcionamiento se puede consultar el sitio de internet y seguir las indicaciones ahi descritas **http://postgis.refractions.net/documentation/**.

Inicialmente el manejo de la información de la base de datos incluye datos geográficos (*coordenadas*), *PostGIS* proporciona soporte para el manejo de este tipo de información a través de *PosgreSQL*, por lo que la manipulación de la información geográfica queda cubierta en el desarrollo de nuestra solución.

3.6. Entorno de desarrollo integrado NetBeans

Un IDE (*Integrated Development Environment*) es un conjunto de herramientas que ayudan al desarrollo de aplicaciones; la mayoría tienen módulos que permiten [26]:

- escribir y editar código fuente;
- ver los errores a medida que se escribe código;
- ver la sintaxis del código resaltado;
- automatizar tareas repetitivas;
- compilar código;
- usar la función arrastrar y soltar para facilitar la construcción de interfaces gráficas de usuario.

Netbeans es un *IDE* desarrollado por el proyecto de código abierto del mismo nombre que permite escribir, compilar, depurar y ejecutar programas, fundado por la empresa de software *Sun Microsystems*. Esta disponible libremente y proporciona compatibilidad integral con *Java EE5*, el estándar de la industria para el desarrollo de aplicaciones Java del lado del servidor, portátiles, robustas, escalables y seguras. Permite que el desarrollo de aplicaciones Java de tipo empresarial sea rápido y sencillo además asegura el aspecto y funcionamiento de las aplicaciones a través de las plataformas *Solaris*, *OpenSolaris*, *Linux*, *Microsoft Windows* y *Apple Macintosh OS X*. La interfaz gráfica de nuestra herramienta se diseño con este *IDE* por las ventajas que proporciona para el desarrollo de aplicaciones, algunas de ellas son [25]:

- es un producto de código abierto, con todos los beneficios del software disponible en forma gratuita, el cual ha sido examinado por una comunidad de desarrolladores;

- es una herramienta de desarrollo Java, escrita totalmente en base a la tecnología Java, de modo que puede ejecutarse en cualquier ambiente que soporte Java;
- simplifica la creación de interfaces gráficas de usuario para grandes aplicaciones de Internet y cliente, además de permitir a los desarrolladores manejar diferentes guías de estilo en diversas plataformas;
- cuenta con el mejor soporte a estándares industriales de la tecnología Java, el proyecto NetBeans ha hecho que el desarrollo de aplicaciones Java de tipo empresarial sea rápido y sencillo;
- administración de interfaces de usuario (e.g. menús y barras de herramientas);
- gestión de configuraciones del usuario;
- administración del almacenamiento (e.g. guardando y cargando cualquier tipo de dato);
- manejo de ventanas;
- marcos de trabajo basado en asistentes (e.g. diálogos paso a paso).

CAPÍTULO 4

Estándares para datos geográficos

En este capítulo se explica el origen, propósito y algunos componentes de la especificación *OpenGIS*. Comenzamos describiendo que son los geodatos, algunas clasificaciones de éstos y la complejidad en el manejo y control de información geográfica, así como el tipo de aplicaciones que maneja esta clase de información. Después, se aborda el origen y propósito de *OGIS*, las partes que lo integran, su funcionamiento, su ámbito y los beneficios que ofrece a los desarrolladores, gestores de la información y usuarios. Por otro lado, se describe el modelo de datos geográficos abierto como parte del entorno de trabajo de *OGIS* y finalmente se explica el estándar utilizado para el modelo de datos geográficos realizado en este trabajo de tesis.

4.1. Seguimiento de estándares

Aunque los *SIG* se han caracterizado por su heterogeneidad, cada vez se vienen consolidando más los estándares geográficos gracias a los esfuerzos de comites conformados por la indutria, la academia y el sector público. Una metodología, como la que se busca proponer, debe tener en cuenta dichos estándares desde el inicio de labores de la ingeniería de requerimientos, para así garantizar que serán tomados en cuenta en el resto del ciclo de vida del sistema. La siguiente clasificación pone de manifiesto la importancia de los estándares en la construcción de aplicaciones *SIG*, dada su creciente naturaleza distribuida [5]:

- **estándares para productos de datos**. Estos se preocupan por definir la estructura de las capas espaciales de modo que se provea un marco de trabajo común sobre el cual almacenar la información recolectada con el propósito de construir aplicaciones *SIG*;

- **estándares para transferencia de datos**. Como su nombre lo indica, estos estándares permiten facilitar el intercambio de datos entre aplicaciones prin-

cipalmente propietarias sin embargo, dado el auge de Internet y de las arquitecturas de sistemas por capa, estos estándares proponen marcos de referencia comunes para todo un conjunto de aplicaciones y no solamente traductores entre parejas de formatos;

- **estándares para calidad de datos**. Son documentos que listan los requerimientos de calidad de datos para aplicaciones específicas o para escalas específicas, por lo cual dependen del propósito de la aplicación y de las capas temáticas que se esten modelando;

- **estándares para metadatos**. Presentan una descripción detallada de distintas características asociadas a los datos como quién, cómo y cuándo se produjo la información. Son de vital importancia para el establecimiento de redes de distribución de datos en donde es necesario catalogar la información con precisión.

4.2. La problematica del geoprocesamiento

Datos geográficos, o *"geodatos"* son aquellos que describen sucesos relacionados directamente o indirectamente con una ubicación (tiempo y orientación) relativa a la supercie de la Tierra y han sido recopilados en forma digital por más de 37 años. El índice global de recopilación aumento rápidamente con los avances de las tecnologías como satélites de alta resolución de imágenes, sistemas de posicionamiento global y el creciente aumento de personas y organizaciones que estan seleccionando y usando datos geográficos. Ese número seguirá creciendo con la conciencia cada vez mayor entre los tecnólogos de la información que consideran a la organización de datos por sitios como una forma fundamental para la estructuración y uso de datos digitales. Los geodatos se refieren a la amplia gama de datos geográficos digitales, algunos son [29]:

- **mapas digitales**. Contienen información representada por medio de regiones cuyos contenidos son: el tipo de suelo de una determinada área, las fronteras políticas, los distritos de votación, el número de habitantes promedio en una zona específica;

- **datos de la imagen "raster"**. Son datos normalmente adquiridos a través del escaneo o cámaras digitales. Algunos tipos de imágenes son: multiespectrales de imágenes *LANDSAT* y *SPOT* [1], ortoimágenes digitales, polígonos de datos de uso de tierra y niveles promedio de nitrógeno en el suelo en una cuadrícula de puntos de muestreo aleatorio;

- **punto de vectores de datos**. Este tipo de información representa puntos de control, pozos de agua, antenas de radio, árboles y puntos que probablemente sean golpeados por un rayo;

[1]Son imágenes tomadas vía satélite [8].

- **datos vectoriales**. Es la información relacionada con carreteras en los sistemas de análisis de tráfico, medidas y colindancias en medición de propiedades, gasoductos y rutas de transmisión de microondas de satélites a teléfonos móvil;
- **representaciones en tres dimensiones de sucesos geográficos**. Incluyen proyecciones estadísticas de los contaminantes en el suelo, modelos de huracanes y modelos de características de la sub-superficie calculados a partir de radar de pruebas;
- **datos espacio-temporal**. Son representados por medio series de tiempo animadas que muestran las tierras agrícolas sacrificadas por la expansión humana y modelos de dispersión de las aguas residuales de una tubería de descarga de agua en las distintas profundidades a través del tiempo.

Los formatos de geodatos suelen ser más complejos que otros tipos de formatos de datos digitales. Esto se debe a una amplia gama de información que debe ser capaz de representar y a los puntos que mencionaremos a continuación [29]:

- existen diferentes tipos de software para crear, almacenar, recuperar, procesar y visualizar datos geográficos, estableciendo cada uno de ellos un formato de datos diferente para el manejo de la información;
- las organizaciones y empresas que usan un determinado software adoptan las normas y conveciones impuestas por éste para el manejo datos geográficos, originando la existencia de diversas formas para medir y representar geodatos;
- la falta de comunicación y colaboración entre las empresas dedicadas al desarrollo de herramientas que son aplicadas para el manejo de geodatos. No ha sido posible llegar a un acuerdo sobre como los datos geográficos deben ser estructurados y compartidos, también no se ha establecido la manera en que sus sistemas se podrían comunicar mejor entre sí.

El proceso de creación y el manejo de datos geográficos realizado por los sistemas se denomina geoprocesamiento, el cual consiste en el cómputo digital de datos geográficos, los cuales incluyen la siguente información:

- sistemas de información geográfica;
- sistemas de información de tierras (*LIS, Land Information Systems*);
- imágenes de la Tierra y procesamiento de imágenes;
- almacenamiento de geodatos en bases de datos;
- navegación, meteorología y sismología;
- simuladores de vuelo.

La integración de datos geográficos procedentes de diversas fuentes es cada vez más importante por el aumento de las preocupaciones medioambientales, por las presiones sobre gobiernos y empresas para llevar a cabo de manera más eficiente la administración de recursos naturales y por el simple hecho de la existencia de un organismo en rápido crecimiento en el uso de datos geográficos y herramientas de geoprocesamiento.

4.3. OGIS

Para proveer soluciones al problema de la interoperabilidad en el geoprocesamiento se creo en 1994 el *Open GIS Consortium, Inc.* (*OGC*), una organización sin fines de lucro dedicada a la promoción de nuevos enfoques técnicos y comerciales para geoprocesamiento interoperable. Los miembros del *OGC* comparten una visión positiva de la infraestructura nacional y global de la información en donde los datos geográficos y recursos de geoprocesamiento circule libremente, plenamente integrado con las últimas tecnologías de computación distribuida. Algunas organizaciones como: proveedores de software de geoprocesamiento, de bases de datos, de visualización; las compañias, las universidades y las agencias federales se ha unido al consorcio *OGC* para participar en la creación de una especificación de software y para establecer estrategias que ayuden a resolver el problema del geoprocesamiento.

La especificación del software *OGC* es la descripción de los servicios y componentes de la interoperabilidad de datos geográficos abiertos (*OGIS, Open Geodata Interoperability Specification*), la cual es una extensa especificación de un marco de software para distribuir el acceso a los datos geográficos y recursos de geoprocesamiento. *OGIS* proporcionará a los desarrolladores de software de todo el mundo una plantilla detallada de interfaz común para escribir software que puedan interoperar con software compatible *OGIS* escrito por otros desarrolladores. El marco de trabajo de *OGIS* incluye tres partes [29]:

- **modelo de datos geográficos abierto (OGM, Open Geodata Model)**. Es un medio común para la representación digital de los elementos geográficos de la Tierra y sucesos geográficos relacionados con ésta, matemáticamente y conceptualmente;

- **modelo de servicios OGIS**. Consiste en una especificación común para la aplicación de servicios de acceso a geodatos, gestión, manipulación, representación e intercambio entre comunidades de la información;

- **modelo de información entre las comunidades**. Es un marco de trabajo para el uso de modelos de datos geográficos abiertos y modelos de servicios *OGIS* para resolver no solo el problema técnico de no-interoperabilidad, sino también el problema de la no-interoperabilidad institucional.

4.3.1. Funcionamiento de OGIS

El procedimiento de *OGIS* para lograr la integración de los diferentes recursos y aplicaciones que usan datos geográficos, consiste en el desarrollo de sistemas con interfaces *OGIS* que crearán componentes *middleware* y aplicaciones que puedan manejar una gama completa de datos geográficos y funciones de geoprocesamiento. Los usuarios de estos sistemas podrán compartir una red datos potencialmente enorme en la que todos los datos geográficos se ajusten a un modelo de datos genéricos, aunque los datos pueden haber sido producidos en diferentes momentos por grupos no relacionados usando distintos sistemas de producción para diferentes fines.

La especificación de interoperabilidad de datos geográficos abiertos (*OGIS*) proporciona un marco de trabajo a los desarrolladores de software para crear sistemas que permitan a sus usuarios acceder y procesar los datos geográficos de una variedad de fuentes a través de una interfaz de cómputo común en un ambiente de distribución gratuita de la información [29].

La frase anterior es analizada en los siguientes puntos:

- *un marco de trabajo para los desarrolladores de software* significa que *OGIS* es una especificación de software detallada basado en un extenso y común plan de geoprocesamiento interoperable (formado por el consenso de la industria para uso general);

- *acceder y procesar* significa en este contexto que los usuarios de datos geográficos pueden realizar consultas remotas a bases de datos, procesamiento de recursos información de manera remota y aprovechar las tecnologías de computación distribuida (e.g. middleware, servicios Web);

- *una variedad de fuentes* implica que los usuarios tendrán acceso a los datos obtenidos en una variedad de formas y se almacenan en distintas maneras y en diversas de bases de datos relacionales y no relacionales;

- *una interfaz de cómputo común* significa que las interfaces *OGIS* proporcionan una comunicación fiable entre diferentes plataformas de software que están equipados para usar estas interfaces, i.e. la comunicación para el envio y recepción de datos se realiza a través de plataformas de software que tengan en común definidas interfaces *OGIS*;

- *un ambiente de distribución gratuita de la información* significa que los paradigmas de los sistemas de cómputo están cambiando de sistemas cerrados a sistemas de distribución gratuita, de sistemas aislados a sistemas que puedan interactúar en tiempo real, de aplicaciones hechas a la medida a aplicaciones provistas de componentes de software que interactúan para ofrecer capacidades más flexibles para el usuario. La evolución de éstos paradigmas le brindan a

OGIS la oportunidad de trabajar con aplicaciones implementadas en diferentes plataformas operativas.

4.3.2. Beneficios

El internet y otras redes de computadoras están proporcionando acceso a la creciente fuente de datos y servicios a los millones de usuarios. Las ventajas de utilizar esta tecnología son evidentes para la mayoría de los usuarios de datos geográficos y recursos de geoprocesamiento. Las grandes organizaciones necesitan integrar datos geográficos y de geoprocesamiento a través de recursos de redes de área amplia. Dentro de los entornos de escritorio, datos geográficos diferentes y recursos de geoprocesamiento deben integrarse para permitir un trabajo útil. *OGIS* facilita la integración en el entorno de red y el entorno de escritorio.

Los desarrolladores de aplicaciones, gestores de la información, y los usuarios finales los cuales forman parte de la revolución global de la informática, se benefician del software *OGIS*. El desarrollador de la aplicación puede más fácilmente: [29]:

- escribir programas para acceder a datos geográficos y acceder a recursos de geoprocesamiento;
- diseñar aplicaciones a la medida para satisfacer necesidades concretas de los usuarios;
- elegir un entorno de desarrollo.;
- entregar las aplicaciones en una amplia variedad de plataformas;
- reutilizar código de geoprocesamiento.

El gestor de la información cuenta con una mayor flexibilidad para:

- acceder y/o distribuir datos geográficos;
- proporcionar a los clientes capacidades de geoprocesamiento;
- integrar datos geográficos y procesamiento en una arquitectura de computación corporativa;
- elegir las plataformas adecuadas de acuerdo al tipo de computadora personal, tipo de servidor y tipo de plataforma de computación distribuida (e.g. *CORBA*, *OLE/COM*, *DCE*);
- ajustar las necesidades del usuario con la herramienta de geoprocesamiento correcta.

Los usuarios finales son beneficiados, recibiendo:

- acceso en tiempo real a un universo de información geográfica;
- más aplicaciones (*middleware* y documentos compuestos) que aprovechen la información geográfica;
- la capacidad de trabajar con diferentes tipos y formatos de datos geográficos dentro de un único entorno de aplicación y flujos de trabajo continuos, sin preocuparse de los detalles de estos tipos de formatos.

4.3.3. Ámbito

OGIS aborda tres aspectos básicos del problema de acceso y utilización de los datos geográficos de una variedad de fuentes [29]:

- **obtener una conexión**. *OGIS* no aborda el dominio de la plataforma de computación distribuida (*DCP, Distributed Computing Platform*) que permite a las aplicaciones interactuar entre sí a pesar de que se encuentren en diferentes computadoras. El tema de la conectividad sigue siendo abordada por otros proveedores de tecnología y *OGIS* continuará el seguimiento de este progreso. *OGIS* no se limita a una determinada *DCP*;
- **obtener un servicio**. Es el dominio de OGIS es permitir a las aplicaciones interactuar con otras aplicaciones que administran, entregan y procesan datos geográficos. Se refiere a la forma de solicitar un servicio, cómo determinar si una petición es una solicitud de datos o una solicitud para realizar alguna operación sobre los datos o ambas. Define un conjunto estándar de tipos de datos y operaciones sobre esos tipos de datos, proporcionando así un marco de trabajo común para la interoperabilidad entre proveedores y clientes de datos geográficos. También proporciona los servicios que facilita el intercambio de datos entre diferentes comunidades de la información (i.e. grupos de usuarios con diferentes significados, semántica y sintaxis para el procesamiento de datos geográficos y espaciales);
- **comprensión de los resultados**. Este es el dominio de los individuos o grupos que tienen un interés común en el significado de los datos. Ellos proveen el marco de trabajo para la intepretación de los datos, su significado, su precisión y su nivel de certificación.

4.4. Modelo de datos geográficos abierto

El modelo de datos geográficos abierto es un modelo de información extensible constituido de tipos de datos geográficos que codifican abstracciones de sucesos del mundo real en determinado espacio/tiempo. Los datos geográficos son el tipo de

información fundamental que se puede utilizar para modelar una aplicación específica. *OGIS* es descrito técnicamente en términos de la tecnología orientada a objetos y la geometría, por lo tanto se consideran tres niveles de diseño orientado a objetos [29]:

- el *modelo esencial* es una descripción de una situación del mundo real. La construcción de un modelo esencial ayuda a establecer los hechos que deben ser modelados. Los componentes de éste modelo son tipos de objetos y tipos de eventos;

- la *especificación del modelo* describe software a un alto nivel de abstracción. *OGIS* es una especificación de modelo;

- la *implementación del modelo* se refiere a establecer planes de control de flujo dentro de los sistemas. Este modelo tiene en cuenta las limitaciones impuestas por las computadoras, *DCPs* y lenguajes de programación.

4.4.1. Representación de los elementos geográficos

El modelo de datos geográfico abierto debe coincidir con el punto de vista de las personas especializadas en el estudio de la información geográfica acerca de las entidades y las características del mundo real que pueden ser modeladas y manipuladas a través de las aplicaciones que manejan datos geográficos. Esto significa que el tipo de información que se puede expresar en el *OGM* debe ser coherente con el desarrollador de software para datos geográficos comprendiendo las formas, los sistemas y procesos de la Tierra. En la práctica sin embargo, todas las representaciones de la Tierra son abstracciones o generalizaciones de la complejidad con algún aspecto de la simbología o la presentación indicando el grado o la naturaleza de la abstracción. Desarrolladores de software utilizarán los tipos de objetos definidos por el *OGM* para el modelado de las características del mundo real, una manera más eficiente y completa que modelando mapas y presentado gráficas, que indican el grado o la naturaleza de la abstracción, esta manera ha sido la limitación de algunas implementaciones de *SIG* tradicionales.

Los elementos geográficos a ser modelados pueden clasificarse dentro de dos grupos: entidades y fenómenos [29].

- *Entidades* son reconocibles i.e. objetos discretos que tienen bien definidos los límites o extensión espacial (e.g. camiones, edificios, lagos y ciertas formas terrestres).

- *Fenómenos* que varían más o menos continuamente en el espacio y no tienen ninguna medida específica (e.g. temperatura, composición del suelo y topografía).

4.4.2. Ubicación: lugar y tiempo

Además de entidades y fenómenos, otro aspecto central del modelado de datos geográficos es el posicionamiento de los datos geográficos. Las entidades y los fenómenos no son significativos en el geoprocesamiento a menos que se expresen en términos de un modelo que los posicione por encima o por debajo de la superficie de la Tierra y quizás también en el tiempo. En el modelo esencial del *OGM*, lugar y tiempo no corresponden a las entidades de software, sino a la realidad física de las entidades y fenómenos que se modelan. El lugar es una parte que se puede medir del mundo real. El tiempo es un punto, intervalo o colección de puntos e intervalos que percibimos como el tiempo continuo. La ubicación es la idea del modelo esencial de espacio y tiempo; está representada por la geometría en la especificación del modelo.

En la figura 4.1 se presenta un diagrama de objeto del modelo esencial que describe la relación entre un lugar (suponemos que se adjunta a la entidad o a algún fenómeno que se modela) y la geometría utilizada para representar aspectos importantes de la localización de la entidad o fenómeno. Cada rectangulo representa un tipo de objeto y las líneas entre los rectangulos representan asociaciones entre los tipos de objetos. El pequeño círculo negro indica un conjunto, lo que significa que una ubicación tiene más de una coordenada geométrica. El símbolo del medio ovalo indica que el sistema de referencia espacial-temporal es una propiedad de la asociación de la relación existente entre los objetos. El símbolo del pequeño diamante se utiliza para describir la agregación (dependencia de existencia); en este caso describe la relación entre una ubicación y una coordenada geométrica.

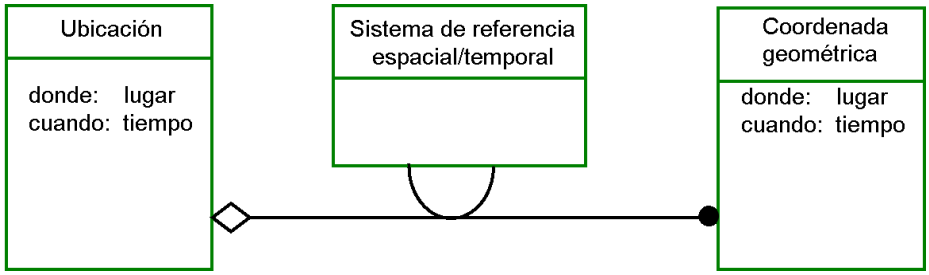


Figura 4.1: Ubicación y representación geométrica del modelo esencial

4.4.3. Propiedades y coberturas

El *OGM* proporciona un medio común para resumir los fenómenos de la Tierra, tanto matemáticamente como conceptualmente. Ambos puntos de vista están en el dominio del modelo esencial.

- Un *OGM* consiste en la representación matemática de los modelos geométricos en el espacio de uno, dos o tres dimensiones más una dimensión temporal.

También define el sistema de referencia espacial y temporal en el cual estos modelos están integrados.

- Una vista conceptual de *OGM* se refiere a cómo los seres humanos perciben el mundo que les rodea en un contexto geográfico. La vista conceptual esta integrada por: un componente visual y un componente intuitivo/interpretativo. El componente visual es una abstracción o simbolización de la realidad (incluyendo gráficos, textos y elementos de imágenes), ésta es única para cada persona o grupo de personas. El componente intuitivo/interpretativo proprociona significado y comprensión a la vista conceptual.

Los dos tipos de información geográfica fundamentales son los *elementos* y las *coberturas*. Ambos pueden ser utilizados para representar entidades del mundo real en un mapa y su correspondiente vista conceptual a una representacion del modelo de datos geográficos abiertos (*OGM*) [29].

- Un *elemento* es la representación de una entidad del mundo real o una abstracción de éste. Tiene un dominio espacial, un domino temporal o un domino espacial/temporal como una entidad (e.g. edificios, ciudades, bosques, pozos de petróleo). Los elementos suelen ser gestionados en grupos como *colecciones de elementos*. Una capa temática de un *SIG* que sólo muestra carreteras es una colección de elementos. Los elementos representan entidades geográficas.

- Una *cobertura* es una asociación de puntos dentro de un dominio espacial/temporal a un valor de un tipo de dato definido. Es décir en una cobertura cada punto tiene un particular valor simple o complejo. Una cobertura es una función de un domino espacial/temporal a un dominio de atributo.

Normalmente hay tres componentes relacionados con los *elementos* (como se muestra en la figura 4.2):

Figura 4.2: Característica del modelo esencial OGIS

1. una descripción de la *geometría* del fenómeno con relación al sistema de referencia espacial/temporal, incluida una declaración de la resolución y precisión del modelo geométrico;

2. una descripción de las *propiedades semánticas* del fenómeno es i.e. la forma en que se define el léxico de un determinado grupo de domino (un grupo de domino es un grupo de usuarios que utilizan varias definiciones de las propiedades);

3. otro componente que puede ser necesario para posicionar el fenómeno en el contexto del entorno de la aplicación o comunidad de usuarios son los *metadatos*.

4.4.4. Semántica y metadatos

El *OGM* define un modelo para describir las propiedades semánticas de un elemento geográfico. Los elementos tienen un conjunto de atributos asociadas a ellos. Éstos pueden ser arbitrariamente complejas mientras sean expresados como tipos bajo el esquema de *OGM*. Como los sistemas de referencias, es necesario facilitar descripciones del modelo de elementos para las propiedades o familia de propiedades. Por lo tanto el componente semántico de una propiedad incluye su esquema (una descripción de las propiedades de los elementos en términos de tipos de datos primitivos y restricciones sobre éstos). El *OGM* debe tener en cuenta las propiedades de los metadatos que se asocian con los elementos. La necesidad de algún tipo de metadato varían ampliamente de una aplicación a otra.

Los metadatos son simplemente un subconjunto de las propiedades de los elementos (o más típicamente de una colección de elementos). Al igual que con las propiedades de los elementos, es necesario proporcionar descripciones del modelo de elementos de metadatos elegido para las propiedades o familia de propiedades. De esta manera, el componente de los metadatos para las propiedades incluirán el esquema de metadatos, el cual es una descripción del modelo de elementos específicos para los metadatos en términos de tipos de datos primitivos [29].

4.4.5. Geometrías en elementos OGIS

Las aplicaciones geográficas deben tratar con una amplia gama de geometrías. El *OGM* debe proporcionar la capacidad de hacer frente a las geometrías, independientemente de la representación geométrica elegida por la aplicación. Esto significa que el modelo de geometría debe ser general e incluir todos los tipos de geometría, proporcionando una representación común con todas la aplicaciones.

Debido a la naturaleza de los datos geográficos, las entidades deben ser descritas en aplicaciones geográficas. OGIS necesita expresar topologías de una, dos y tres dimensiones. Los tipos de datos geográficos definidos en la especificación son los siguientes [29]:

- **punto**. Es una topología cero-dimensional, este tipo especifica una ubicación geométrica;

- **curva**. Abarca una topolgía uni-dimensional; esta clase especifica una familia de entidades geométricas incluyendo segmentos de línea, líneas, arcos;
- **superficie**. Comprende una topología bidimensional; este tipo especifica un grupo de entidades geométricas que incluyen medidas de áreas (e.g. la cantidad de puntos en una curva cerrada o en un conjunto de curvas cerradas) y superficies;
- **sólido**. Es una topología tridimensional; este tipo especifica un grupo de entidades geométricas que incluye volúmenes (e.g. el número de puntos dentro de una superfice cerrada o en un conjunto cerrado de superficies) y sólidos que son definidos en otra manera.

4.5. ISO/TC 211

A partir de 1990, se ha experimentado una gran evolución en tecnología de geoprocesamiento, observándose también un aumento significativo en el número de *SIG* instalados. Este tipo de sistemas se ha adoptado en ambientes de administración pública y privada. El creciente desarrollo de los *SIG* se debe por una parte a la reducción del precio en el diseño y desarrollo de los sistemas de captura de datos y de software de geoprocesamiento. Por otra parte, tal divulgación se debe a la amplia difusión de la cultura del geoprocesamiento y a la consiguiente expansión de las aplicaciones que están siendo implementadas en *SIG*.

Cada vez más, aumenta el número de usuarios de *SIG* y por lo tanto, también se incrementa la búsqueda de los datos digitales geo-espaciales para ser utilizados en el análisis espacial de datos.

Existe una tendencia a incrementar el intercambio de datos geoespaciales, principalmente con la ayuda de sistemas ejecutándose sobre Internet (como el *FGDC*, *Federal Geographic Data Committee*)[2] y con la colaboración de los centros de distribución de datos *clearinghouses* (son localidades a las cuales se accede a través de un sitio Web para buscar conjuntos de datos espaciales). Sin embargo, poco se ha hecho para facilitar el intercambio de soluciones de modelado de base de datos geográficas. Además, el uso poco sistemático de los datos importados de diversas fuentes, puede dar lugar a problemas de fiabilidad en el sistema y llevaría a la incoherencia y redundancia de datos en las bases de datos geográficas locales.

Actualmente está en aumento el uso de *repositorios geoespaciales*[3] para almacenar el contenido conceptual de esquemas de bases de datos y diccionarios alineados con

[2]Es un comité creado por el gobierno de Estados Unidos de América, que describe datos geoespaciales.

[3]Es una colección de datos organizada para proporcionar información, acerca de la semántica, la geometría, la temporalidad y la integridad de datos almacenados en una base de datos geográfica.

las normas internacionales de información geográfica (*ISO/TC 211* y *OGC*).

ISO/TC 211 es el comité técnico de la *ISO*[4] encargado de definir las normas internacionales en el ámbito de la información geográfica digital. Las labores del comité incluyen el desarrollo de métodos, normas y servicios necesarios para adquirir, procesar, gestionar y acceder a datos geospaciales. Las normas internacionales proporcionan las bases para el desarrollo de aplicaciones geoespaciales. La figura 4.3 muestra las lista de estándares que *ISO/TC* desarrolló y sobre las que trabaja actualmente. A continuación, explicaremos en que consisten algunos estándares [17].

Estándar	**Descripción**
ISO 19101	Modelo de referencia
ISO 19102	Visión general
ISO 19103	Esquema conceptual
ISO 19104	Terminología
ISO 19105	Conformidad y pruebas
ISO 19106	Perfiles
ISO 19107	Esquema espacial
ISO 19108	Esquema temporal
ISO 19109	Normas para la aplicación de esquemas
ISO 19110	Metodología para la clasificación de elementos
ISO 19111	Referencia espacial por medio de coordenadas
ISO 19112	Referencia espacial por medio de identificadores geográficos
ISO 19113	Principios de calidad
ISO 19114	Procedimientos de evaluación de la calidad
ISO 19115	Metadatos
ISO 19116	Servicios de posicionamiento
ISO 19117	Representación
ISO 19118	Codificación
ISO 19119	Servicios
ISO 19120	Normas funcionales
ISO 19121	Imágenes y datos en cuadrícula
ISO 19122	Cualificación del personal
ISO 19123	Esquema para cobertura geométrica y funciones
ISO 19124	Imágenes y componentes de datos en cuadrícula
ISO 19125	Funciones de acceso - opción SQL

Figura 4.3: Estándares ISO/TC 211

[4]Se traduce como Organización Internacional para la Estandarización (International Organization for Standardization).

- **ISO 19103 (Esquema conceptual)**. Se refiere a la selección de un lenguaje de esquema conceptual que satisfaga las necesidades de los modelos de información geográfica y del desarrollo de estándares.

- **ISO 19107 (Esquema espacial)**. Define un conjunto de datos espaciales estándar y las operaciones de tipo geométrico y espacios topologicos. La geometría suministra los medios para describir las formas de los objetos con coordenadas y funciones matemáticas.

- **ISO 19108 (Esquema temporal)**. Es la contraparte del esquema espacial. En este se definen las características temporales y las funciones necesarias para describir los eventos que ocurren en el tiempo dentro del contexto espacial.

- **ISO 19109 (Normas para la aplicación de esquemas)**. Define el GFM (*General Feature Model*), el cual consiste en el metamodelo para la abstracción de características del mundo real. Las normas de aplicación proporcionan los principios sobre el proceso de abstracción y la realización de aplicación de esquemas que documenta una percepción de la realidad.

- **ISO 19110 (Metodología para la clasificación de elementos)**. Determina un metamodelo para la documentación de entidades del mundo real.

- **ISO 19111 (Referencia espacial por medio de coordenas)**. Define la metodología para documentar el sistema de referencia de coordenas.

- **ISO 19115 (Metadatos)**. Especifica el contenido y la estructura de los componentes de los metadatos para describir conjunto de datos.

4.6. Esquema conceptual GeoFrame

En el diseño de un *SIG* colabora personal de diversas disciplinas (e.g. cartógrafos, biólogos y arquitectos), por lo tanto tienen muy poco conocimiento de los métodos de desarrollo de software. El trabajo de estos diseñadores puede ser facilitado a través de la existencia de un modelo conceptual común, para el diseño de aplicaciones *GIS* bajo un mismo patrón.

El desarrollo de nuestra solución en la parte del diseño conceptual de la base de datos geográfica se basó en *GeoFrame*, el cual es un esquema conceptual para modelar bases de datos geográficas y que consiste de un marco conceptual que proporciona una diagrama de clases con el objetivo de ayudar al diseñador en el modelado conceptual de sucesos geográficos [19].

Las características que se contemplaron para el uso del esquema conceptual *GeoFrame* son:

- es un marco de trabajo conceptual orientado a objetos, el cual hace uso del paradigma POO (*Programación Orientada a Objetos*) para el diseño de un sistema *SIG*;

- el esquema conceptual bajo el cual se encuentra estructurado es útil para describir el modelo espacial de datos de una base de datos geográfica englobando los niveles de modelo de objetos y de campo;

- proporciona un conjunto de símbolos (iconos) para la especificación de esquemas conceptuales de bases de datos geográficos.

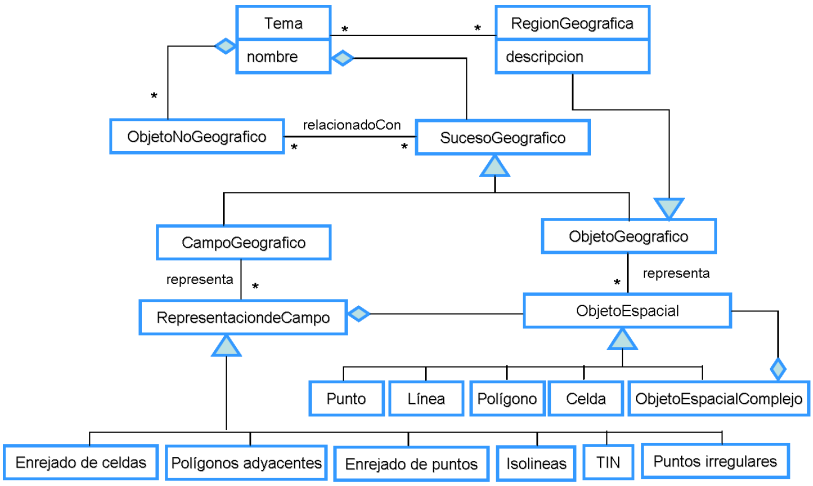


Figura 4.4: Diagrama de clases del esquema conceptual GeoFrame

En la figura 4.4 presentamos el diagrama de clases que describe los objetos geográficos y las relaciones entre ellos. El objetivo del diagrama es la construcción de modelos de bases de datos geográficas. Se definen cuatro clases principales: *RegionGeográfica, Tema, ObjetoNoGeográfico, FenomenoGeográfico* que generalizan, en un alto nivel de abstracción, los elementos de un esquema de datos geográficos.

Las clases *Tema* y *RegionGeográfica* son la base de cualquier aplicación geográfica, ya que tienen como principal objetivo la administración y manipulación de un conjunto de datos para cierta región de interés. Podemos imaginar que en una aplicación *SIG* de tipo urbana, el área donde está establecida una ciudad puede ser especificada como una región geográfica de interés. Para esta región geográfica, se podrían definir los temas: *límites de la zona urbana, vecindarios, edificios.*

En bases de datos geográficas, existen algunos objetos que no contienen información georeferenciada, i.e. que describen atributos no gráficos. En *GeoFrame*, estos objetos son considerados instancias de la subclase *ObjetoNoGeográfico*. La clase abstracta *SucesoGeográfico* abarca objetos que pueden ser descritos con información

georeferenciada (e.g. una parcela puede ser considerada como una instancia de la clase *FenomenoGeográfico*). La clase *SucesoGeográfico* y *ObjetoNoGeográfico* pueden ser relacionadas una con otra por medio de la asociación *relacionadoCon* (e.g cada parcela pertenece a algún propietario).

Los *SucesosGeográficos* pueden ser modelados considerando dos formas diferentes y están representados por medio de las clases *CampoGeográfico* y *ObjetoGeográfico*. La clase *ObjetoGeográfico* representa elementos geográficos que pueden ser descritos a través de un conjunto de atributos (e.g. parcelas, ríos, carreteras y ciudades). La clase abstracta *CampoGeográfico* generaliza sucesos que son modeladas en base a funciones de una variable. Algunos elementos utilizan variables que son distribuidas sobre la superficie de manera continua (e.g. relieve, temperatura y atributos de suelo), mientras que otras variables son distribuidas de una manera discreta (e.g. población y ocurrencia de epidemias).

Algunas entidades geográficas pueden representar dimensiones espaciales complejas, i.e. están constituidas por otros objetos espaciales. La clase *ObjetoEspacial* generaliza clases que son usadas para la especificación de la representación de componentes espaciales y éstas son: *punto, línea, polígono, celda y objeto espacial complejo*. Los modelos *cuadrículas de celdas, polígonos adyacentes, isolineas, cuadrícula de puntos, TIN, polígonos irregulares* son subclases que provienen de la clase *Representaciónde-Campo*. Para representar estos elementos en un modelo conceptual de base de datos geográfica, se manejan los iconos mostrados en la figura 4.5, los cuales simbolizan cada tipo de objeto geográfico de acuerdo a la clasificación representada en *GeoFrame*: si pertenece a la clase *CampoGeográfico* u *ObjetoGeográfico* y forman parte de nuestra solución para la representación gráfica del modelo conceptual de la base de datos geográfica.

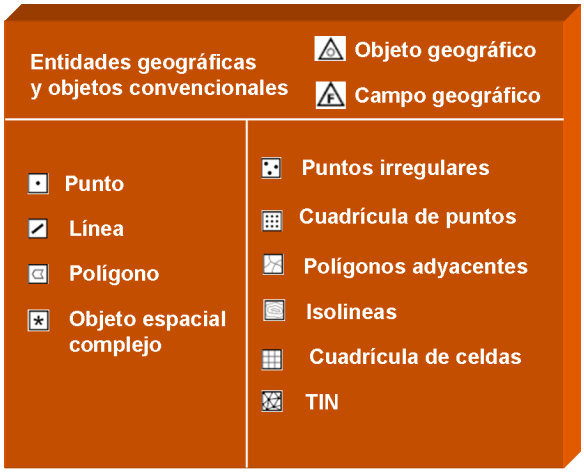


Figura 4.5: Iconos para la representación de entidades geográficas

CAPÍTULO 5

Diseño de la solución

En este capítulo presentamos las fases que integran la solución propuesta. Iniciamos describiendo el problema y enseguida explicamos los componentes que integran la arquitectura de la solución así como su funcionamiento y las relaciones existentes entre ellos. A continuación, definimos los componentes del entorno que forman la interfaz gráfica del sistema y la función que desempeñan. Planteamos el caso de estudio sobre el cual se trabajó en el desarrollo de la implementación y finalmente se presenta una serie de diagramas que definen el diseño del sistema para su posterior implementación.

5.1. Descripción del problema

Las características tratadas hasta ahora confieren a los *SIG* una dificultad extra al efectuar el proceso inicial de ingeniería de requerimientos sobre sus aplicaciones. La mayoría de los estudios muestra que los *SIG* son difíciles de usar o que su implementación aún esta lejos de la cultura del usuario normal, lo cual evidentemente complica un proceso que de hecho no es fácil. A esto debe sumarse que las metodologías propias de la ingeniería de requerimientos no fueron concebidas para este tipo de sistemas. En general, estas problemáticas pueden clasificarse siguiendo la división entre el proceso y el producto de la ingeniería de software así:

- **metodología incompletas**. La mayoría de los trabajos de modelado y de la especificación de requerimientos han sido concebidas para sistemas alfanuméricos en donde los atributos espaciales no forman parte del núcleo de diagramas y operaciones y, aunque existen extensiones que permiten incluir información georeferenciada, éstas aún no están lo suficientemente difundidas y en muchos casos son complementos que no se integran de manera fácil a las metodologías;

- **stakeholders**[1] **heterogéneos**. Durante mucho tiempo los *SIG* formaron parte de grupos donde los usuarios pertenecían a la comunidad científica en muchos casos sin ninguna experiencia en computación por lo cual la recolección de requerimientos del sistema se hacia en un ambiente muy diversificado. La masificación de Internet y el uso cada mez más frecuente de los *SIG* como herramientas de apoyo en la toma de decisiones por el gobierno y particulares, agravó la situación ampliando enormemente los usos del sistema;

- **complejidad de la información**. La información espacial cuenta con propiedades que no hacen fácil la labor de modelado de datos que se desarrolla en el análisis de requerimientos. La información georeferenciada es bastante voluminosa y sus altos costos causan que la mayoría de las organizaciones deban acudir a distintas fuentes para proveerse de los datos necesarios para una aplicación. Como consecuencia, se dispone generalmente de una base de información poco homogénea, con distintos rasgos de calidad, escala y actualidad. Por este motivo, muchas veces es difícil establecer relaciones entre entidades de información que se encuentran lógicamente conectadas.

En el diseño e implementación de sistemas se ejecutan diversas actividades como la definición de requerimientos, el modelado, la implementación y la prueba del sistema, cada una de las cuales conlleva un esfuerzo y tiempo considerable. El reto en el desarrollo de software es contar con procesos automatizados que apoyen a las actividades mencionadas con el objetivo de agilizar el proceso de desarrollo de software y posicionarse en cualquier etapa del desarrollo para afrontar y realizar los cambios de manera fácil, rápida y eficiente sin afectar los componentes existentes.

Después de la captura de requerimientos del sistema, la siguiente fase del proceso es el modelo de datos que expresa la lógica y la funcionalidad del sistema por medio de representaciones gráficas o pictográficas de todos los conceptos abstractos, lo cual conduce a un modelo de fácil entendimiento y uso.

Actualmente, existen algunas herramientas de modelado datos que se enfocan a modelar sólo la base de datos generando la descripción de ésta; otras herramientas modelan las relaciones entre las entidades que integrarán el sistema generando descripciones sin un significado relevante.

En este trabajo de tesis proponemos una metodología para el modelado de datos (base de datos y sistema), a través del diseño de una herramienta para modelar datos desde tres perspectivas: *base de datos, interfaz de usuario y comportamiento*, cada aspecto tiene asociado una representación gráfica y la traducción de esa representación está proporcionada por las descripciones *XML*, las cuales proveerán de funcionalidad a la capa de Web (generador automático de código).

[1]Este termino se utiliza para referirse a cualquier persona que tenga influencia directa o indirecta sobre los requerimientos del sistema e.g. los programadores, los usuarios finales [38].

5.2. Arquitectura propuesta

La arquitectura diseñada para la solución del problema se presenta en la figura 5.1. Esta arquitectura representa una alternativa para el desarrollo de aplicaciones centrandose en la parte de modelado de datos (*base de datos, interfaz de usuario y comportamiento*) para la generación de modelos de datos y archivos de descripciones *XML* de estos modelos.

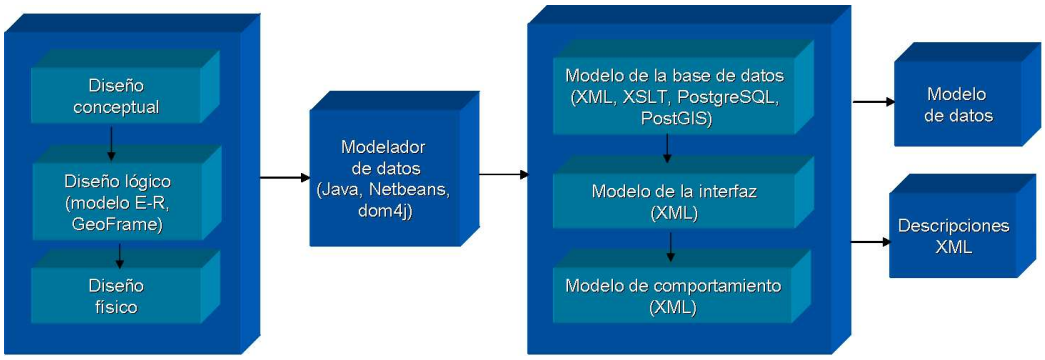


Figura 5.1: Arquitectura de la solución

Los componentes que la integran son:

- *las fases del diseño de bases de datos* (diseño conceptual, diseño lógico y diseño físico) forman la parte inicial de nuestra arquitectura porque el objetivo es el diseño de una herramienta para modelar datos. Por lo tanto, es indespensable realizar un análisis de la información que se desea modelar considerando las tres etapas. En la fase de diseño conceptual, se describe el contenido de la información de la base de datos; se tiene que el caso de estudio es un *SIG* enfocado a la *capa de hidrología*, por lo que el tipo de información analizada son datos espaciales y nomimales. En la parte del diseño lógico se utilizó la técnica de modelado entidad-relación para definir las entidades y las relaciones entre éstas. El tipo de objeto geográfico asociado a cada entidad se representa por medio de un icono obtenido del esquema conceptual GeoFrame. Por último, el diseño físico es la estructura de la base de datos que es implementada en el manejador de base de datos *PostgreSQL*;

- el *modelador de datos* es el núcleo de la arquitectura y representa la herramienta que proporciona los elementos necesarios (entidades y relaciones) para modelar la base de datos obtenida de las fases de diseño. El *IDE* utilizado para el desarrollo de la interfaz gráfica de usuario es *Netbeans* y el marco de trabajo empleado para la creación y manipulación de documentos *XML* es *dom4j*;

- *los modelos de datos* (base de datos, interfaz de usuario y comportamiento) son generados por el modelador de datos; cada uno de estos modelos tiene respectivamente una representación gráfica. El *modelo de la base de datos* consiste de las

entidades y las relaciones entre éstas, las cuales se definen en un archivo de descripción *XML*, y en un archivo con extensión *.sql*. Éste describe la estructura de la base de datos en lenguaje *SQL* para ser implementada en el manejador de base de datos *PostgreSQL*. El *modelo de interfaz de usuario* describe la estructura y el contenido de una interfaz gráfica por medio de entidades predefinidas (interfaz, menú, submenús, jsp, conexion a base de datos) en archivos de descripción *XML*. Finalmente, la definición del *modelo de comportamiento* se proporciona a través de un archivo de descripción *XML* que contiene la navegación entre las páginas del sistema modelado;

- el *modelo de datos* es el resultado final que implica la representación de los tres aspectos de modelo de datos (*base de datos, interfaz de usuario y comportamiento*) de manera gráfica;
- las *descripciones XML* son archivos *XML* generados por la herramienta. Éstos contienen la estructura de la base de datos, las especificaciones de la interfaz de usuario y del comportamiento del sistema. Éstos son leídos por una aplicación que genera de manera automática código, cuyo resultado es un conjunto de subsistemas para consulta y captura de información geográfica en un ambiente Web.

5.2.1. Proceso de creación del modelo de datos

El desarrollo de un modelo de datos utilizando el diseño de la arquitectura de la solución debe cumplir ciertas fases que se muestran en la figura 5.2 y se listan enseguida:

- el *modelo de la base de datos* es el primero que se debe crear debido a que contiene las entidades y relaciones que integran la base de datos donde se almacenará la información del sistema. Además con base a las entidades geográficas definidas se crean los módulos del *SIG*. Una vez realizado el modelo gráfico se procede a generar la descripción de éste en un archivo *XML* y a aplicar transformaciones de *XSLT* para generar el documento con extensión *.sql*, el cual contiene el diseño físico de la base de datos, listo para ser ejecutado en un servidor que tenga instalados y configurados PostgreSQL y PostGIS;
- la *estructura de directorios* de la aplicación consiste en definir el nombre del directorio donde se va almacenar el sistema completo producido por el generador automático de código. También se configuran los nombres de los subdirectorios que van a contener los subsistemas, cada uno asociado con una entidad geográfica. La estructura de directorios es guardada en un archivo XML; la información definida en éste es utilizada posteriormente por los modelos de interfaz de usuario y comportamiento para la creación de sus descripciones *XML*;
- después de establecer la estructura de directorios se realiza el *modelo de interfaz de usuario* creando las entidades de interfaz de usuario en el siguiente orden:

1. entidad conexionBD;
2. entidad interfaz de usuario;
3. entidad menú;
4. entidad submenú;
5. entidad paginajsp.

La secuencia lógica de las funciones que realiza cada entidad y la dependencia entre las entidades menu y submenu determinó el orden mencionado. Las entidades conexionBD, interfaz, paginajsp son almacenadas en un archivo XML cada una, mientras que para las entidades menú, submenú se genera un archivo *XML* por cada menú y submenú definidos para el sistema.

- El *modelo de comportamiento* es el último en diseñarse debido a que utiliza información del modelo de la base de datos y del modelo de interfaz para la generación del archivo de descripción *XML*.

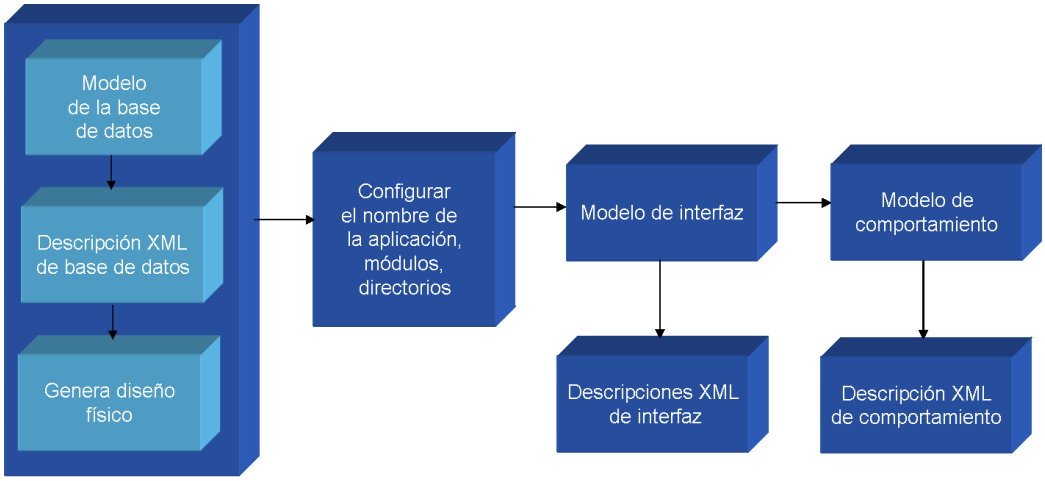


Figura 5.2: Proceso de creción del modelo de datos

Los archivos son almacenados en una estructura de directorios predefinida (ver figura 5.3), la carpeta raíz es *Description* la cual, mantiene la descripción del modelo de la base de datos, la descripción para generar las páginas jsp y contiene los subdirectorios: *menus* donde se guardan las descripciones de menús, *submenus* donde se almacenan las descripciones de submenús, *interfaz* contiene las descripciones de interfaz de usuario que integrarán el sistema y el subdirectorio *contenidos* guarda la descripción del modelo de comportamiento.

5.3. Componentes del entorno

La interfaz gráfica de usuario (ver figura 5.4) esta constituida por:

- *barra de menu*, describe las operaciones que se pueden realizar por la herramienta (e.g abrir o guardar un archivo, cerrar la aplicación) las opciones que constituyen este menú son: Archivo, Ver, Herramientas, Ayuda;
- *barra de herramientas*, contiene la representación en forma gráfica (iconos) de las operaciones de la barra de menú;
- *barra para la generación de descripciones*, comprende los botones que permiten la generación de las descripciones XML y el script de la base de datos, una vez diseñado el modelo de datos;
- *paneles para modelado*, representan las secciones que contiene un área de dibujo para cada perspectiva del modelo;
- *barra para modelo de base de datos*, contiene los iconos para representar las entidades y sus relaciones en el modelo de base de datos;
- *barra para modelo de interfaz de usuario*, comprende las figuras que simbolizan las entidades que integran el modelo de interfaz;
- *barra para modelo de comportamiento*, representa el símbolo de la entidad para el modelo de comportamiento;
- *panel de dibujo para modelado de datos*, es el área donde se elabora el diagrama del modelo de datos del sistema que se este diseñando;
- *barras de desplazamiento*, útiles para visualizar todo el contenido del diagrama cuando el recuadro establecido no es lo suficientemente grande.

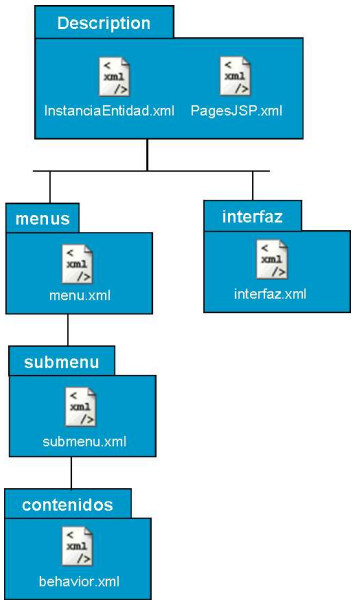


Figura 5.3: Estructura de directorios

5.4. Caso de estudio

El área de aplicación hacia el cual se enfoco el desarrollo de este trabajo de tesis fue *SIG*, por ello se conto con información geográfica proporcionada por el *INEGI*. Dicha información corresponde al estado de Colima. Se diseño un modelo de datos constituido por 9 capas geográficas, cada una de las cuales cuenta con distintas entidades que se diferencian por sus atributos, así como la capa geográfica y la capa de nivel a la que pertenecen. A continuación se listan las capas geográficas:

1. altimetría y datos de elevación;
2. hidrografía e infraestructura hidráulica;
3. localidades y rasgos urbanos;
4. límites;
5. instalaciones diversas e industriales;
6. tanques de almacenamiento, conductos y líneas de transmisión;
7. comunicación y transporte;
8. elementos de referencia topográfica;

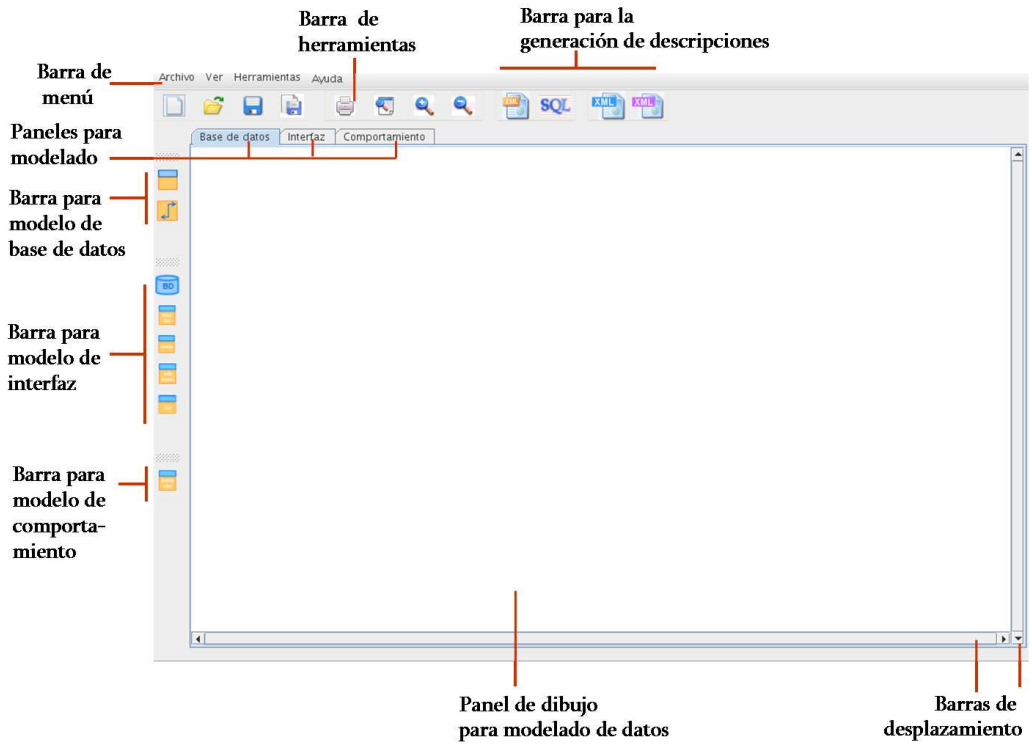


Figura 5.4: Componentes del entorno

9. áreas protegidas y de interés.

La capa geográfica sobre la que se trabajó es la capa de *hidrografía e infraestructura hidráulica* para tener un modelo de datos de esta capa que sirva como enlace para el diseño y la creación de *SIG* enfocados hacia predicción de flujos, monitoreo de la calidad del agua y erosión del suelo.

5.4.1. Modelo de la base de datos

El modelo de la base de datos consiste en representar las entidades y sus relaciones de manera gráfica para la generación de un diseño lógico, el cual engloba la estructura de la base de datos (*tablas, atributos, relaciones*) en lenguaje *SQL* que es ejecutado en un servidor de base de datos (*PostgreSQL*) para almacenar la información del sistema y realizar consultas.

Las entidades que integran el modelo de hidrografía esta dividido en entidades geográficas y entidades no geográficas como se puede apreciar en las figuras 5.5 y 5.6. La distinción entre uno y otro tipo de entidad radica en que las primeras contiene información georeferenciada especificada por puntos y coordenas y las segundas contiene información nominal acerca de la entidades geográficas. En la figura 5.7 muestra parte del modelo de la base de datos diseñado con nuestra herramienta.

Entidades geográficas	**Descripción**
ha_cuerposdeagua	Contiene información geográfica de las entidades canal, cuerpo de agua, estanque, salina, tanque de agua.
hl_corrientesdeagua	Comprende información geométrica acerca de acueducto, bordo, presa, rápido.
hp_rasgoshidrograficospuntuales	Describe información georeferenciada de manantial, salto de agua, tanque de agua.

Figura 5.5: Entidades geográficas del modelo de hidrología

.

5.4.2. Modelo de interfaz de usuario

Para un diseño eficiente de sistemas Web, el contenido y la presentación son aspectos que el diseñador no debe descuidar con el objetivo de realizar un sitio planificado, estructurado y usable. Si se tiene un modelo de interfaz de usuario predefinido la posibilidad de realizar cambios en la presentación y el diseño en una etapa posterior serán más rápidos y sencillos de aplicar y también se evitará afectar el funcionamiento

Entidades no geográficas	**Descripción**
acueducto	Tubería u obra civil sobre o bajo el terreno para la conducción de agua.
bordo	Canal construido en tierra con propósito de drenaje o irrigación.
canal	Tubería u obra civil sobre o bajo el terreno para la conducción de agua.
catalogo_capas	Contiene información acerca de las capas por medio de los campos layer y descripción
corrienteagua	Agua que fluye en río, lago o laguna.
corrienteenextincion	Un flujo natual de agua que florece de la superfice de la tierra.
cuerpoAgua	Un estancamiento natural de un curso de agua.
estanque	Una construcción hecha por el hombre para almacenamiento de agua.
manantial	Un flujo natual de agua que florece de la superfice de la tierra.
muroContencion	Una barrera sólida de material pesado utilizado como límite o protección.
Objeto	Permite hacer referencia a cierta región y capa a través de dos atributos (región y layer).
presa	Una barrera permamente que cruza un curso de agua utilizado para recoger agua.
rapido	Un flujo natural de un curso de agua de menor ancho que el río doble.
salina	Establecimentos con agua de mar que al evaporarse producen sal.
saltoAgua	Construcciones y artefactos destinados a aprovechar la caída del agua de un río, arroyo o canal.
tanqueAgua	Depósito para almacenar agua.

Figura 5.6: Entidades no geográficas del modelo de hidrología

.

del sitio.

El modelo de interfaz de usuario describe la presentación del sistema Web (e.g. titulos de encabezado, tamaño y tipo de letra, colores de letra, colores de fondo, menus, submenus). Las entidades que integran el modelo de interfaz de usuario son descritas en la figura 5.8.

5.4.3. Modelo de comportamiento

El modelo de comportamiento se refiere a la navegación entre las páginas del sistema Web. El comportamiento indica el resultado de realizar determinada acción (e.g. pulsar un botón, seleccionar un menu, seleccionar un submenu). Las entidades que componen este modelo estan definidas por las entidades geográficas y se muestra en la figura 5.9.

5.5. Diagramas UML

En la fase de diseño del sistema se utilizó *UML* (*Unified Modeling Language*), el cual es un lenguaje de modelado usado para expresar diseños en desarrollos orientados a objetos. Los tipos de diagramas utilizados para expresar el diseño son: diagrama de paquetes y diagramas de clases.

5.5.1. Diagrama de paquetes

El diseño de sistemas utilizando métodos estructurados conlleva el uso de la descomposición funcional, en la cual el sistema en su conjunto se correlacionaba como función y se dividía en subfunciones, que a su vez se dividían en otras subfunciones, y así sucesivamente. Con el surgimiento del paradigma de programación orientado

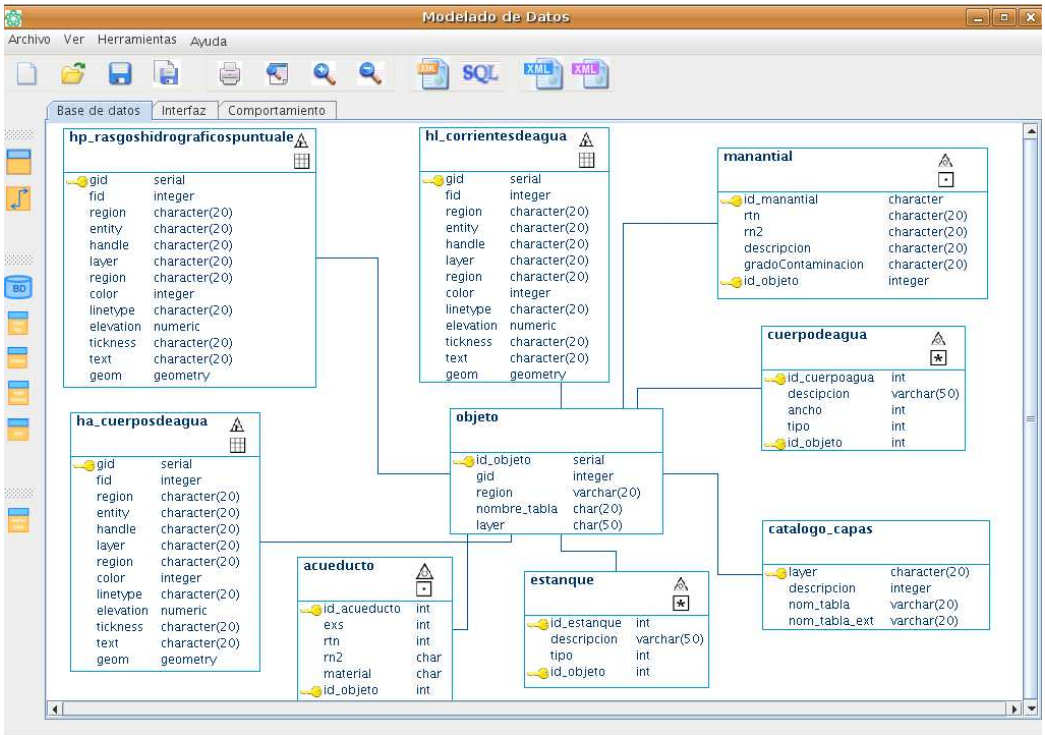


Figura 5.7: Modelo de datos de la base de datos

Entidades de interfaz	**Descripción**
ConexionBD	Representa información del servidor de base de datos para establecer conexiones.
interfaz	Describe las características de una interfaz de usuario (e.g. tipo de letra, tamaño, color, titulo).
menú	Contiene información acerca de los menús que integran el sistema y sus características. Los menús definidos en este modelo de datos son: *menucorrientes, menucuerpos, menurasgos.*
submenú	Describe las opciones que integran un submenú y sus propiedades. Las entidades de submenú especificadas son: *submenucorrientes, submenucuerpos, submenurasgos.*
paginajsp	Especifica los parámetros que forman parte de una página jsp.

Figura 5.8: Entidades del modelo de datos de interfaz de usuario

.

Entidades de comportamiento	**Descripción**
ha_cuerposdeagua	Describe el comportamiento de las entidades canal, cuerpo de agua, estanque, salina, tanque de agua.
hl_corrientesdeagua	Especifica el comportamiento acerca de acueducto, bordo, presa, rápido.
hp_rasgoshidrograficospuntuales	Describe el comportamiento de las entidades manantial, salto de agua, tanque de agua.

Figura 5.9: Entidades del modelo de datos de comportamiento

.

a objetos en lugar de realizar una descomposición funcional, la idea es agrupar las clases en unidades de nivel más alto, a este mecanismo de agrupamiento se le llama paquete. El diagrama de paquetes se usa para mostrar los paquetes de clases y las dependencias entre ellos y también permiten mantener el control sobre la estructura global de un sistema [23].

En la figura 5.10 se presenta el diagrama de paquetes que ofrece una vista sobre la estructura global del sistema. En él se muestran los paquetes de clases y las dependencias entre ellos.

Los paquetes que integran la aplicación son:

- *clases de la IGU (Interfaz Gráfica de Usuario).* Contiene los componentes que integran el diseño de la interfaz gráfica de usuario de la aplicación;
- *swing.* Comprende un conjunto de las API's de Java para el diseño e implementación de IGU;
- *clases de las formas de captura.* Definen la estructura de los formularios para la captura de datos de las entidades;
- *construye modelo.* Contiene las clases y métodos que definen la creación de las figuras (entidad y relación) que son representadas en los modelos de datos; abarca también la visualización y manipulación de las figuras en el área de dibujo;
- *clases para los métodos set/get.* Establecen y obtienen los valores de las variables de instancia utilizadas en el sistema;
- *clases para la creación de XML.* Engloban las clases y métodos para leer la información obtenida de las formas de captura y generar los archivos de descripción XML;

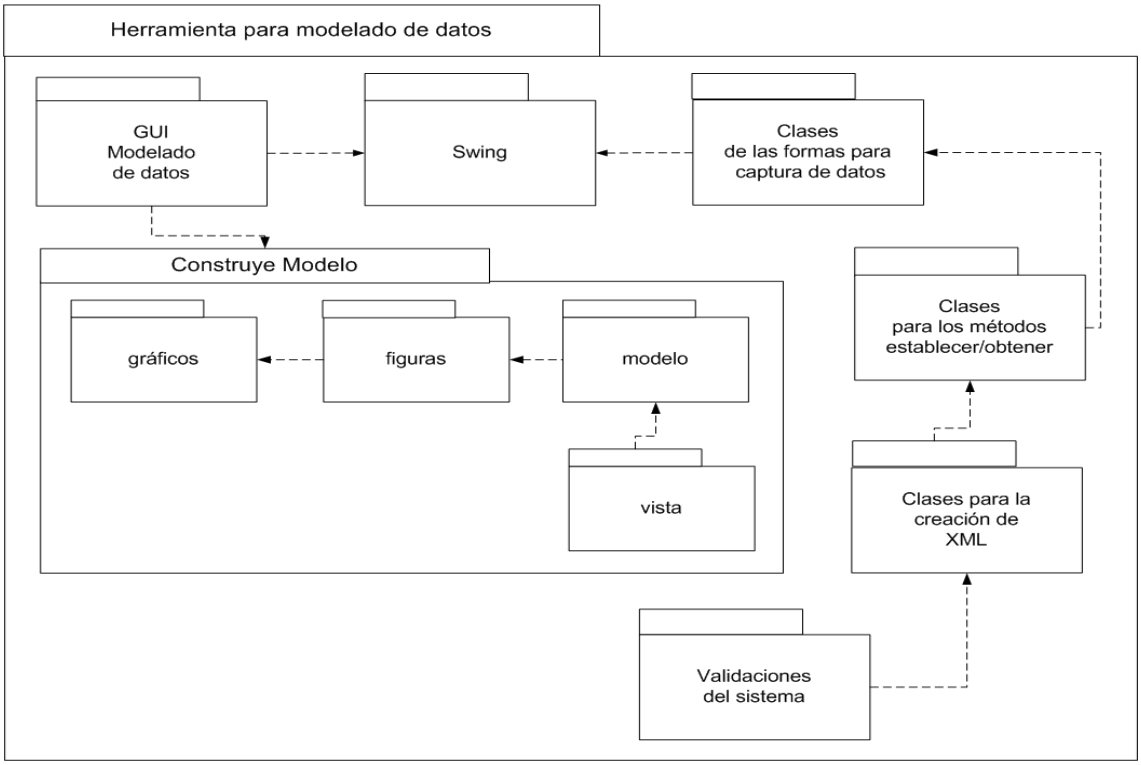


Figura 5.10: Diagrama de paquetes

- *validaciones del sistema*. Son clases cuya función es validar la entrada de datos al sistema y asegurar que estén en un formato específico.

5.5.2. Diagramas de clases

El diagrama de clase describe los tipos de objetos que hay en el sistema y las diversas clases de relaciones estáticas que existen entre ellos. Hay dos tipos de relaciones estáticas [23]:

- **asociación**. Es una relación estructural que describe una conexión entre objetos (e.g. un cliente puede rentar varias videocintas);
- **subtipos**. Es la relación entre una superclase y sus subclases (una enfermera es una persona).

Los diagramas de clase también muestran los atributos y las operaciones de una clase y las restricciones a que se ven sujetos, según la forma en que se conecten los objetos [23].

Los diagramas de clases que se mostrarán forman parte del diagrama de clases de todo el sistema, sólo se han tomado las partes más representativas para describir el diseño y funcionalidad de los módulos más importantes del sistema. El primer diagrama de clases contiene el diseño referente al editor de diagramas de modelos. El segundo diagrama de clases describe lo relacionado con el diseño para generar la estructura de la base de datos. Finalmente, el tercer diagrama de clases engloba el procedimiento para la generación de descripciones *XML*.

En la figura 5.11 se muestran las clases que brindan la funcionalidad para la parte del editor de diagramas de modelo, las clases *Figure, BoxedFigure, EntityFigure* engloban la lógica del diseño para dibujar una entidad, desplegar el nombre de la entidad y sus atributos. La clase *Figure* define los objetos que van a ser representados en el área de dibujo y la declaración de algunos métodos abstractos que serán utilizados posteriormente. Las clases *Entity y Attribute* almacenan los atributos que van a ser desplegados en la figura que representa una entidad. Las relaciones entre entidades son definidas en la clase *Conector*, y la zona donde será visualizado el modelo esta determinada por las clases *Canvas*. Se aprecia en el diagrama 5.11 que hay dos clases *Canvas*, el motivo es que la clase base *Canvas* contiene las funciones del manejo de coordenas de las figuras, la manipulación de una figura en el área de dibujo (adición, borrado, aumentar la escala de la imagén, disminuir la escala de la imagén). La subclase *Canvas* comprende la definición de los menus contextuales del *Canvas* y entidad, así como también la llamada a las funciones definidas en la clase base *Canvas*.

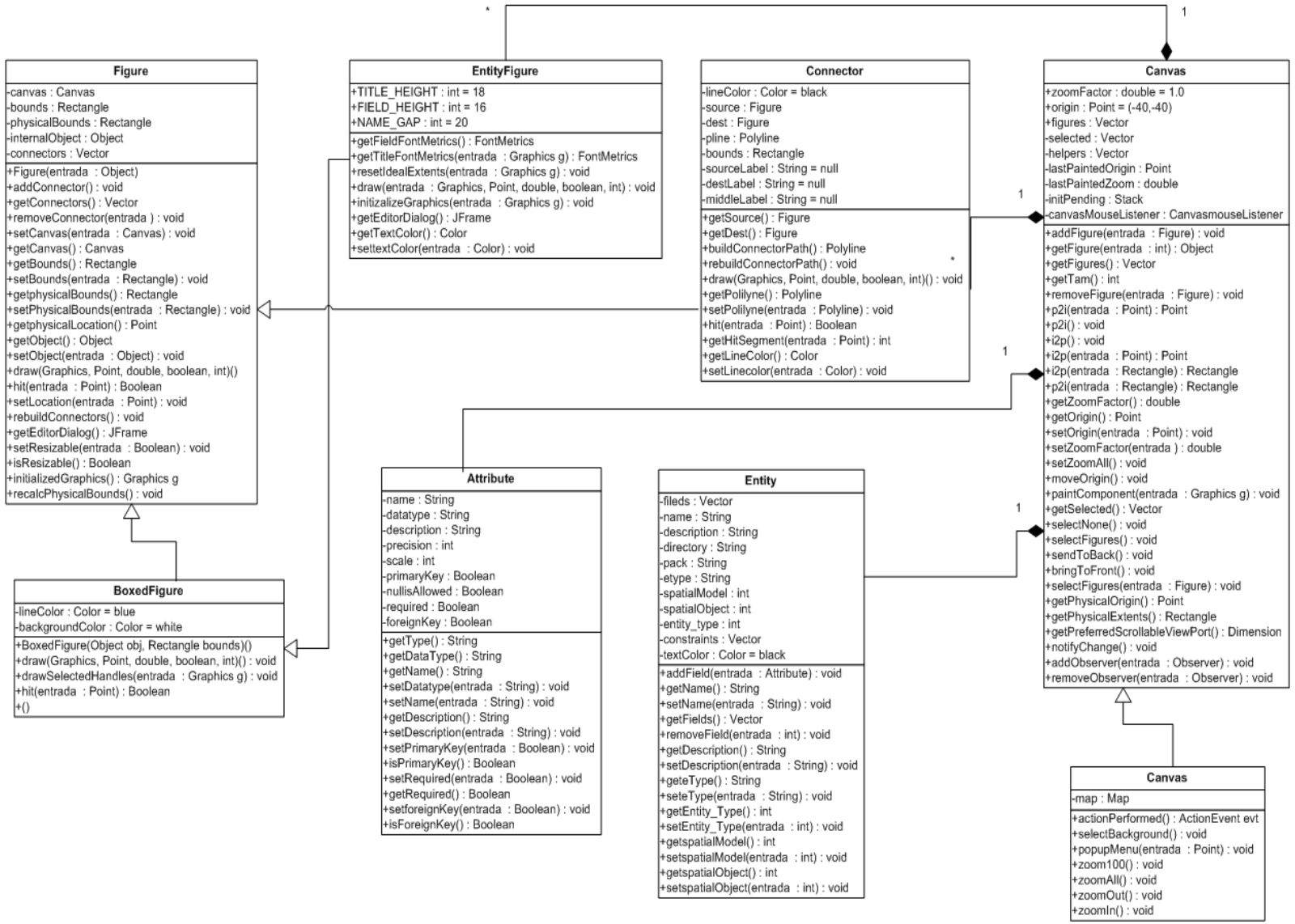


Figura 5.11: Diagrama de clases para el editor de diagramas

El proceso de creación del script de la base de datos se muestra en la figura 5.12. Una vez que se realizó el modelado de datos en el editor, éste es almacenado en un ArrayList y se envia como parámetro a la clase *BuildXMLEntities* , la cual se encarga de leer el arreglo y por medio del método *generateDocument()* genera el documento *XML* con la descripción de entidades de la base de datos. Una vez creado el documento *XML* se procede a generar el script de la base de datos pulsando el icono *SQL* en la interfaz gráfica de usuario. Al realizar esa acción, se ejecuta la interfaz *ListenerButonSQL* y se crea una instancia de la clase *TransformXMLtoSQL*, cuya tarea (a través de la función *xmlToSql*) es leer el archivo *styleScriptSQL.xsl*, el cual contiene la plantilla de la hoja de estilo para generar la estructura de la base de datos y después aplicar la transformación al documento *XML* para escribir el archivo con extensión *.sql*.

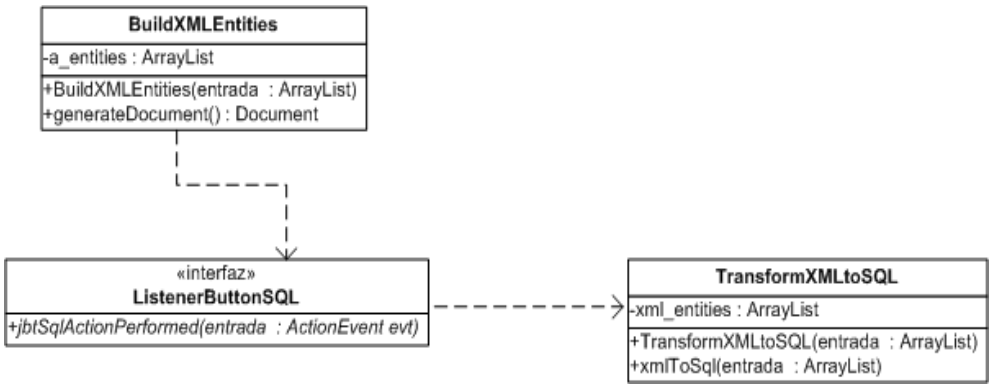


Figura 5.12: Diagrama de clases para la generación de la estructura de la base de datos

La generación de descripciones *XML* engloba el grupo de clases que se muestran en la figura 5.13. Éstas se encuentran dentro del paquete *BuildXML*; cada clase representa la generación de una descripción de acuerdo a los parámetros definidos para su creación. La funcionalidad es descrita a continuación:

- *BuildXMLEntities*, toma las entidades, relaciones y atributos de la base de datos realizados en el diagrama a través de un parámetro de tipo ArrayList que es enviado al construtor de la clase. En la función *generateDocument* se recorre el arreglo recibido y se van obtenido los valores para ir creando el documento *XML* de acuerdo a la estructura del esquema predefinido. Una vez que termina el recorrido del arreglo se procede la escritura del documento *XML*; el nombre del archivo es *InstanciaEntidad.xml*.

- *BuildXMLAppMod*, una vez establecidas las entidades y relaciones del modelo de la base de datos, a través de una pantalla de captura, se indican el nombre de la aplicación donde se almacenará el sistema, los nombres de los módulos con su respectivo subdirectorio. Esta información es guardada en un arreglo, éste es posteriormente leído para generar el archivo *AppModules.xml* cuyo contenido

es el nombre la carpeta raíz donde se establecera el sistema, los nombres de los módulos del sistema con su respectivo subdirectorio y la entidad geográfica a la que pertenece.

- *BuildXMLDatabase, BuildXMLInterface, BuildXMLMenu, BuildXMLSubmenu, BuildXMLJsp*, conforman las descripciones *XML* de interfaz de usuario. Las dependencias existentes entre la clase *BuildXMLApp*, se debe a que estos esquemas poseen un atributo dentro de los documentos *XML* que definen el nombre de la aplicación, los módulos y subdirectorios donde será almacenado el código creado a partir de dichas descripciones.

- *BuildXMLBehavior*, genera la descripción de comportamiento por cada entidad geográfica definida para el caso de estudio. Las entidades geográficas son: *ha_cuerposdeagua, hl_corrientesdeagua, hp_rasgoshidrograficospuntuales*. Para el proceso de generación de la descripción es necesario obtener información de los documentos *XML* creados por las clases *BuildXMLAppMod* y *BuildXMLJsp*.

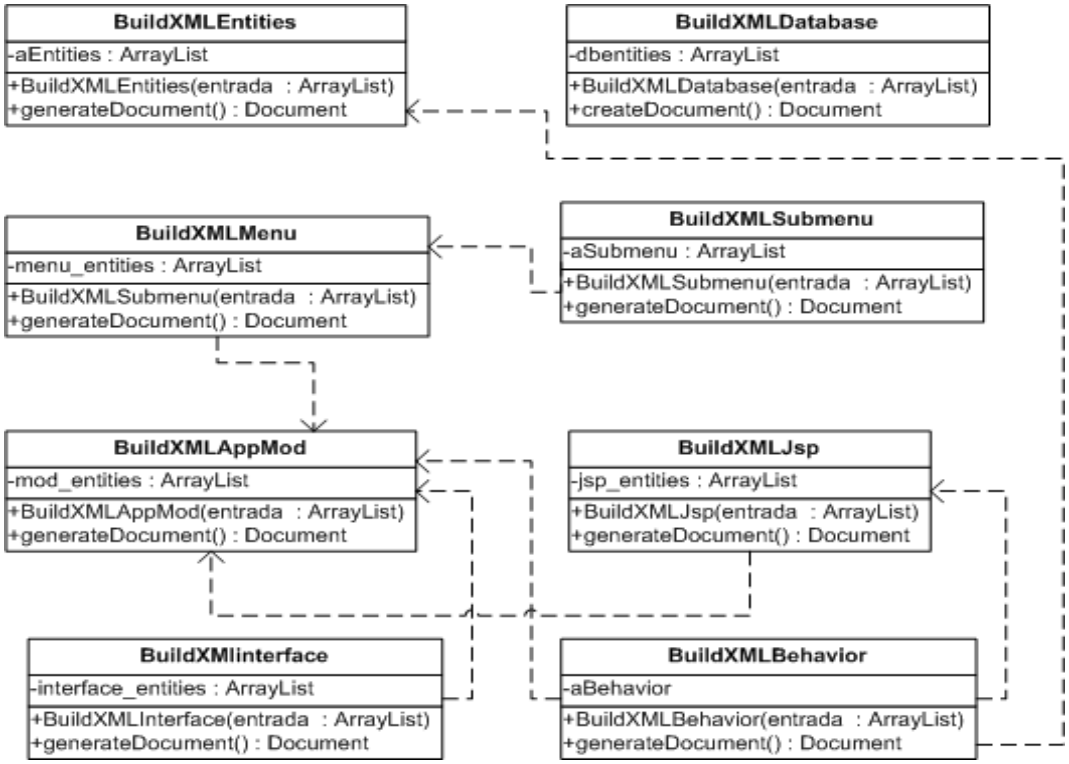


Figura 5.13: Diagrama de clases para la generación de descripciones XML

CAPÍTULO 6

Implementación

Una vez definidos los componentes que integran la arquitectura del sistema, se prodece a describir los aspectos de la implementación. Comenzamos con una breve descripción de los módulos que integran el sistema y la interacción entre ellos. Después explicamos el proceso de la implementación del editor de modelos, de la generación de la estructura de la base de datos y finalmente de las descripciones *XML*.

6.1. Descripción general

La implementación del modelador de datos esta integrada por tres módulos principales: el editor de modelos, el generador de la estructura de la base de datos, el generador de descripciones *XML*. Éstos proporcionan las funciones principales que describen la mayor parte del funcionamiento del sistema. El resto del trabajo es delegado a otras funciones que auxilian las tareas de las funciones primarias.

6.2. Implementación del editor

Este módulo es el encargado de realizar los diagramas de los modelos de datos, representarlos en el área de dibujo para su visualización y la manipulación de los mismos a través de un conjunto de clases descritas en la figura 5.11. Los principales métodos que se utilizan en este módulo son descritos en las figuras 6.1, 6.2 y 6.3.

El editor de diagramas esta integrado por un grupo de paquetes y clases que definen un *contexto de gráficos*[1] que permiten representar objetos gráficos (como líneas, elipses, rectángulos y otros polígonos). A continuación, describimos los paquetes que integran este módulo:

[1]Un contexto de gráficos permite dibujar en la pantalla [14].

- **paquete figures**. Esta compuesto por las clases que contienen las funciones que dibujan las entidades (atributos, tipos de datos, tipo de objeto geográfico) y las relaciones existentes entre éstas en el editor de modelos;

- **paquete graph**. Integrado por las clases cuya funcionamiento consiste en proporcionar los métodos para dibujo y pintado de objetos gráficos, manejo de coordenadas en el área de dibujo y manejo de eventos de ratón. También se incluyen métodos cambiar el color de fondo del canvas, el de línea, de fondo y el texto de las entidades;

- **paquete model**. Contiene los métodos "establecer" y "obtener" para manipular las propiedades de los objetos entidad y atributo que son dibujados dentro de la figura entidad y representados en el editor de diagrama;

- **paquete view**. Formado por las clases cuyos métodos proporcionan la visualización de los objetos gráficos en el área de de dibujo, la definición y llamada de menús contextuales.

El procedimiento para el desarrollo de los modelos de datos en la herramienta desarrollada es similar para los tres tipos de modelos definidos en el capítulo 5. La similitud radica en la representación gráfica designada para una entidad dentro del panel de dibujo. La diferencia consiste en que, en el modelo de la base de datos, se establecen relaciones (conectores) entre las entidades, a diferencia del modelo de la interfaz y del comportamiendo en donde sólo se representan las entidades sin ningún vínculo relacionado entre ellas. El proceso es descrito en la figura 6.4.

Una parte importante dentro de la herramienta de modelado es la representación de un componente entidad como una pieza fundamental del modelo de datos. Esta es simbolizada a través del objeto gráfico *Rectángulo*. La razón por la cual se eligió este objeto para representar una entidad es la de contar con una notación gráfica estándar similar a otras herramientas de modelos de datos como *UML* específicamente tomando como referencia la representación empleada en los diagramas de clases. El contenido de la información dentro de la figura entidad esta integrada por su nombre, las figuras que indican el tipo de información geográfica que representa, los nombres de los campos y sus tipos de datos, el icono que indica si es llave primaria o llave foránea. La implementación para la creación de una entidad con sus atributos y el pintado de ésta en el panel de dibujo se muestra en la figura 6.5.

El editor de diagramas cuenta con dos menús contextuales, un menú contextual que pertenece al panel del dibujo (ver figura 6.6) y el otro es para introducir las características de una entidad en el modelo de datos (ver figura 6.7).

6.3. Implementación del script SQL de la base de datos

Este módulo sólo tiene una función que recibe como parámetro el archivo *XML* que contiene la descripción de la base de datos y también la ruta donde se encuentra la hoja de estilo (*styleScriptSQL.xsl*) que contiene las transformaciones que se aplicarán al documento *XML* para la generación del script de la base de datos.

La hoja de estilo realiza el proceso de transformación por medio de los lenguaje *XSLT* y *XPath*. Ésta inicia con el elemento raíz, después se especifica el formato de salida del documento. Finalmente, se define la plantilla que contiene la tranformación del documento, cuyos atributos e instrucciones están asociados con el contenido del archivo *XML* que va a permitir generar el archivo final con extensión *.sql* en donde estará alojada la estructura de la base de datos.

En la parte de la hoja de estilo donde se especifica la plantilla, se usa el lenguaje *XPath* para definir las partes del documento *XML* que deben coincidir con las propiedades precisadas en las plantillas. Cuando una coincidencia es encontrada, *XSLT* aplica la tranformación al documento *XML* para generar el documento resultante (*databasescript.sql*).

6.4. Implementación de descripciones XML

En lo que respecta al módulo para la generación de descripciones *XML* éste, se encuentra dividido en tres partes:

- clases para la generación de la descripción de la base de datos;
- clases para la generación de las descripciones de interfaz de usuario;
- clases para la generación de la descripción de comportamiento.

Con estas tres partes se realiza la creación del modelo de datos desde tres perspectivas (*base de datos, interfaz de usuario y comportamiento*) en lo que se refiere a descripciones *XML*. Existen también otras clases que generan archivos *XML* los cuales proporcionan apoyo para la generación de las descripciones. Las clases mencionadas anteriormente son las más importantes, ya que en ellas se encuentra implementada la funcionalidad para la generación de documentos *XML* con la información requerida. La figura 6.8 presenta de manera gráfica el desarrollo de este proceso.

6.4.1. Descripción de base de datos

La clase que permite la creación del archivo de descripción de base de datos es *BuildXMLEntities.java*, la cual recibe como parámetro un arreglo *ArrayList* que contiene la información de las entidades que integran el modelo de la base de datos. Esta información es obtenida del arreglo por medio de variables declaradas en el método *createDocument()* cuya función es la creación del documento *XML*. Para ello, se utilizó el procesador *dom4j* (sección 3.4) que proporciona una biblioteca de funciones para el manejo de documentos *XML*.

En la tabla 6.1 se presenta el contenido de los métodos que forman parte de la clase *BuildXMLEntities.java*. La función de ésta consiste en construir el documento *XML* que va a contener la descripción de la estructura de la base de datos, la información es adquirida por medio de una pantalla de captura y almacenada en un arreglo el cual es pasado como parámetro al constructor de la clase. La pantalla de captura esta dividida en tres secciones la primera, implica información básica de la entidad como su nombre, una breve descripción, como el caso de estudio es un *SIG*, por lo tanto el modelo de la base de datos implica la definición de entidades goegráficas y no geográficas. Para diferenciar si una entidad es geográfica o no, se elige el valor *geography* o *none* por medio de una lista desplegable. La segunda sección, consiste en establecer los nombres de los atributos de la entidad, elegir su tipo de datos, marcar si el campo es una llave primaria o llave foránea. La tercera sección engloba la elección del tipo de modelo espacial (*modelo de campo, modelo de objeto*), y de una primitiva de representación espacial (e.g. punto, línea), para ser visualizadas en la figura entidad por medio de un icono, se puede apreciar las tres secciones mencionadas en la figura 6.9.

Método	**Descripción**
BuildXMLEntities(ArrayList aEntities)	Es el constructor de la clase y recibe como parámetro el arreglo que contiene las entidades, relaciones y atributos para creación del documento *XML*.
generateDocument()	Función cuya tarea es obtener los valores del arreglo e ir formado el documento *XML* para escribirlo en un archivo.

Tabla 6.1: Métodos de la clase BuildXMLEntities

.

La figura 6.12 especifica el desarrollo de la plantilla para crear la hoja de estilo que se aplicó al documento que contiene la descripción de la estructura de la base de datos para obtener el archivo con extensión .sql, el método encargado de realizar las llamadas al documento XML y la hoja de estilo es descrita en la figura 6.11.

Finalmente, se puede apreciar en la figura 6.10 un fragmento de la descripción *XML* del modelo de base de datos.

6.4.2. Descripciones de interfaz de usuario

Los métodos que forman parte de la descripción de interfaz de usuario se muestran en la tabla 6.2. El conjunto de clases que forman estas descripciones se presentan en el diagrama 5.13 y las relaciones de dependencia existente entre ellas. El proceso de cómo se realizó la creación de los esquemas de interfaz de usuario son mostrados en las figuras 6.13, 6.14, 6.15, 6.16 y 6.17.

A continuación se muestran y describen brevemente los esquemas generados para la interfaz de usuario. Después de crear el documento *XML* que contiene la descripción de la base de datos, se continua con la descripción de la información que permita establecer una conexión a base de datos. La información es adquirida por medio de una pantalla de captura que requiere ser llenada por el diseñador del sistema. Los datos requeridos más importantes son: el controlador de base de datos, la dirección IP del servidor donde se encuentra instalada la base de datos, el puerto de conexión, el nombre de usuario, contraseña. Puede verse en la figura 6.18 otros atributos del esquema. Para asegurarse que la información introducida es la correcta, se aplican validaciones a la forma de captura. La función *matches* de la clase *String* brinda esta funcionalidad.

Método	**Descripción**
BuildXMLDatabase(ArrayList adbentities)	Constructor de la clase que recibe como parámetro un arreglo que contiene la información para establecer una conexión a base de datos.
createDocument()	Esta función lee los datos guardados en el arreglo y, con base a la estructura predefinida para el esquema de base de datos, se genera el documento *XML*.
BuildXMLMenu(ArrayList amenu_entities)	Es el constructor de la clase y recibe como argumento un arreglo que comprende los menús que forman parte del sistema.
generateDocument()	La función realiza la transformación del contenido del arreglo en varios archivos *XML*, dependiendo del número de módulos del sistema.
BuildXMLSubmenu(ArrayList aSubmenu)	El constructor de la clase recibe el arreglo que contiene la información de los submenús a través del parámetro *aSubmenu*.
generateDocument()	Brinda la funcionalidad de generar los archivos de descripción de submenús, al igual que las descripciones de menús los archivos son generadas por separado, dependiendo de los mdulos existentes en el sistema.
BuildXMLJsp(ArrayList jsp_entities)	Constructor de la clase que recibe un arreglo como argumento, el cual contiene el nombre las páginas jsp y la ruta donde serán almacenados.
generateDocument()	Genera el documento *XML* que contiene el nombre de las páginas jsp y las clases que van integradas (beans).

Tabla 6.2: Métodos para la generación de descripciones de interfaz

.

Método	**Descripción**
Figure(Object obj)	Constructor de la clase Figure que recibe como parámetro un objeto que va a ser representado en el canvas.
addConnector(Connector c)	Es un arreglo que contiene los conectores de las entidades dibujados en el diagrama.
draw(Graphics g, Point p, double z, boolean isSelected)	Dibuja la entidad en el canvas y también carga la información de la misma (nombre de la entidad y atributos).
setCanvas()	Asigna el canvas al objeto que se va a graficar.
setBounds(Rectangle r)	Asigna la posición/tamaño del objeto en coordenadas ideales.
boolean hit(Point p)	Determina si un punto físico coincide con la imagen del objeto en coordenas físicas.
setLocation(Point p)	Mueve la figura (entidad) a la localización del punto p.
initializeGraphics(Graphics g)	Es llamado para inicializar el objeto cada vez que va a ser graficado en el canvas.
setBackgroundColor(Color color)	Establece el color de fondo de la entidad.
setLineColor	Define el color de línea de la entidad.
setTextColor(Color c)	Establece el color de texto de la entidad.
Entity(String name)	Constructor de la clase Entity que recibe como parámetro el nombre de la entidad.
addField(Attribute field)	Función que recibe como argumento el nombre del atributo de la clase Attribute y lo agrega a un arreglo de atributos.

Figura 6.1: Métodos para crear el editor de modelos

.

Método	**Descripción**
setName(String name)	Se encarga de asignar el nombre a una entidad.
setDescription(String setdesc)	Establece la descripción de una entidad.
setEntity_Type(int et)	Asigna el tipo de entidad (*catalogo, geography, none*).
setspatialModel(int spm)	Define el tipo de modelo espacial al que pertenece la entidad (*Objeto geográfico, Campo geográfico*).
setspatialObject(int spo)	A través de esta función se asigna el tipo de objeto espacial que corresponde a la entidad *Punto, Línea, Polígono, etc.*
Attribute(String n, String t, Boolean req, Boolean pk, Boolean fk)	Constructor de la clase Attributo y recibe como parámetro el nombre del atributo, el tipo de dato, si es requerido y si es llave primaria o llave foránea. Esta clase contiene las propiedades que describen los atributos de una entidad.
setName(String name)	Asigna el nombre de un atributo de la entidad.
setDataType(String sdtype)	Establece el valor del tipo de dato de un atributo.
setRequired(Boolean req)	Es utilizado para definir si un atributo (campo) es requerido.
setPrimaryKey(Boolean k)	Si el atributo es llave primaria le asigna a la variable un valor booleano true o false.
setforeignKey(Boolean fk)	Si el atributo es llave foránea establece el valor de la variable a true o false.

Figura 6.2: Métodos para crear el editor de modelos

.

Método	**Descripción**
Canvas()	Inicializa el canvas con el color de fondo establecido. Activa los *listener* para el manejo de eventos de movimiento del ratón.
addFigure(Figure f)	Agrega una figura al canvas.
removeFigure(Figure f)	Remueve una figura y los conectores de la figura.
addMouseHelper	Define el tipo de modelo espacial al que pertenece la entidad (*Objeto geográfico, Campo geográfico*).
setOrigin(Point p)	Establece el origen de la pantalla en las coordenadas ideales.
setZoomFactor(double z)	Cambia el factor de zoom manteniendo el centro de la pantalla.
sendToBack()	Cambia la ubicación de la figura. Para ello la borra del canvas y la vuelve a agregar por medio de un arreglo.
bringToFront	Se encarga de traer una figura al frente si dos figuras se encuentran traslapadas.

Figura 6.3: Métodos para crear el editor de modelos

.

Entrada. La instancia del objeto entidad a gráficar.
Salida. El dibujo de la entidad dentro del panel de dibujo.
Procedimiento. Crear modelo.
{
Inicializar todos los objetos gráficos y funciones para el
el dibujo de las entidades y también los escuchadores de eventos
al ejecutar la aplicación
Iniciar la creación del modelo
Repetir:
Pulsar en el icono para dibujar la entidad
En ese momento crear la instancia entidad y dibujar la figura
sin información dentro de ella en el panel de dibujo
Introducir información a la entidad a través de una forma de
captura que se despliega al dar clic sobre el menú contextual de
la entidad
Definir a través de esa forma el nombre de las entidad, de los
atributos, de los tipos de datos y del tipo de objeto geográfico
Por último, guardar la información escrita
Hasta crear todas las entidades que componen el modelo de datos
}

Figura 6.4: Procedimiento para la creación de modelos en el contexto gráfico.

Entrada. Información de la entidad (nombre y atributos),
los bordes de un rectángulo.
Salida. El dibujo de la entidad en el canvas.
Procedimiento. Creación de la figura entidad.
{
Inicializar el color de línea y el color de texto de la entidad
Establecer el tipo y el tamaño de letra del título de la entidad y sus atributos
A través de una función obtener el nombre de la entidad y sus atributos
para calcular la longitud del texto y así poder establecer la dimensión
de la figura entidad
Implementar un método abstracto que permite dibujar la figura en el
canvas. Algunos parámetros que recibe son: el contexto gráfico,
un punto de origen, un valor de tipo booleano.
Las operaciones que realiza el método son:
Obtener el nombre de la entidad y sus atributos
Establecer el color de fondo de la figura y llenar con dicho color
Establecer el color del borde de la figura y dibujar el rectángulo
Configurar el tamaño del nombre de la entidad y el tipo de letra
misma que se asigna al contexto gráfico
Dibujar el nombre de la entidad dentro de los márgenes (x, y) precisados
Dibujar la línea que separa el nombre de la entidad de los atributos
entre los puntos (x1, y1) y (x2, y2) definidos
Establecer el tamaño del nombre de los atributos y el tipo de letra
misma que se asigna al contexto gráfico
Leer el vector que contiene los atributos, éste contiene el nombre
y el tipo de dato
Si el atributo es llave primaria o llave foránea, dibujar la imagén
de una llave dentro de la figura entidad
Dibujar dentro de la figura entidad el nombre del atributo
y el tipo de dato en la coordenada (x,y) establecida
Dibujar la figura por primera vez en el canvas, ejecutar una función
que permite inicializar el objeto dentro de un contexto gráfico
}

Figura 6.5: Procedimiento para la creación de una entidad.

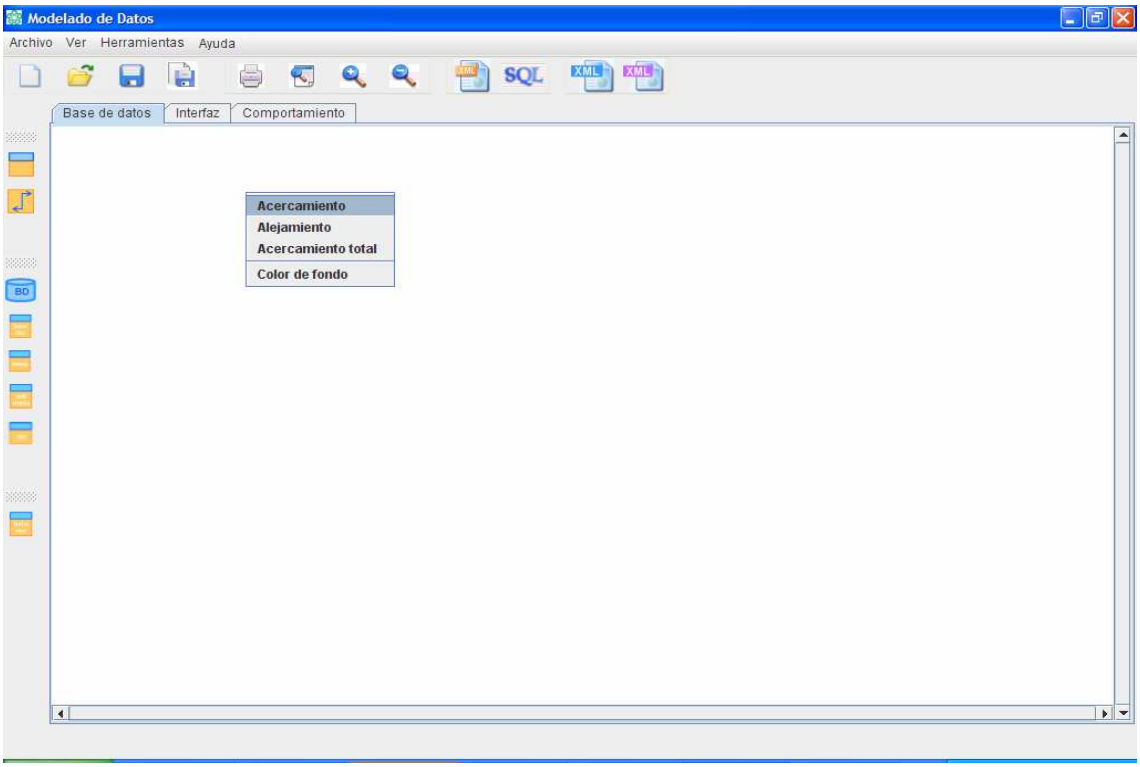


Figura 6.6: Menú contextual del panel de dibujo

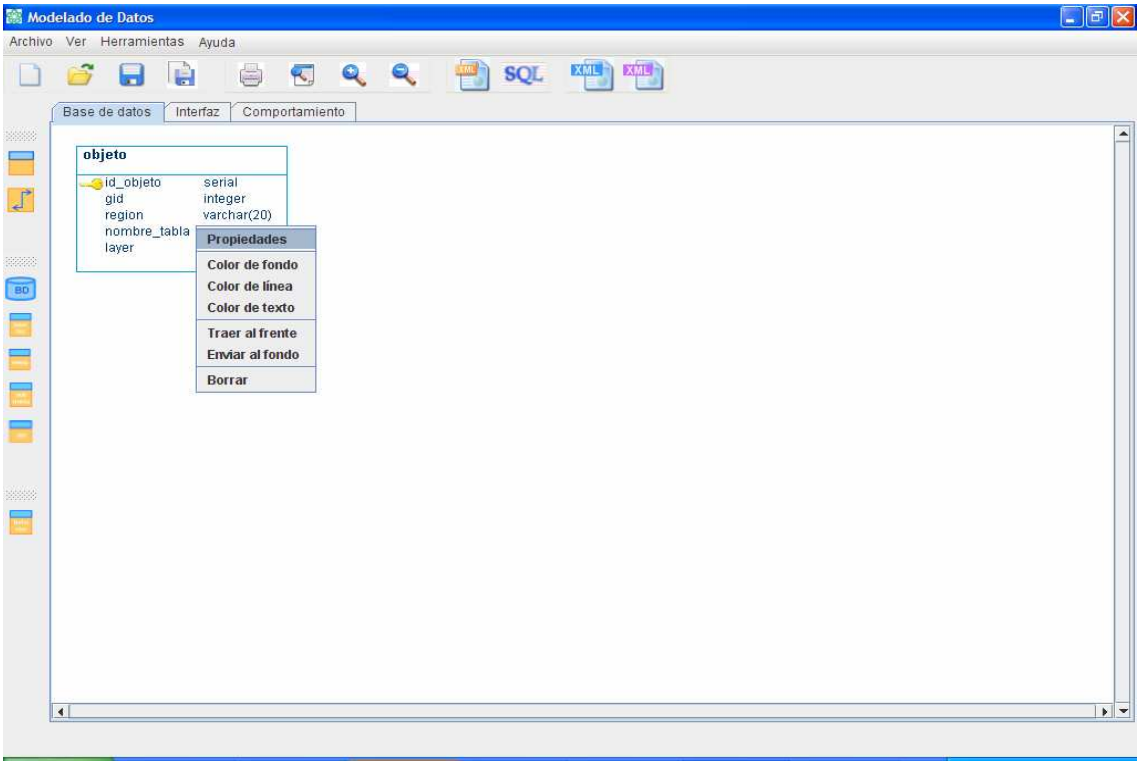


Figura 6.7: Menú contextual de la entidad

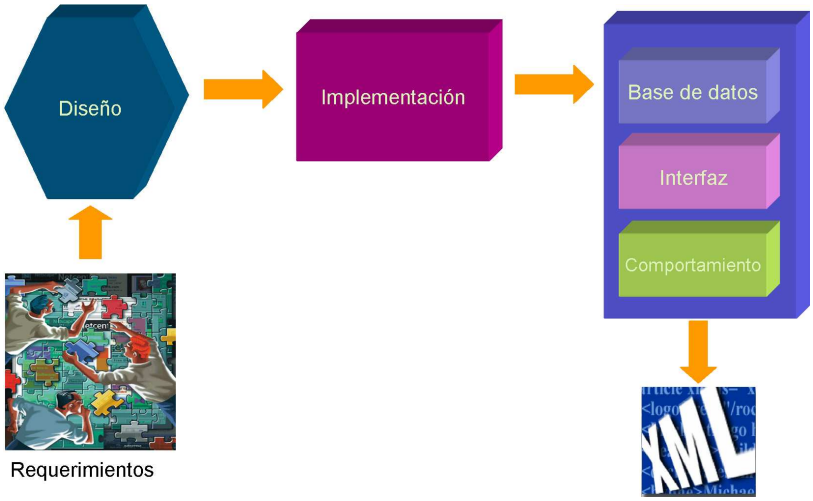


Figura 6.8: Proceso de generación de documentos XML

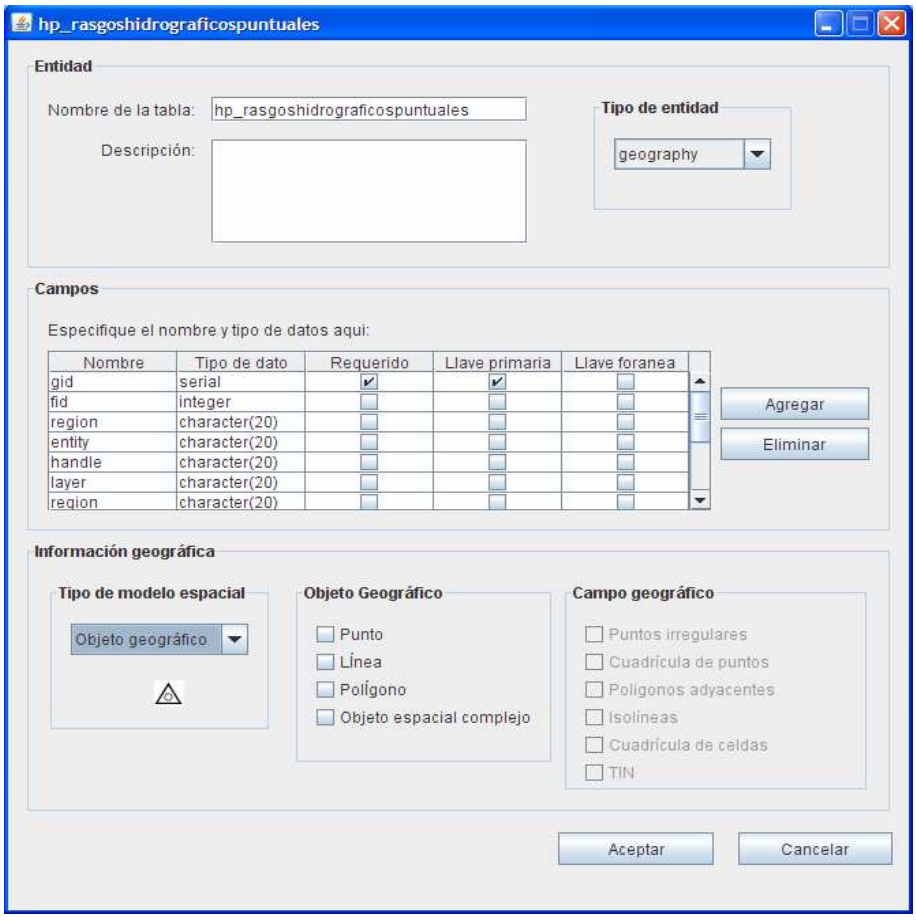


Figura 6.9: Pantalla de captura para los datos de la entidad

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

<entities
        <entity name="Cuerpo_de_agua" table="ha_cuerposdeagua" directory="Application/WEB-INF/classes/bc/entity/"
pack="bc.entity" type="geography">
            <property name="gid" type="serial" column="gid" primaryKey="true" id="gid" nulo="false"/>
            <property name="fid" type="integer" column="fid_" id="entity"/>
            <property name="region" type="character" column="region" id="region"/>
            <property name="entity" type="character" column="entity" id="entity"/>
            <property name="handle" type="character" column="handle" id="handle"/>
            <property name="layer" type="character" column="layer" id="layer"/>
            <property name="color" type="integer" column="color" id="color"/>
            <property name="linetype" type="character" column="linetype" id="linetype"/>
            <property name="elevation" type="numeric" column="elevation" id="elevation"/>
            <property name="thickness" type="numeric" column="thickness" id="thickness"/>
            <property name="text" type="character" column="text" id="text"/>
            <property name="geom" type="geometry" column="the_geom" id="geometry"/>
        </entity>
        <entity name="Corriente_de_agua" table="hl_corrientesdeagua" directory="Application/WEB-INF/classes/bc/entity/"
pack="bc.entity" type="geography">
            <property name="gid" type="serial" column="gid" primaryKey="true" id="gid" nulo="false"/>
            <property name="fid" type="integer" column="fid_" id="entity"/>
            <property name="region" type="character" column="region" id="region"/>
            <property name="entity" type="character" column="entity" id="entity"/>
            <property name="handle" type="character" column="handle" id="handle"/>
            <property name="layer" type="character" column="layer" id="layer"/>
            <property name="color" type="integer" column="color" id="color"/>
            <property name="linetype" type="character" column="linetype" id="linetype"/>
            <property name="elevation" type="numeric" column="elevation" id="elevation"/>
            <property name="thickness" type="numeric" column="thickness" id="thickness"/>
            <property name="text" type="character" column="text" id="text"/>
            <property name="geom" type="geometry" column="the_geom" id="geometry"/>
        </entity>
```

Figura 6.10: Descripción XML del modelo de base de datos

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

<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
    <xsl:output method="text" indent="yes"/>

    <!-- TODO customize transformation rules.
         syntax recommendation http://www.w3.org/TR/xslt.
    -->
    <xsl:template match="/">
        <xsl:apply-templates/>
    </xsl:template>
    <xsl:template match="table">
        CREATE TABLE <xsl:value-of select="@name"/>.
        <xsl:text>&#40;</xsl:text>

      <xsl:for-each select="column">
        <xsl:variable name="null" select="@required"/>
        <xsl:variable name="key" select="@primaryKey"/>
        <xsl:variable name="fkey" select="@foreignKey"/>

        <xsl:value-of select="@name"/>
        <xsl:text>&#32;</xsl:text>
        <xsl:value-of select="@type"/>
        <xsl:value-of select="@size"/>
        <xsl:if test ="$null = 'true'"> NOT NULL</xsl:if>
        <xsl:if test ="$key = 'true'"> PRIMARY KEY </xsl:if>
         <xsl:if test ="$fkey = 'true'"> REFERENCES </xsl:if>
           <xsl:value-of select="@table_ext"/>
        <xsl:if test="not(position()=last())">,</xsl:if>
        <xsl:text>&#32;</xsl:text>
     </xsl:for-each>
        <xsl:text>&#41;</xsl:text>
        <xsl:text>&#59;</xsl:text>
    </xsl:template>
```

Figura 6.11: Hoja de estilo para la creación de la estructura de la base de datos

1) Se inicia la plantilla de la hoja de estilo
2) Repetir
    a) Para cada atributo “table” del nodo “entity” encontrado escribir CREATE TABLE y abrir paréntesis izquierdo
    b) Aplicar la plantilla repetidamente para todos los elementos “property” del documento xml
    c) Declarar las variables para obtener los valores de llave primaria, llave foránea y si el campo es requerido
    d) Obtener el nombre y tipo de dato de los campos de las tablas
    e) Verificar si el campo es nulo, llave primaria o llave foránea para escribir NOT NULL, PRIMARY KEY o REFERENCES
    f) Si es llave foránea, obtener el nombre de la tabla con que tiene relación
    g) Cerrar paréntesis izquierdo y colocar punto y coma
Hasta leer todos los nodos del documento xml

Figura 6.12: Definición de la hoja de estilo para aplicar las transformaciones.

Entrada. El arreglo que contiene con la información.
Salida. El archivo de descripción XML con la
información para conectarse a una base de datos.
Procedimiento. Generar xml para conexión a BD.
{
  El arreglo recibido como parámetro es asignado a otra variable de tipo arreglo
  Crear el método de tipo Document que regresara el archivo xml
  y se lleva a cabo lo siguiente:
    Declarar las variables que van a contener la información
    que se encuentra dentro del arreglo
    Iniciar la creación del documento xml
    Leer el arreglo y se obtienen sus elementos, mismos
    que son asignados a las variables ya declaradas
    Crear las etiquetas del documento xml de
    de acuerdo a la estructura definida en los requerimientos
    El documento creado se envia a un archivo especificando la ruta
    donde es almacenado
  Regresar el documento xml y finalizar el método
}

Figura 6.13: Procedimiento para la creación de la descripción xml para conectarse a una base de datos.

Entrada. El arreglo que contiene los datos de la interfaz de usuario.
Salida. El archivo de descripción XML con la
información de la interfaz de usuario del subsistema.
Procedimiento. Generar xml de la interfaz de usuario.
{
El arreglo recibido como parámetro es asignado a otra variable de tipo arreglo
Crear el método de tipo Document que regresara el archivo xml
y se lleva a cabo lo siguiente:
Declarar las variables que van a contener la información
que se encuentra dentro del arreglo
Iniciar la creación del documento xml
Leer el arreglo y obtener sus elementos, mismos
que son asignados a las variables ya declaradas
Crear las etiquetas del documento xml de
de acuerdo a la estructura definida en los requerimientos
El documento creado se envia a un archivo especificando la ruta
donde será almacenado
Regresar el documento xml y terminar el método
}

Figura 6.14: Procedimiento para la creación de la descripción xml de la interfaz de usuario.

Entrada. El arreglo que contiene la información de los menús.
Salida. Los archivos de descripción xml con la
información de los menús del sistema.
Procedimiento. Generar los archivos xml de los menús.
{
El arreglo recibido como parámetro se asigna a otra variable de tipo arreglo
Crear el método de tipo Document que regresara los archivos xml
y realizar lo siguiente:
Declarar las variables que van a contener la información
que se encuentra dentro del arreglo
Iniciar la creación del documento xml
Repetir:
Recorrer el arreglo que almacena la información
de los menús y obtener sus elementos para asignarselos
a las variables ya declaradas
Construir la parte de parámetros y encabezado
de la descripción xml de menú
Si existen uno o dos espacios entre la sección de
parámetros y las opciones del menú colocar la(s) etiquetas
que indican espacios en blanco
Generar las etiquetas de las opciones del menú
Verificar si existen espacios entre las opciones de menú para
colocar las etiquetas que indican espacios en blanco
Revisar si el menú lleva la opción para consultar un mapa,
colocar la etiqueta en la estructura del documento
Establecer la etiqueta de la opción "Salir" del subsistema
Establecer la etiqueta del fin del menú
Crear un archivo de descripción por cada menú en un documento xml,
él cual se almacena en la ruta especificada
Hasta que se haya leido todo el arreglo que contiene la información
de los menús del sistema
}

Figura 6.15: Procedimiento para la creación de las descripciones xml de menús.

Entrada. El arreglo que contiene la información de los submenús.
Salida. Los archivos de descripción xml con la
información de los submenús del sistema.
Procedimiento. Generar los archivos xml de los submenús.
{
  El arreglo recibido como parámetro se asigna a otra variable de tipo arreglo
  Crear el método de tipo Document que regresará los archivos xml
  y se lleva a cabo lo siguiente:
    Declarar las variables que van a contener la información
    que se encuentra dentro del arreglo
    Iniciar la creación del documento xml
    Repetir:
      Recorrer el arreglo que almacena la información
      de los submenús y obtener sus elementos para asignarselos
      a las variables ya declaradas
      Construir un elemento submenú de la descripción xml
      con la sección de parámetros y encabezado
      Generar las etiquetas de las opciones del submenú
      Crear un archivo de descripción por cada submenú en un documento xml,
      él cual se almacena en la ruta especificada
    Hasta que se haya leido todo el arreglo que contiene la información
    de los submenús del sistema
}

Figura 6.16: Procedimiento para la creación de las descripciones xml de submenús.

Entrada. El arreglo que contiene la información acerca las páginas jsp.
Salida. El archivo de descripción xml con la
información de las páginas jsp del sistema.
Procedimiento. Generar el archivo xml de páginas jsp.
{
  El arreglo recibido como parámetro se asigna a otra variable de tipo arreglo
  Crear el método de tipo Document que regresará el archivo xml
  y se lleva a cabo lo siguiente:
    Declarar las variables que van a contener la información
    que se encuentra dentro del arreglo
    Iniciar la creación del documento xml
    Repetir:
      Recorrer el arreglo que almacena la información
      de las páginas jsp y obtener sus elementos para asignarselos
      a las variables ya declaradas
      Construir un elemento "module" de la descripción xml
      Generar los subelementos "page" con sus propiedades
      Verificar si éstas páginas contienen algún bean
      conexion, consulta, generalxml para integrarlos
      como subelementos de "page"
    Hasta que se haya leido todo el arreglo que contiene la información
    de las páginas jsp del sistema
    Agregar un elemento "page" denominado "Catalogo" con los beans
    de *conexion, consulta, generalxml*
    Generar el documento xml en donde se escribe la estructura creada
    en el ciclo anterior especificando la ruta donde será almacenado
}

Figura 6.17: Procedimiento para la creación de la descripción xml de páginas jsp.

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

<DBconnection>
      <database>
        <driver>org.postgresql.*</driver>
        <ip>148.247.102.138</ip>
        <db>hidrologia</db>
        <user>colima</user>
        <pass>pwdcolima</pass>...
        <directory>Application/WEB-INF/classes/bean/DB/</directory>
        <package>bean.DB</package>
        <method>yes</method>
      </database>
</DBconnection>
```

Figura 6.18: Descripción para la conexión de base datos

La generación del esquema de *interfaz de usuario* es creado tomando en cuenta que cada entidad geográfica es considerada un módulo del sistema. Es decir, dependiendo de las entidades geográficas existentes en el modelo de datos son el número de interfaces de usuario creadas y enviadas al archivo de descripción *XML*. Las partes que integran la descripción de una interfaz de usuario son:

- *cenefa*, las propiedades que se definen son ancho, alto, tipo y tamaño de letra, título de la cenefa, imagén o logotipo, fecha y hora;
- *menu*, esta parte proporciona las dimensiones del menú (ancho y alto);
- *submenu*, esta sección establece las dimensiones del submenú (ancho y alto);
- *despliegue*, esta parte determina el área de visualización total de la pantalla.

Estas mismas propiedades son requeridas en una pantalla de captura dentro de la aplicación de modelado de datos al momento de crear la entidad de interfaz. Una vez capturada toda la información de cada interfaz se genera el documento *XML* con el nombre *interfaz.xml*.

Los menús son elementos que integran un sistema, en ellos se describen las opciones y la funcionalidad que contiene un sistema. En la herramienta de modelado la representación de los menús en el modelo de la interfaz de usuario se realiza a través de la entidad *menús*. Por cuestiones de diseño, se optó por definir el contenido de cada menú en un archivo por separado. Al igual que en la descripción de interfaz, por cada entidad geográfica del modelo de base de datos se define un módulo del sistema, lo mismo sucede para los menús que integran el sistema. Para nuestro caso de estudio lo documentos *XML* de menús generados son tres (*menucorrientes, menucuerpos, menurasgos*).

La pantalla de captura diseñada para obtener la información requerida que permita crear los esquemas de menús contiene las siguientes secciones:

- *nombre de menú*, en esta parte se debe seleccionar el módulo del sistema al que pertenece el menú y escribir una descripción breve de lo que va a contener ese menú;
- *parámetros del menú*, aqui el usuario debe introducir información relacionada con las características generales del menú tales como el tipo, tamaño y color de fuente, así como el título del menú;
- *opciones de menú*, en esta sección se especifican los nombres de las opciones del menú, el color de fondo de la opción, el nombre de la página jsp hacia donde será enviado el usuario al dar clic sobre él;
- *opción Mapa*, si se trata de un *SIG* esta opción debe estar activa ya que indica el nombre de menú, de preferencia debe estar relacionado con la palabra *Mapa* porque esas opción desplegará una página jsp con un Mapa;
- *opción Salir*, requiere las características del menú salir como el nombre, color de fondo, nombre de la página jsp hacia donde enviará esta opción.

Una vez capturada la información de la entidad *menús* del modelo de interfaz los datos son almacenados en un arreglo, posteriormente el arreglo es leido para generar los documentos *XML menucorrientes.xml, menucuerpos.xml, menurasgos.xml*. En la figura 6.20 se muestra un fragmento de la descripción menucorrientes.xml.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<interfaces_principales>
    <interfaz name="rasgos" directory="Application/RHP/" color="#007800">
        <cenefa>
            <ancho>82</ancho>
            <largo>518</largo>
            <!--Se definen las caracteristicas de la cenefa-->
            <titulo font="arial" size="3" fontcolor="black">Hidrograf&#237;a e Infraestructura
Hidraulica</titulo>
            <extratitulo font="arial" size="2" fontcolor="white">CINVESTAV</extratitulo>
            <tiempo>true</tiempo>
            <fecha>true</fecha>
            <logo>../imagenes/cinvestavc.gif</logo>
        </cenefa>
        <menu>
            <ancho>665</ancho>
            <largo>135</largo>
        </menu>
        <submenu>
            <ancho>32</ancho>
            <largo>*</largo>
        </submenu>
        <despliegue>
            <ancho>32</ancho>
            <largo>*</largo>
        </despliegue>
    </interfaz>
    <interfaz name="corrientes" directory="Application/CVC/" color="#CCCC00">
        <cenefa>
            <ancho>82</ancho>
            <largo>518</largo>
            <!--Se definen las caracteristicas de la cenefa-->
            <titulo font="arial" size="3" fontcolor="black">Corrientes y V&#237;as de
```

Figura 6.19: Descripción de interfaz de usuario

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<menus>
  <parametros>
        <fuente>arial</fuente>
        <colorfuente>#FFFFFF</colorfuente>
        <tamafuente>11</tamafuente>
  </parametros>
  <encabezado id="I">
        <colorfuente>white</colorfuente>
        <colorfondo>#999900</colorfondo>
    <titulo>Corrientes y V&#237;as de conducci&#243;n de agua</titulo>
  </encabezado>
  <menu id="0">
    <clase>espacio</clase>
  </menu>
  <menu id="1">
    <nombre>Acueducto</nombre>
        <colorfondo>#CCCC00</colorfondo>
    <link>menu/submenuacueductocorrientes.jsp</link>
    <target>submenu</target>
    <tooltip>Acueducto</tooltip>
    <clase>menu</clase>
  </menu>
  <menu id="2">
    <nombre>Bordo</nombre>
        <colorfondo>#CCCC00</colorfondo>
    <link>menu/submenubordocorrientes.jsp</link>
    <target>submenu</target>
    <tooltip>Canal</tooltip>
    <clase>menu</clase>
  </menu>
  <menu id="2">
    <nombre>Canal</nombre>
        <colorfondo>#CCCC00</colorfondo>
```

Figura 6.20: Descripción de menú

La creación de submenús sigue un enfoque parecido a la construcción de menús en el sentido de que se crea el mismo número de archivos *XML*. De acuerdo a las entidades geográficas existentes son los módulos desarrollados y también las descripciones *XML* de los menús generados. Para la creación de la descripción de submenús de un módulo del sistema se debe saber cuales son las opciones de menú de ese módulo. En nuestra herramienta de modelado se muestra una pantalla de captura en la cual aparecen previamente cargados los nombres de los archivos *XML* que contienen descripciones de menú. Enseguida, selecciona la opción a partir de la cual se crearán los submenús y de ahi se determinan las propiedades como el tipo, color y tamaño de letra y las opciones del submenú (e.g. Agregar, Eliminar, Modificar, Consultar). En la figura 6.21 se expone una parte de la descripción *submenuacueductocorrientes*.

Otra entidad que forma parte del modelo de interfaz de usuario es la de Pages_jsp. Ésta contiene información relacionada con el nombre de las páginas jsp y los *beans* asociadas a ella. Cada página debe tener al menos un *bean* especificado. Los beans existentes son: *conexion, consulta, xml*. Al igual que las otras descripciones que forman parte de la interfaz de usuario, la de páginas jsp es por cada módulo que integran el sistema. En la figura 6.22 se muestra un fragmento de esta descripción.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<submenus name="submenuacueductocorrientes" directory="Application/CVC/menu/">
 <parametros>
       <fuente>arial</fuente>
       <colorfuente>#FFFFFF</colorfuente>
       <tamafuente>11</tamafuente>
 </parametros>
 <encabezado id="I">
   <colorfuente>white</colorfuente>
       <colorfondo>#999900</colorfondo>
   <titulo>Acueducto</titulo>
 </encabezado>. .
 <submenu id="1">
   <nombre>Agregar</nombre>
       <colorfondo>#CCCC00</colorfondo>
   <link>behavior/InsertarAcueducto.jsp</link>
   <target>contenido</target>
       <tooltip>Agregar Acueducto</tooltip>
   <clase>submenu</clase>
 </submenu>
 <submenu id="2">
   <nombre>Eliminar</nombre>
       <colorfondo>#CCCC00</colorfondo>
   <link>behavior/EliminarAcueducto.jsp</link>
   <target>contenido</target>
       <tooltip>Eliminar Acueducto</tooltip>
   <clase>submenu</clase>
 </submenu>
 <submenu id="3">
   <nombre>Modificar</nombre>
       <colorfondo>#CCCC00</colorfondo>
   <link>behavior/ModificarAcueducto.jsp</link>
   <target>contenido</target>
       <tooltip>Modificar Acueducto</tooltip>
```

Figura 6.21: Descripción de submenú

```
<?xml version="1.0" encoding="UTF-8"?>
  <module name="corrientes">
    <page name="ListaAcueducto" location="Application/CVC/menu/behavior/" js="../../comunes/js/js.js">
          <useBean id="conexion" class="bean.DB.ConexionBD"/>
          <useBean id="consulta" class="bean.DB.ArmQuery"/>
          <useBean id="generaxml" class="bean.ArmXML"/>
    </page>
        <page name="MapaAcueducto" location="Application/CVC/menu/behavior/" js="../../comunes/js/js.js">
          <useBean id="conexion" class="bean.DB.ConexionBD"/>
          <useBean id="consulta" class="bean.DB.ArmQuery"/>
          <useBean id="generaxml" class="bean.ArmXML"/>
    </page>
        <page name="ListaBordo" location="Application/CVC/menu/behavior/" js="../../comunes/js/js.js">
          <useBean id="conexion" class="bean.DB.ConexionBD"/>
          <useBean id="consulta" class="bean.DB.ArmQuery"/>
          <useBean id="generaxml" class="bean.ArmXML"/>
    </page>
        <page name="DetalleAcueducto" location="Application/CVC/menu/behavior/" js="../../comunes/js/js.js">
          <useBean id="conexion" class="bean.DB.ConexionBD"/>
          <useBean id="consulta" class="bean.DB.ArmQuery"/>
          <useBean id="generaxml" class="bean.ArmXML"/>
    </page>
        <page name="InsertarAcueducto2" location="Application/CVC/menu/behavior/" js="../../comunes/js/js.js">
          <useBean id="conexion" class="bean.DB.ConexionBD"/>
          <useBean id="consulta" class="bean.DB.ArmQuery"/>
    </page>
        <page name="EliminarListaAcueducto" location="Application/CVC/menu/behavior/" js="../../comunes/js/js.js">
          <useBean id="conexion" class="bean.DB.ConexionBD"/>
          <useBean id="consulta" class="bean.DB.ArmQuery"/>
          <useBean id="generaxml" class="bean.ArmXML"/>
    </page>
        <page name="EliminarDetalleAcueducto" location="Application/CVC/menu/behavior/" js="../../comunes/js/js.js">
          <useBean id="conexion" class="bean.DB.ConexionBD"/>
          <useBean id="consulta" class="bean.DB.ArmQuery"/>
```

Figura 6.22: Descripción de página jsp

Finalmente, el modelo de datos de interfaz de usuario que generan las descripciones mencionadas en los párrafos anteriores se muestra en la figura 6.23.

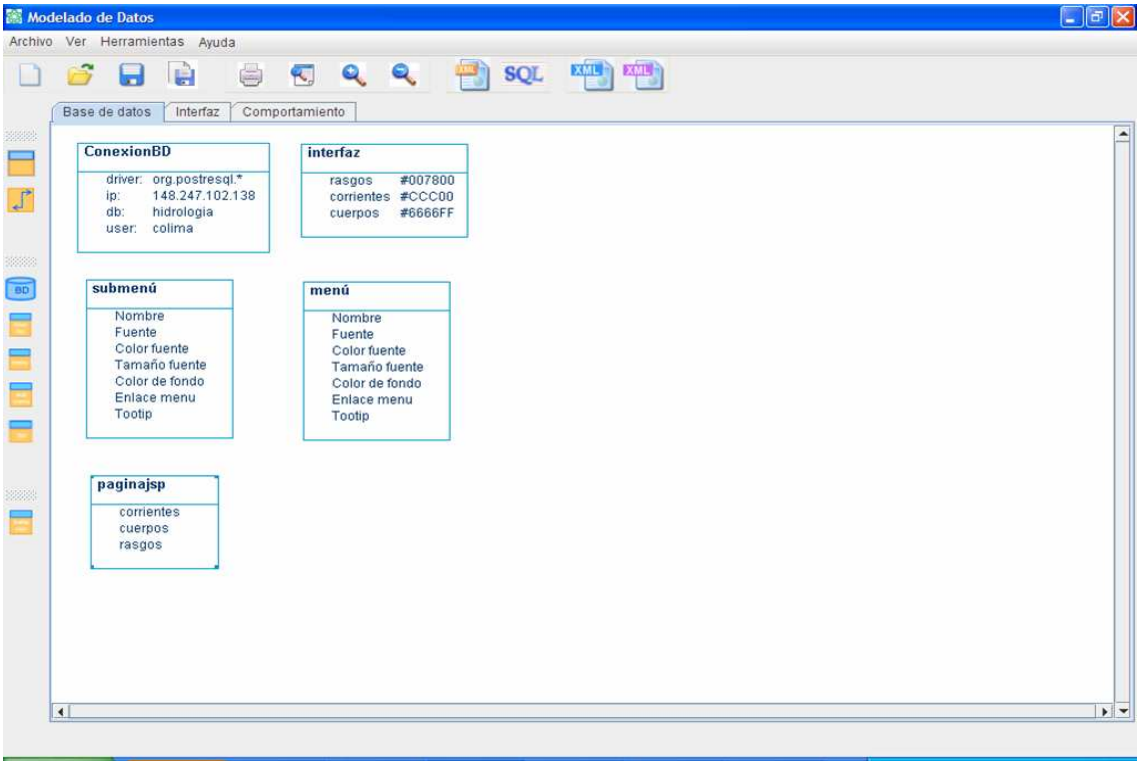


Figura 6.23: Modelo de interfaz de usuario

6.4.3. Descripción de comportamiento

Una vez definido el modelo de base de datos y el modelo de interfaz de usuario del sistema con la herramienta de modelado quedan cubiertos los aspectos del manejo de datos y la presentación del sistema. Sin embargo, la interacción entre todas la páginas del sistema no esta determinada, por lo tanto el esquema de comportamiento va a aportar, esa funcionalidad con el fin de tratar de generar las descripciones del sistema lo más completas posibles.

La estructura del esquema de comportamiento principalmente se encuentra dividido por módulos del sistema y dentro de cada módulo se encuentran definidas las opciones del sistema. Entonces por cada opción de sistema existen operaciones como consultar, agregar, eliminar, modificar. Debido a que las acciones que realiza cada operación son diferentes la interacción entre las páginas también lo será. La opción de consultar el esquema de comportamiento se encuentra dividida en cuatro secciones: *forma básica, forma de mapa, forma de resultado*, la opción agregar contiene la sección forma de transacción, la operación eliminar comprende las opciones *forma*

básica, forma de resultado, forma de detalle y forma de transacción. Finalmente, la operación modificar contiene las mismas opciones que la operación eliminar.

CAPÍTULO 7

Conclusiones y trabajo a futuro

En este capítulo abordamos las conclusiones obtenidas del desarrollo de nuestro trabajo de tesis. Asimismo, mencionamos las contribuciones realizadas con esta investigación y finalmente las propuestas consideradas para realizar una extensión de dicha investigación.

7.1. Conclusiones

La ingeniería de software desempeña un papel importante en el desarrollo de sistemas, ya que a través de ella es posible la aplicación de un enfoque sistemático, disciplinado y cuantificable hacia el desarrollo, la operación y el mantenimiento del software. El contar con métodos o procedimientos dentro de una organización que ayuden a la construcción del software es determinante para la generación de productos de calidad. Una vez establecido algún procedimiento para el desarrollo de software resulta conveniente que estos procesos sean ágiles y soporten los cambios a los requerimientos. Las herramientas *CASE* ofrecen estas características y por lo tanto la construcción de herramientas similares a éstas pueden considerarse como una opción para apoyar el proceso desarrollo de software.

El diseño de base de datos forma parte activa en el desarrollo de un sistema de información. Con el objetivo de contar con una metodología y un sistema que ayude a la automatización del proceso en el diseño de base de datos se inicio el proyecto *CADBD* y el desarrollo de éste ha implicado diversos trabajos para brindarle solución al problema (sección 1.2.2). El proceso de desarrollo de bases de datos es un trabajo que requiere un análisis detallado de las variables y restricciones que intervienen en su diseño. El objetivo es crear una base de datos con la menor redundancia de datos y establecer las relaciones necesarias entre las entidades que integran la base de datos. Por ello, es importante seguir una metodología de diseño de base de datos para obtener un buen esquema lógico de ésta así, cuando se realicen operaciones en ella

sean de una forma eficiente.

Por otra parte están los *SIG* los cuales son cada vez más importantes en las ciencias de la computación por su naturaleza multidisciplinaria y la diversidad de conceptos tecnólogicos (bases de datos, computación gráfica, computación de alto desempeño) que involucran. Específicamente, la ingeniería de software y su fase de requerimientos son fundamentales para la difusión de este tipo de sistemas. Sin embargo, la falta de conocimiento tecnológico no solo de los usuarios, sino de los mismos diseñadores de estos sistemas y el hecho de que las metodologías existentes de ingeniería de software no fueron concebidas originalmente teniendo en cuenta la información georeferenciada. Debido a ello ha surgido la necesidad de desarrollar metodologías y herramientas para poder satisfacer las necesidades de especificación de los *SIG* que incluyan apoyo para el manejo de la información georeferenciada, interoperabilidad, metadatos y estándares geográficos de dicha información.

El manejo de una gran cantidad de datos complejos en base de datos en áreas como *SIGs*, sistemas de información ambiental, biología; conlleva el uso de tipos de datos con una organización y estructura de acuerdo a la naturaleza de la información almacenada en este tipo de sistemas. En el diseño y desarrollo de sistemas utilizados para el almacenamiento, el manejo y la explotación de información que maneja de datos complejos, es importante definir la forma lógica de los datos que serán procesados por el sistema, y una forma de lograrlo es especificando un modelo de datos por medio del cual la información del dominio del sistema es representada siguiendo una semántica.

Para el manejo y representación de la información geográfica es conveniente implementar estándares para el acceso e interoperabilidad de este tipo de información con instituciones internacionales. Esta área, dentro de los *SIG*, ha recobrado mucha importancia debido al aumento de datos espaciales y la preocupación por el control de la calidad. Por ello el trabajo realizado dió seguimiento a estándares en información geográfica. El estándar utilizado para representar la información geográfica fue tomado del *ISO/TC 211 y OGC*.

Las herramientas y tecnologías de desarrollo actuales (*Java, XML, dom4j, IDEs*), para el diseño y programación de aplicaciones, han brindado los componentes con la funcionalidad requerida para la implementación de la herramienta de modelado donde el manejo de funciones gráficas, la manipulación de archivos y el uso de procesadores *XML* para la generación de descripciones son parte esencial de la programación del sistema desarrollado en este trabajo. Además éstas tecnologías son de software de distribución gratuita que brindan el soporte y la documentación para su manejo de manera gratuita sin tener que pagar las licencias por utilizarlas.

Nuestro trabajo de tesis se ha enfocado en el modelo de datos de un *SIG*. El

propósito es contar con una metodología integrada en una herramienta que administre la información relacionada con estos sistemas y además forme parte del proceso del software. El proceso de desarrollo de un sistema comprende la consideración de diversas variables para obtener un producto de software lo más cercano posible a los requerimientos del usuario y con fallas mínimas. Algunos factores considerados para el desarrollo de nuestra herramienta de modelado es que fuera un sistema portable capaz de ejecutarse en cualquier plataforma (*Linux, Windows, Mac*), simple para brindar al usuario una herramienta que se pueda utilizar sin complicaciones y escalable para poder seguir agregando más funcionalidad al sistema a través de módulos o subsistemas. Todos estos factores forman parte de nuestro sistema de modelado de datos. La organización y clasificación de la información geográfica no es una tarea sencilla, sin embargo se dedicó un gran esfuerzo para representar el modelo de la base de datos de la capa de hidrología en nuestra herramienta, obteniendo un modelo que cumple con las expectativas de lo requerido.

El desarrollo de este trabajo de tesis aporta soluciones para el desarrollo de aplicaciones orientadas a *SIG* tomando en cuenta los aspectos de proceso de desarrollo de sofware, proceso de diseño de base de datos, estándares para el manejo de información geográfica, uso de herramientas y tecnologías portables de software de distribución gratuita. También representa una pieza fundamental en la investigación de soluciones para el diseño de bases de datos que manejan datos complejos y el diseño de *SIGs* complementados con metadatos y estándares de información.

7.2. Contribuciones

El trabajo realizado presenta ciertas novedades en el proceso de diseño de sistemas en lo referente al modelado de datos, En el contexto de los SIGs, tales novedades han resultado en las siguientes contribuciones:

- se ha diseñado una metodología para la creación de sistemas a partir de un modelo de datos;
- nuestro trabajo ha adoptado el estándar *ISO/TC 211* por medio del esquema conceptual GeoFrame para la representación de objetos geográficos en el modelo de datos (entidades geográficas). El objetivo final es tener interoperabilidad con otros sistemas geográficos;
- se ha construido una herramienta que permite desarrollar modelos de datos considerando tres perspectivas: base de datos, interfaz de usuario y comportamiento. También permite generar las descripciones *XML* relaciondas con cada perspectiva.

A continuación discutiremos brevemente las contribuciones:

La metodología que hemos desarrollado considera la arquitectura propuesta y el proceso de creación del modelo de datos (sección 5.2) para la producción eficaz y eficiente de un producto de software. La ausencia de la definición de un proceso en el desarrollo de sistemas implica la falta de integración de la tareas y esto conlleva a incosistencia y confusiones en el proceso. El funcionamiento de la metodología se aplica en el contexto de un *SIG*, sin embargo realizando ciertas modificaciones en algunos parámetros y atributos, nuestra metodología se puede considerar en el diseño de sistemas convencionales que no hacen uso de información geográfica.

En relación a la adopción del estándar *ISO/TC 211* nuestro trabajo consiste en usar las normas establecidas para representar la información geográfica en la herramienta de modelado. Esto permitirá tener compatibilidad con productos de otras instituciones que hagan uso del estándar y por lo tanto la posibilidad de compartir información entre diferentes aplicaciones. Otra ventaja en el uso de estándares es que permite estructurar de manera homologada la información geográfica con sus características espaciales y no espaciales, como las coordenadas del sistema de referencia, la geometría, la topología, las unidades de medida y el tiempo entre otros. Nuestra herramienta desarrollada representa la información geográfica considerando estándares definidos, si no de una manera completa si toma las características básicas y esenciales de dicha información.

Nuestra herramienta ha sido desarrollada con el objetivo de proveer un sistema que permita la creación de modelos de bases de datos y sistemas de forma rápida. Con esto se muestra que es posible diseñar un sistema a partir del uso de notaciones gráficas para la creación de sus descripciones en formato *XML*, mismas que serán leidas por un generador de código para obtener un conjunto de subsistemas. Nuestro trabajo también tiene el objetivo de mejorar el proceso del software mediante la unificación de las etapas (*diseño, codificación, mantenimiento*) a través de una herramienta de alto nivel.

7.3. Trabajo a futuro

El desarrollo de este trabajo de tesis pone en evidencia una etapa que puede ser retomada para darle continuidad a proyectos relacionados con las áreas de ingeniería de software, bases de datos y desarrollo de sistemas de información enfocados al manejo de datos complejos. El objetivo es desarrollar herramientas que ayuden al diseño y manipulación de la información que usan datos complejos, buscar soluciones para la organización y representación de los datos en campos como biología, química, *SIG* y otros.

Nuestra herramienta de modelo de datos sólo representa la información de un sistema geográfico, los diseños conceptual, lógico y físico son considerados en la parte de generación de la estructura de la base de datos. Las entidades que representan

una interfaz de usuario, y el comportamiento del sistema están predefinidas en unos esquemas *XML* que se procesan por medio de un generador de código. Consideramos importante hacer notar que se pueden construir ciertos módulos o subsistemas para extender la funcionalidad de la herramienta como los siguientes:

- el diseño de un *diccionario de datos*, será de gran utilidad para administrar toda la información proveniente de todos los tipos de datos de los modelos, las ventajas de utilizar un diccionario de datos son: proporcionar un mecanismo para administrar nombres, verificando la exclusividad del nombre de las entidades, relaciones, y tipos de datos. Por otra parte sirve como depósito de información de la organización que vincula el análisis, el diseño, la implementación y la evolución de un sistema;

- un *subsistema para captura de requerimientos*, que permita definir las especificaciones del sistema para establecer lo que éste hara y no la manera en que se implementará. También ayudan a mantener una organización de los diferentes subsistemas que componen el sistema.

Así, nuestra herramienta sería complementada para que el proceso de diseño de sistemas sea más rápido y adaptable a los cambios en los requerimientos. Como se puede apreciar, nuestra herramienta ofrece la ventaja de limitar el consumo de tiempo necesario para la construcción de modelos de datos y sistemas. De esta manera las tareas que se desarrollaban de manera manual o en etapas separadas serán desarrolladas de manera integrada y un poco más automatizada. Nuestra herramienta implementa una metodología básica para desarrollar modelos de manera clara y definida. Esta herramienta puede ser modificada para poseer un sistema de desarrollo más poderoso si agregamos más funcionalidades que permitan extender la utilidad de nuestro sistema para contemplar más factores relacionados con el desarrollo de software.

Estamos convencidos de que nuestro trabajo puede ser objeto de muchas mejoras, sin embargo marca una etapa importante en el desarrollo de software en el contexto de los *SIGs*.

APÉNDICE A

Diagrama de clases

BIBLIOGRAFÍA

[1] Minyar Sassi Amel Grissa-Touzi. New approach for the modeling and the implementation of the object-relational database. *Journal Computing and Technology*, pages 263–266, 2005.

[2] Nectaria Tryfona Christian S. Jensen Anders Friis-Christensen. Requirements and research issues in geographic data modeling. *ACM Press*, pages 2–8, Noviembre 2001.

[3] María Mercedes Marqués Andrés. Metodología de diseño de bases de datos. http://www3.uji.es/ mmarques/f47/apun/node81.html, 2001.

[4] Krishnakumar Balasubramian. Developing applicatins using model-driven design environments. *IEEE Computer*, pages 33–40, 2006.

[5] Luis Fernando Medina Cardo. Hacía una metodología para ingeniería de requerimientos en sistemas de información geográfica. Technical report, Universidad Nacional de Colombia.

[6] Francisco Javier Ceballos. *JAVA 2. Interfaces gráficas y aplicaciones para Internet*. Ra-Ma, México D.F, segunda edición edition, 2006.

[7] Open Geospatial Consortium. Opengis. http://www.opengeospatial.org/, 2007.

[8] Germán Vargas Cuervo. Evaluación de imágenes de satélite spot-landsat y sar ers-1 en la cartografía de movimientos en masa. Technical report, 1994.

[9] Salvador Trujillo; Don Batory; Oscar Diaz. Feature oriented model driven development: A case study for portlets. *IEEE Computer Society*, Mayo 2007.

[10] Pedro Enrique Alday Echavarria. Diseño de base de datos con evex entidad vínculo extendido para xwindows. Master's thesis, CINVESTAV, México, D.F, Febrero 1997. Departamento de Ingeniería Eléctrica Seccion de Computación.

[11] Bruce Eckel. *Piensa en Java*. Prentice Hall, Madrid, España, segunda edición edition, 2002.

[12] Enciclopedia encarta. Sistema de información geográfica. http://es.encarta.msn.com/encnet/refpages/RefArticle.aspx?refid=761579503, 2007.

[13] PostgreSQL Global Development Group. Postgresql documentation. http://www.postgresql.org/docs/8.2/interactive/intro-whatis.html, 2007.

[14] Paul J. Deitel Harvey M. Deitel. *Cómo programar en Java*. Pearson Educación, México, quinta edición edition, 2004.

[15] S. Sudarshan Henry F. Korth, Abraham Silberschatz. *Fundamentos de bases de datos*. 1998.

[16] INEGI. Bases de datos geográficas. Technical report, Junio 1999.

[17] Marie-Josée Proulx Jean Brodeur, Yvan Bédard. Modelling geospatial application databases using uml-based repositories aligned with international standars in geomatics. *ACM Press*, pages 39–46, Noviembre 2000.

[18] Juan Trujillo Jose-Norberto Mazon. Applying mda to the development of data waerehouses. *ACM Press*, (57-66), Noviembre 2005.

[19] Cirano Iochpe Jugurta Lisboa Filho. Specifying analysis for geographic databases on the basis of a conceptual framework. *ACM Press*, pages 7–13, Noviembre 1999.

[20] MetaStuff Ltd. Frequently asked questions. http://www.dom4j.org/faq.html#whats-dom4j, 2003.

[21] Ernst Denert Manfred Broy. *Software Pioneers*. 2002.

[22] Merche Marqués. Diseño de sistemas e bases de datos, Abril 2002.

[23] Kendall Scott Martin Flower. *UML gota a gota*. Prentice Hall, 1999.

[24] Dr. Guillermo De Ita Mc. Carlos Guillén, Dr. Aurelio López. Diseño de algoritmos combinatorios para #sat y su aplicación al razonamiento proposicional. Technical report, INAOE, 2005.

[25] Sun Microsystems. Conozca el nuevo netbeans. http://www.sun.com/emrkt/innercircle/newsletter/latam/0207latam_feature.html.

[26] Sun Microsystems. Getting started with an integrated development environment (ide). http://java.sun.com/developer/technicalArticles/tools/intro.html.

[27] Sun Microsystems. About the jfc and swing. http://java.sun.com/docs/books/tutorial/uiswing/start/about.html, 2007.

[28] Stefan Biffl; Richard Mordinyi. A model-drive architecture approach using explicit stakeholder quality requirement models for building dependable information systems. *IEEE Computer Society*, Mayo 2007.

[29] Inc OGIS Project Technical Commitee of the OpenGIS Consortium. The opengis guide. Kurt Buehler and Lance McKee, Mayo 1996.

[30] Claudia Bauzer Medeiros; Fatima Pires. Databases for gis. *ACM Press*, 23(1):107–115, Marzo 1994.

[31] Javier Quiroz. El modelo relacional de bases de datos, 2003.

[32] Refractions Research. Postgis. http://postgis.refractions.net/, 2005.

[33] Rubén Isaí Rivera Rodríguez. Meta-x: Sistema para creación de metadatos. Master's thesis, CINVESTAV, 2005.

[34] Noé Sierra Romero. Hevicop herramienta visual para la construcción de programas. Master's thesis, CINVESTAV, México D.F, Julio 1995. Departamento de Ingeniería Eléctrica Sección de Computación.

[35] Miguel Jesús Torres Ruiz. *Representación ontólogica basada en descriptores semánticos aplicada a objetos geográficos*. PhD thesis, Centro de Investgación en Computación, 2007.

[36] Douglas C. Schmidt. Model-driven engineering. *IEEE Computer Society*, pages 25–31, Febrero 2006.

[37] José U. Cruz Cedillo Sergio V. Chapa Vergara. Vinculación academiaindustria en el área de computación, Julio 1996.

[38] Ian Sommerville. *Ingeniería de software*. Addison Wesley, Naucalpan de Juaréz, Edo. de México, sexta edition, 2002.

[39] Kurt Stirewalt Spencer Rugaber. Model-driven reverse engineering. *IEEE Software*, pages 45–53, 2004.

[40] Raúl Hernández Stefanoni. Sistema para diseño de base de datos eve. Master's thesis, CINVESTAV, México, D.F, Marzo 1994. Departamento de Ingeniería Eléctrica Sección de Computación.

[41] Rolando Quintero Tellez. *Representación semántica de datos espaciales raster*. PhD thesis, Centro de Investigación en Computación, 2007.

[42] Bernhard Thalheim. *Entity-Relationship Modeling*. Springer, 2000.

[43] Sergio V. Chapa Vergara. Sistema de información geográfico para el desarrollo sustentable e hidrología. Technical report, CINVESTAV, Distrito Federal, Abril 2007.

[44] Sergio Víctor Chapa Vergara. *Programación Automática a partir de Descriptores de Flujo de Información*. PhD thesis, CINVESTAV, México, D.F, Marzo 1991. Departamento de Ingeniería Eléctrica Sección de Computación.

[45] Sergio V. Chapa Vergara y Noé Sierra Romero. Diseño de base de datos asistido por computadora. Technical report, CINVESTAV, México D.F, Abril 2007.

[46] Alonso Rodríguez Zamora. *Publicación en Internet y tecnología XML*. RA-MA, Madrid España, 2004.

[47] Xuetong Xie Zhanli Wang, Yu Fang. A spatio-temporal data model based on the parcel in cadastral. *IEEE*, pages 951–954, 2005.