UNIVERSIDAD CATÓLICA ANDRÉS BELLO

FACULTAD DE INGENIERÍA

ESCUELA DE INGENIERÍA INFORMÁTICA

DISEÑO E IMPLANTACIÓN DE UNA HERRAMIENTA PARA MANEJO DE MODELOS DE DATOS BASADA EN UN METAMODELO INDEPENDIENTE DE LA PLATAFORMA DESARROLLO DE APLICACIONES.

Este Jurado; una vez realizado el examen del presente trabajo ha evaluado su contenido con el resultado: 20 puntos. Mención Publicación.

JURADO EXAMINADOR

Firma:	Firma:	Firma:
Nombre: Carlos Gonzáles.	Nombre: Susana García	Nombre: Lúcia Cardoso.

REALIZADO POR	Pinto Suárez, María Alejandra. Martinez Marcano, Luis Santiago.
PROFESOR GUIA	Cardoso Militao, Lúcia.
FECHA	Caracas, Mayo de 2005.

ÍNDICE GENERAL

DEDICATORIA i
AGRADECIMIENTOS iii
ÍNDICE GENERAL vii
ÍNDICE DE TABLAS ix
ÍNDICE DE FIGURAS ix
SINOPSIS x
INTRODUCCIÓN 1
CAPÍTULO I. PROBLEMA 2
I.1. Planteamiento del Problema 2
I.2. Objetivo General 2
I.3. Objetivos Específicos 3
I.4. Justificación 3
I.5. Limitaciones y Alcances 4
I.5.1. Limitaciones 4
I.5.2. Alcance 5
CAPÍTULO II. MARCO TEÓRICO 6
II.1. Ingeniería del Software 6
II.2. CASE 10
II.2.1.Herramientas CASE 12
II.2.1.1. Clasificación 13
II.2.1.2. Componentes y funcionalidades 16
II.2.1.3. Diccionario de datos 18
II.2.1.4. Metamodelos 19
II.2.1.5. Comparación de herramientas CASE. 20
CAPÍTULO III. MARCO METODOLÓGICO 23
III.1. Modelo de Cascada. 23
III.1.1. Investigación Preliminar. 24
III.1.2. Adaptación de requerimientos. 24
III.1.3. Diseño global. 25
III.1.4. Diseño detallado 25
III.1.5.Codificación. 25
III.1.6.Pruebas. 26
CAPÍTULO IV. DESARROLLO 27
IV.1. Investigación preliminar 27
IV.2. Adaptación de requerimientos 28
IV.3. Diseño Global 32
IV.3.1. Arquitectura del sistema 34
IV.3.1.1. Capa de acceso a datos 34
IV.3.1.2. Capa lógica del negocio 34

IV.3.1.3. Capa de presentación 35
IV.4. Diseño Detallado 35
IV.4.1. Descomposición de la arquitectura del sistema. 40
IV.4.1.1. Capa de acceso a datos. 41
IV.4.1.2. Capa lógica del negocio. 52
IV.4.1.3. Capa de presentación. 60
IV.5. Codificación y depuración. 62
IV.5.1. Estructura de paquetes de la herramienta. 63
IV.5.2. Librerías utilizadas para la implementación de los componentes de la arquitectura. 64
IV.5.3. Codificación del API (Application Program Interface) para manipulación del metamodelo. 64
IV.6. Pruebas. 65
IV.6.1.Pruebas de caja negra. 66
IV.6.2.Pruebas con usuarios. 67
CAPÍTULO V. RESULTADOS 69
CONCLUSIONES 72
RECOMENDACIONES 74
BIBLIOGRAFÍA 76
GLOSARIO 78
APÉNDICE A. ENCUESTA REALIZADAS PARA LA CAPTURA DE REQUISITOS DEL SISTEMA 81
APÉNDICE B. TRADUCTORES E INTÉRPRETES. 96
APÉNDICE C. UML 101
APÉNDICE D. CASOS DE USO 114
APÉNDICE E. DIAGRAMAS DE ACTIVIDADES 137
APÉNDICE F. DIAGRAMAS DE SUCUENCIA 149
APÉNDICE G. XML & DOM 163
APÉNDICE H. ESQUEMAS XML DE ARCHIVOS DE ALMACENAMIENTO DE LA HERRAMIENTA. 169
APÉNDICE I. HERRAMIENTAS DE PROGRAMACIÓN. 177
APÉNDICE J. ESTÁNDARES DE CODIFICACIÓN UTILIZADOS PARA LA IMPLEMENTACIÓN DE LA HERRAMIENTA. 183
APÉNDICE K. GRAMÁTICA PARA LOS PROCESOS DE INGENIERÍA REVERSA 187
APENDICE L.VERSIONES DE LOS MODELOS 190
APÉNDICE M.PANTALLAS 192
APÉNDICE N. ESTRUCTURA DE REPORTES DE DICCIONARIO DE DATOS GENERADOS POR LA HERRAMIENTA 204
APÉNDICE Ñ. API DE LA HERRAMIENTA. 207
APÉNDICE O. DICCIONARIO DE DATOS 254
APÉNDICE P.PRUEBAS 313
APÉNDICE Q. TECNOLOGÍAS UTILIZADAS PARA LA REALIZACIÓN DE LA INTERFAZ. 335

ÍNDICE DE TABLAS

Tabla 1:.Ventajas y Desventajas de las Herramientas Case. 15
Tabla 2. Cuadro comparativo entre herramientas del mercado. 21
Tabla 3. Funciones de almacenamiento. 29
Tabla 4. Funciones de eliminación. 29
Tabla 5. Funciones de recuperación. 30
Tabla 6. Funciones de modificación. 30
Tabla 7. Funciones de generación. 30
Tabla 8. Ambientes de desarrollo. 63

ÍNDICE DE FIGURAS

Figura 1: Capas de la Ingeniería del Software. 9
Figura 2: Bloques de construcción CASE. 11
Figura 3. Modelo cascada con fases solapadas. 24
Figura 4. Diagrama de casos de usos del sistema (Opciones de inicio). 31
Figura 5. Diagrama de casos de usos del sistema (Opciones de uso general). 31
Figura 6. Diagrama de casos de usos del sistema(Opciones para elaboración de modelos). 31
Figura 7. Diagrama de casos de usos del sistema (Opciones de generación). 32
Figura 8. Diagrama de actividades (Crear entidad). 33
Figura 9. Arquitectura del sistema. 34
Figura 10. Diagrama de clases (Paquete conceptual). 39
Figura 11. Diagrama de clases (Paquete lógico). 39
Figura 12. Diagrama de clases (Paquete de intérpretes). 39
Figura 13. Diagrama de secuencia (Crear Entidad). 40
Figura 14 Arquitectura detallada del sistema. 41
Figura 15. Metamodelo de datos conceptual. 42
Figura 16. Metamodelo de datos lógico. 44
Figura 17. Sistema de archivos de la Herramienta. 46
Figura 18. Esquema general de un documento XML. 47
Figura 19. Estrategia de almacenamiento. 48
Figura 20. Estrategia de recuperación. 49
Figura 21. Esquema general del archivo de configuración. 50
Figura 22. Fragmento del archivo de sintaxis de ORACLE. 51
Figura 23. Estructura de almacenamiento de los espacios de trabajo de usuarios. 51
Figura 24. Proceso de interpretación y creación en memoria de árbol DOM. 53
Figura 25. Interpretes implementados en la herramienta. 53
Figura 26. Proceso de validación de acceso a la herramienta. 55
Figura 27. Manejo de versiones. 56
Figura 28. Traducción del modelo relacional al modelo físico. 58
Figura 29. Pantalla principal de la Herramienta. 61
Figura 30. Pantalla para crear atributos a una entidad. 61
Figura 31. Estructura de paquetes de la herramienta. 63

SINOPSIS

El presente Trabajo Especial de Grado, titulado **"Diseño e implementación de una herramienta para el manejo de modelos de datos basada en un metamodelo independiente de la plataforma de desarrollo de aplicaciones"**, consistió en diseñar e implementar una herramienta, con la cual, el usuario pueda construir modelos de datos, de una manera sencilla y sin la necesidad de poseer conocimientos en la utilización de este tipo de sistemas. A través de esta herramienta, el usuario, podrá realizar las actividades propias de cada una de las etapas de la construcción de una base de datos (conceptual, lógica, física), así como también realizar consultas y reportes de los modelos realizados.

La metodología utilizada para cumplir con los objetivos del desarrollo, es cascada con fases solapadas, la cual consiste en una serie de fases que constituyen cada una de las etapas del Ciclo de Vida de Desarrollo del Software, desde la investigación preliminar hasta las pruebas del sistema. La implementación del sistema fue realizada principalmente a través de los lenguajes Java y Javacc.

Como resultado de la implementación, se obtuvo una herramienta portable e independiente del sistema manejador de base de datos, la cual posee una arquitectura a tres capas, en las que se distinguen: capa de presentación, encargada de la interacción entre la herramienta y el usuario; la capa lógica, en la que se llevan a cada uno de los procesos de transformación y adaptación entre las capas, y la capa de acceso a datos, la cual se encuentra soportada por un metamodelo de datos.

INTRODUCCIÓN

En la actualidad existe un número considerable de herramientas que apoyan el diseño y desarrollo de aplicaciones, las cuales poseen ciertas características que las identifican de acuerdo a su objetivo o metodología utilizada. Dentro de estas utilidades se encuentran las herramientas CASE.

Una herramienta CASE es un conjunto de programas y ayudas que dan asistencia a los usuarios durante todos o algunos de los pasos del Ciclo de Vida de Desarrollo de Software. Generalmente dichos sistemas están dirigidos a personas con conocimientos avanzados en el área de aplicación, por lo que en ocasiones, resulta difícil su utilización por personas con nociones limitadas.

Basándose en la idea anteriormente expuesta, la meta de este Trabajo Especial de Grado fué obtener una herramienta CASE, para el diseño e implantación de modelos de datos, que sea fácil de utilizar y que logre asistir a las personas que no poseen ningún tipo de experiencia previa en la utilización de este tipo de sistemas. Este documento pretende describir el proceso el desarrollo del presente Trabajo Especial de Grado, siguiendo la siguiente estructura:

En el Capítulo I, se presenta el planteamiento del problema de una forma clara y precisa, así como también los objetivos, justificación, limitaciones y alcances del sistema. Posteriormente en el Capítulo II se exponen las definiciones teóricas requeridas, para la comprensión de las bases sobre las que se apoya el sistema. El Capítulo III, expone la metodología empleada durante el proceso de desarrollo del Trabajo Especial de Grado. Seguidamente se presenta el Capítulo IV, que detalla cómo se alcanzaron los productos resultantes en cada etapa de la metodología. En el Capítulo V, se presentan qué productos se obtuvieron como resultado en cada una de las etapas. Y por último, se muestran las conclusiones, recomendaciones, bibliografía de los libros consultados y apéndices.

CAPÍTULO I. PROBLEMA

I.1. Planteamiento del Problema

En el mundo de la Ingeniería del Software, existe una gran cantidad de herramientas que facilitan a los usuarios el diseño y desarrollo de aplicaciones. A pesar del número de utilidades existentes, algunos usuarios no las adoptan; bien sea por su complejidad o simplemente porque proveen funciones que no se adaptan a sus requerimientos.

Por lo general este tipo de instrumentos, están dirigidos hacia una población de usuarios que han adquirido conocimientos poco triviales en el diseño y desarrollo de aplicaciones. Esto conlleva a que aquellos usuarios, con poca experiencia en el área, se vean menos beneficiados en su utilización, ya que carecen del dominio de ciertos aspectos que en muchos casos resultan ser avanzados.

Si se logra construir una herramienta cuya complejidad esté orientada hacia aquellas personas que no posean conocimientos previos en la utilización de este tipo de sistemas, se podrá disminuir la posibilidad de cometer errores producto de la inexperiencia y por tanto, se podrá captar la atención de este tipo de usuarios.

De esta manera, el presente proyecto pretende desarrollar una herramienta case dedicada a la implementación de modelos de datos, cuyo objetivo fundamental esté basado en proveer una serie de funcionalidades que le permitan al usuario realizar sus tareas de una forma intuitiva y/o asistida.

I.2. Objetivo General

Diseñar e implantar un metamodelo de datos que permita la abstracción de modelos físicos a desarrolladores de manera escalable, flexible e independiente de la plataforma de desarrollo.

I.3. Objetivos Específicos

- Permitir la flexibilidad de crear y alterar el modelo de datos que se desea implantar, sin necesidad de poseer conocimientos previos de lenguajes de manipulación de datos (SQL).

- Proveer el manejo de múltiples versiones de modelos activos y la compatibilidad entre ellas.

- Proveer el manejo de seguridad de datos, en cuanto a la creación de perfiles de usuario y administración de estos en cada perfil.

- Permitir la manipulación del metamodelo de datos mediante un API (Aplication Programming Interface).

- Definir un lenguaje de recuperación de datos basado en lenguajes extensibles existentes.

- Desarrollar una herramienta que cumpla con todo lo descrito anteriormente.

I.4. Justificación

La idea de diseñar y desarrollar una herramienta case para implementar modelos de datos, no es un pensamiento que conlleve un aporte innovador en el mundo de la Ingeniería del Software; sin embargo, se puede esperar que dicho sistema represente una clara oportunidad para que usuarios inexpertos en el uso de este tipo de utilidades perfeccionen sus habilidades y optimicen el tiempo para el diseño y desarrollo de modelos de datos.

De esta forma, el poseer un sistema que asista al usuario en cada una de las etapas de la creación de su modelo, aminoraría los riesgos de fallas y por lo tanto garantizaría la

obtención de un modelo de datos cuya calidad esté a tono con los resultados esperados. Más aún, el proveer la posibilidad de crear versiones de los modelos elaborados por el usuario, avalaría el hecho de poder recuperar la información en caso de que ocurriese algún tipo de falla a nivel conceptual y/o lógico, por parte del mismo.

Por otra parte, y basándose en la orientación de la herramienta hacia usuarios con pocos conocimientos en el manejo de este tipo de utilidades, la creación de un lenguaje de recuperación propio del sistema, representaría un aporte importante, ya que lograría solventar las deficiencias de conocimientos con respecto a los lenguajes implementados por los manejadores de base de datos en los que se apoyan las herramientas case convencionales.

Apoyándose en lo descrito anteriormente, el presente trabajo especial de grado, persigue lograr beneficiar a aquellas personas que no posean conocimientos avanzados en el uso de este tipo de sistemas, facilitándoles una herramienta cuyo uso sea sencillo, intuitivo y tolerante a fallas.

I.5. Limitaciones y Alcances

I.5.1. Limitaciones

- La herramienta está dirigida a usuarios con conocimientos limitados en el diseño y análisis de sistemas.
- El paradigma de programación es el de orientación a objetos.
- La herramienta se clasifica dentro de las herramientas Toolkit de acuerdo a su amplitud y Middlecase de acuerdo al ciclo de vida que automatiza, cubriendo las actividades referidas al análisis, diseño e implantación de datos.

I.5.2. Alcance

- Definición de un Metamodelo de datos.
- Realización de una herramienta para definir modelos de datos basados en el Metamodelo anterior.
- Construcción de un API que incluya la manipulación y recuperación de datos.
- Inclusión de mecanismos de seguridad a la herramienta.
- Definición de un lenguaje de recuperación de datos.

CAPÍTULO II. MARCO TEÓRICO

II.1. Ingeniería del Software

Encontrar una definición única con respecto al término "Ingeniería del Software", es algo que puede resultar un tanto difícil. Según Bohem (1976), la Ingeniería del Software es la aplicación práctica del conocimiento científico en el diseño y construcción de programas de computadora y la documentación requerida para desarrollar, operar y mantener dichas aplicaciones. Zelkovitz (1978), por su parte, concibe este término como el estudio de los principios y metodologías, para el desarrollo y mantenimiento de sistemas.

Una terminología, un poco más completa, es la desarrollada por la IEEE [1] (1993), en la cual se describe a esta filosofía como la aplicación de un enfoque sistémico, disciplinado y cuantificable hacia el desarrollo, operación y mantenimiento del software.

Basándose en esta última definición, es oportuno realizar una descripción de los objetivos del enfoque de sistemas[2]. Ludwig von Bertalanffy, propone este enfoque para abordar el problema de la complejidad de los sistemas, a través de una forma de pensamiento basada en la totalidad y sus propiedades. (Daedalus 2005).

Existen diversos tipos de sistemas los cuales poseen ciertas características y objetivos específicos que los identifican. Dentro de estos, se encuentran los sistemas de información; los cuales son considerados como un conjunto de componentes interrelacionados que recuperan, procesan, almacenan y distribuyen información para soportar la toma de decisiones, la coordinación y el control de una organización (Universidad Nacional de Colombia 2004).

---

[1] Institute of Electrical and Electronics Engineers.

[2] Colección de componentes interrelacionados que trabajan conjuntamente para cumplir algún objetivo.

Un sistema de información está conformado por los componentes que se muestran a continuación (Marqués, María M. 2001):

- La base de datos.
- El SGBD[3].
- Los programas de aplicación.
- Los dispositivos físicos.
- El personal que utiliza y desarrolla el sistema.

La base de datos es un componente fundamental en este tipo sistemas, ya que el ciclo de vida de un sistema de información está ligado al ciclo de vida del sistema de base de datos sobre el que se apoya; a éste también se le denomina *ciclo de vida de desarrollo del software*. Las etapas típicas del ciclo de vida de una base de datos son (Marqués, María M. 2001):

- **Planificación:** el objetivo de esta etapa es encontrar una manera eficiente de llevar a cabo todas las demás etapas del Ciclo de Vida. Como apoyo a esta fase, se utilizan los modelos de datos los cuales generalmente se representan a través de diagramas entidad – relación.

  La planificación de la base de datos también incluye el desarrollo de estándares que especifiquen cómo realizar la recolección de datos, cómo especificar su formato, qué documentación será necesaria y cómo se va a llevar a cabo el diseño y la implementación.

- **Recolección y análisis de requisitos:** en esta etapa se obtienen y analizan los requerimientos del usuario y de las áreas de aplicación. El resultado de esta fase es un conjunto de documentos con las especificaciones de los requisitos del usuario, la documentación utilizada, las transacciones requeridas, etc.

[3] Sistema de Gestirón de Base de Datos.

- **Diseño de la base de datos:** está etapa consta de tres fases, las cuales se describen a continuación:

    1. Diseño conceptual: en esta fase se debe construir un esquema con la información que se utiliza en la organización, independientemente de cualquier consideración física. El esquema conceptual se construye utilizando la información que se encuentra en la especificación de los requisitos de usuario ya que representa una fuente de información para el diseño lógico de la base de datos.
    2. Diseño lógico: es el proceso de construir un esquema de la información que se utiliza, basándose en un modelo de base de datos específico e independiente del SGBD. En esta fase se transforma el esquema conceptual en un esquema lógico que utilizará las estructuras de datos del modelo de base de datos en el que se basa el SGBD que se vaya a utilizar.
    3. Diseño físico: es el proceso de producir la descripción de la implementación de la base de datos en memoria secundaria: estructuras de almacenamiento y métodos de acceso que garanticen un acceso eficiente a los datos. En general, el propósito del diseño físico es describir cómo se va a implementar físicamente el esquema lógico obtenido en la fase anterior.

- **Implementación:** en esta etapa se crean las definiciones de la base de datos a nivel conceptual, externo e interno, así como los programas de aplicación. La implementación de la base de datos se realiza mediante sentencias del lenguaje de definición de datos del SGBD, las cuales se encargan de crear el esquema de la base de datos, los archivos donde se almacenarán los datos y las vistas de los usuarios.

- **Prueba:** se encarga de probar y validar el sistema con los requisitos especificados por los usuarios. Para ello se debe diseñar una serie de casos con datos reales, que se deben llevar a cabo de manera metódica y rigurosa.

- **Mantenimiento:** en ésta, se realizan tareas como: monitorización, mantenimiento y actualización del sistema.

Estas etapas no son estrictamente secuenciales, ya que generalmente se repiten algunas fases, generando ciclos de retroalimentación. La ingeniería del software se apoya en estas, para poder cumplir con el objetivo de crear un sistema de calidad[4].

Por otra parte es preciso describir las capas en las que se fundamenta la Ingeniería del Software, ya que estas van de la mano con las etapas del ciclo de vida para poder cumplir con el objetivo mencionado anteriormente. Según Pressman (2002), la Ingeniería del Software se apoya en las siguientes capas (Figura 1):

Figura 1: Capas de la Ingeniería del Software.
Fuente: Modificada por los autores. Adaptada de (Pressman 2002).

- **Enfoque de calidad**: sobre esta capa se debe apoyar cualquier enfoque de ingeniería, ya que se debe poseer un compromiso de organización de calidad.
- **Proceso**: permite un desarrollo racional y oportuno de la ingeniería del software. Define un marco de trabajo para un conjunto de áreas claves de proceso que se deben establecer para la entrega efectiva de la tecnología.
- **Métodos**: indican como construir técnicamente el software, ya que dependen de un conjunto de principios básicos que gobiernan cada área de la tecnología e incluyen actividades de modelado y otras técnicas descriptivas.

---

[4] Conjunto de propiedades y de características de un producto o servicio, que le confieren aptitud para satisfacer una necesidad explícita o implícita.

- **Herramientas**: proporcionan un enfoque automático o semi-automático para el proceso y para los métodos.

Existen diversos tipos de herramientas las cuales se apoyan en las capas descritas anteriormente. Cuando estas se integran, para que la información creada por una herramienta pueda ser utilizada por otra, se establece un sistema de soporte para desarrollo de software llamado *ingeniería del software asistida por computadora (CASE).*

II.2. CASE

Ingeniería del software asistida por computadora (CASE), es una filosofía que se orienta a la comprensión de los modelos de la empresa y sus actividades. La tecnología CASE, supone la automatización del desarrollo del software, contribuyendo a mejorar la calidad y la productividad de los sistemas de información (Kendall & Kendall 1997).

Esta filosofía plantea los siguientes objetivos:

- Aumentar la productividad de las áreas de desarrollo y mantenimiento de los sistemas informáticos.
- Mejorar la calidad del software desarrollado.
- Reducir los tiempos de desarrollo y mantenimiento del software.
- Mejorar la gestión y dominio sobre el proyecto en cuanto a planificación, ejecución y control.
- Automatizar el desarrollo del software, la documentación, la generación de código, el chequeo de errores y la gestión del proyecto.
- Permitir la reutilización y portabilidad del software, además de la estandarización de la documentación.
- Integrar las fases de desarrollo.

En la búsqueda de los objetivos expuestos anteriormente, se proponen los bloques de construcción CASE (Figura 2). Dichos bloques, surgen debido a que la ingeniería

asistida por computadora puede ser tan sencilla como una herramienta que preste apoyo a una actividad específica, o tan compleja como todo un entorno que abarque herramientas, personas, sistemas operativos, etc. (Pressman 2002).

Figura 2: Bloques de construcción CASE.
Fuente: Modificada por los autores. Adaptada de (Pressman 2002).

- **Arquitectura de entorno**: establece los cimientos para un entorno CASE, consta de un *entorno de hardware* y de un soporte de *sistema operativo.*
- **Servicios de portabilidad**: proporcionan un puente entre las herramientas CASE, su *marco de integración* y la *arquitectura del entorno.* Permiten que las herramientas CASE y su marco de integración migren entre distintas plataformas del hardware y sistemas operativos sin un mantenimiento adaptativo significativo.
- **Marco de integración**: es un grupo de programas especializados que permiten a cada una de las herramientas comunicarse entre si, para crear una base de datos del proyecto, y mostrar el mismo aspecto al usuario final.

Los bloques de construcción representados en la figura 2, constituyen un fundamento completo para la integración de herramientas CASE. Sin embargo, de acuerdo a los objetivos para la cual este diseñada la herramienta, puede ser construida empleando algunos de los bloques anteriormente descritos.

Es importante destacar, que la arquitectura sobre la cual este desarrollada la herramienta, representa un factor fundamental en la integración de los bloques de construcción CASE, por ello existen diversos tipos de arquicteturas, las cuales difieren unas de otras de acuerdo a su distribución a nivel de implementación.

La arquitectura a tres capas, plantea la división de la funcionalidad de la herramienta de acuerdo a las capas siguientes: la capa de presentación, la capa lógica y la capa de datos.

- La capa de presentación: proporciona la interfaz mediante la cual se muestra información y se obtienen los datos introducidos por los usuarios a través de las ventanas y/o formularios.
- La capa lógica: representa el puente entre el usuario y los servicios de datos. Responde a peticiones para ejecutar una tarea en específico, aplicando procedimientos formales y reglas de negocio a los datos relevantes.
- La capa de datos: en esta capa se encuentran toda la información necesaria para el funcionamiento de la herramienta, por tanto esta se encuentra almacenada en la estructuras necesarias de acuerdo a la naturaleza de la aplicación.

II.2.1.Herramientas CASE

Según un documento publicado por el Instituto Nacional de Estadística e Informática (1999), se puede definir a las herramientas CASE, como un conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del Ciclo de Vida de Desarrollo de Software.

Al adoptar este tipo de utilidades se obtiene como ventaja principal, una mejora considerable en la calidad de los desarrollos realizados y un aumento de la productividad. Sin embargo, cabe destacar, que para conseguir estos objetivos es conveniente contar

con una organización y una metodología de trabajo, que sirvan de apoyo a la herramienta.

La optimización de la calidad, se consigue reduciendo sustancialmente los problemas de análisis y diseño, inherentes a los proyectos de mediano y gran tamaño; mientras que el aumento de la productividad surge como consecuencia de la automatización de determinadas tareas, como la generación de código y la reutilización de los objetos o módulos.

En la actualidad existen una gran cantidad de herramientas, las cuales poseen funcionalidades y objetivos específicos, por ello se ha creado un sistema de clasificación que las agrupa de acuerdo a ciertos criterios.

II.2.1.1. Clasificación

Existen diversos enfoques para la clasificación de las herramientas CASE, entre los que se pueden mencionar los siguientes (Instituto Nacional de Estadística e Informática 1999):

- De acuerdo a las plataformas que soportan.
- De acuerdo a las fases del ciclo de vida que cubren.
- Por la arquitectura de las aplicaciones que producen.
- Por su funcionalidad.

Apoyándose en los enfoques citados anteriormente, se pueden distinguir las clasificaciones que se muestran a continuación:

- **De acuerdo a su amplitud:**

  - **TOOLKIT:** es una colección de herramientas integradas que permiten automatizar un conjunto de tareas de alguna de las fases del ciclo de de vida del sistema informático.
  - **WORKBENCH:** son conjuntos de herramientas que dan soporte a la automatización del proceso completo de desarrollo del sistema informático. El proceso final aportado por estas, es un sistema en código ejecutable, con su respectiva documentación.

- **De acuerdo a las fases del Ciclo de Vida que automatizan:**

  - **Herramientas integradas, *I-CASE*:** abarcan todas las fases del Ciclo de Vida del Desarrollo de Sistemas. Se conciben como el conjunto de cuatro herramientas que abarcan las disciplinas que van desde la estrategia de la empresa, hasta el análisis, diseño e implementación de los programas. Se basan en una metodología y aportan técnicas estructuradas para todas las fases del Ciclo de Vida.
  - **Herramientas de alto nivel, *U-CASE*:** están orientadas a la automatización y soporte de las actividades desarrolladas durante las primeras fases del desarrollo (análisis y diseño).
  - **Herramientas de bajo nivel, *L-CASE*:** están dirigidas a las últimas fases del desarrollo (generación de código e implantación).
  - ***Tools CASE*:** es el tipo más simple de herramientas CASE, ya que automatiza solo una fase dentro del Ciclo de Vida.

A continuación se muestra una tabla comparativa, que refleja las ventajas y desventajas de las herramientas CASE, basadas en esta clasificación:

Tipo de CASE	Ventajas	Desventajas
I – Case	➢ **Integra el Ciclo de Vida.** ➢ **Permite lograr importantes mejoras de productividad a mediano plazo.** ➢ **Permite un eficiente soporte al mantenimiento de sistemas.** ➢ **Mantiene la consistencia de los sistemas a nivel corporativo.**	➢ **No es tan eficiente para soluciones simples, sino para soluciones complejas.** ➢ **Depende del hardware y del software.** ➢ **Es costoso.**
Upper Case	➢ **Se utiliza en plataforma PC, es aplicable a diferentes entornos.** ➢ **Menos costos.**	➢ **Permite mejorar la calidad de los sistemas, pero no mejora la productividad.** ➢ **No permite la integración del Ciclo de Vida.**
Lower Case	➢ **Permite lograr importantes mejoras de productividad a corto plazo.** ➢ **Permite un eficiente soporte al mantenimiento de sistemas.**	➢ **No garantiza la consistencia de los resultados a nivel corporativo.** ➢ **No garantiza la eficiencia del Análisis y Diseño.** ➢ **No permite la integración del ciclo de vida.**

Tabla 1:.Ventajas y Desventajas de las Herramientas Case.
Fuente: Modificada por los autores.
Adaptada de Instituto nacional de estadística e informática 1999.

- **De acuerdo a su funcionalidad:**

  - **Herramientas de planificación de sistemas de gestión:** su principal objetivo es modelar los requisitos de información estratégica de una organización. Proporcionan un "metamodelo" que sirve de base para obtener sistemas de información específicos, por lo que son de gran ayuda cuando se diseñan nuevas estrategias para los sistemas de información.
  - **Herramientas de Análisis y Diseño:** permiten crear un modelo del sistema que se va a construir, para poder evaluar su validez y consistencia. Este tipo de herramientas proporcionan un grado de confianza y disminuyen los errores. Entre estas se pueden encontrar: las herramientas de modelado, las de creación de prototipos y simulación, y las de diseño y elaboración de interfaces.

- **Herramientas de programación:** dentro de este tipo de herramientas se pueden identificar las herramientas de codificación convencionales, las de cuarta generación y las de programación orientada a objetos.
- **Herramientas de integración y prueba:** representan una ayuda para la medición, simulación y prueba de los equipos lógicos desarrollados. Entre este tipo de herramientas se encuentran: las herramientas de análisis estático y las de generación de casos de prueba.
- **Herramientas de mantenimiento:** este tipo de herramientas se apoya principalmente en la ingeniería inversa [5]y en la reingeniería[6], para lograr la reestructuración y análisis de código.
- **Herramientas de soporte:** dentro de esta clasificación se encuentran las herramientas que recogen las actividades aplicables en todo el proceso de desarrollo, un ejemplo de ellas, son las herramientas de base de datos y control de calidad.

A pesar de la gran cantidad de herramientas que existen en el mercado y de la diversidad de funcionalidades que estas poseen, existen una serie de componentes comunes en las que se apoyan la mayoría de estas. Dichos componentes son de vital importancia, ya que proveen la base para la inclusión de otras características que complementen y optimicen la utilización de estos sistemas.

II.2.1.2. Componentes y funcionalidades

Según Instituto Nacional de Estadística e Informática (1999), las Herramientas CASE, por lo general suelen incluir los siguientes componentes:

- **Repositorio:** Es la base de datos central de una herramienta CASE. El repositorio amplía el concepto de diccionario de datos para incluir toda la

[5] Es el proceso de analizar el software con el objetivo de recuperar su diseño y especificación.
[6] Es la implementación de sistemas heredados para hacerlos más mantenibles.

información que se va generando a lo largo del Ciclo de Vida del sistema, como por ejemplo: componentes de análisis y diseño (diagramas de flujo de datos, diagramas entidad-relación, esquemas de bases de datos, diseños de pantallas), estructuras de programas, algoritmos, etc. La mayoría de las herramientas CASE poseen un repositorio propio, en caso de no poseerlo trabajan sobre un repositorio suministrado por otro fabricante o vendedor.

- **Metamodelo:** Constituye el marco para la definición de las técnicas y metodologías soportadas por la herramienta.

- **Módulos de diagramación y modelización:** algunos de los diagramas y modelos utilizados con mayor frecuencia son los siguientes:
  - Diagrama de flujo de datos.
  - Modelo entidad –relación.
  - Diagrama de estructuras de datos.
  - Diagrama de clases (UML[7]).

- **Herramienta de prototipado:** El objetivo de esta herramienta es poder mostrar al usuario, desde el diseño, el aspecto que tendrá la aplicación una vez desarrollada. Los prototipos han sido utilizados ampliamente en el desarrollo de sistemas tradicionales, ya que proporcionan una retroalimentación inmediata, que ayuda a determinar los requisitos del sistema.

- **Generador de código:** Las características más relevantes de los generadores de código son:
  - Lenguaje generado.
  - Portabilidad del código generado.
  - Generación del esqueleto del programa o del programa completo.

---

[7] Unified Modeling Language.

- Posibilidad de modificación del código.

- **Módulo generador de documentación:** Este módulo se alimenta del repositorio para transcribir las especificaciones allí contenidas. Algunas de sus principales características son:
  - Generación automática a través de los datos del repositorio.
  - Combinación de información textual y gráfica.
  - Generación de referencias cruzadas.

Es importante mencionar el hecho de que las herramientas CASE para el análisis y diseño de aplicaciones incluyen a los diccionarios de datos y a los metamodelos como pilares fundamentales para el manejo de los datos y la definición de las metodologías. Por ello es necesario profundizar a cerca de estos componentes.

II.2.1.3. Diccionario de datos

Según Lania (2000), existen dos enfoques fundamentales en los que se clasifican los diccionarios de datos: el diccionario de datos del proyecto y el diccionario de datos propio de la base de datos.

El diccionario de datos del proyecto, contiene las características lógicas de los sitios donde se almacenan los datos del sistema. Además, identifica los procesos donde se emplean los datos y los sitios donde se necesita el acceso inmediato a la información.

Existen diversas razones para su utilización, entre las cuales se pueden mencionar:

- El manejo de los detalles en sistemas de amplia envergadura.
- La asignación de un significado único a cada uno de los elementos y actividades del sistema.

- La documentación de las características del sistema, incluyendo partes y componentes.
- Facilitación en el análisis de los detalles, con la finalidad de evaluar las características y determinar donde efectuar los cambios.
- Localización de errores y omisiones en el sistema.

Por otra parte, el diccionario de datos de una base de datos, es una estructura de tablas y vistas, de sólo lectura, que contiene información propia de la base de datos, tal como:

- Las definiciones de todos los esquemas en la base de datos.
- Los usuarios de la base de datos, sus roles y privilegios.
- Información general de la base de datos.

Este diccionario se establece cuando la base de datos es creada, para reflejar con exactitud el estado de la misma y es actualizado automáticamente en respuestas de acciones específicas, tales como: la modificación de la estructura de la base de datos.

II.2.1.4. Metamodelos

Los metamodelos son una parte importante de las tecnologías de base de datos, ya que representan una herramienta que permite un nivel de abstracción mayor en la concepción de bases de datos complejas, para diversas aplicaciones que responden a las tendencias actuales de los sistemas informáticos (Rodríguez M., 2000).

Fundamentalmente esta representación constituye un nivel de abstracción mayor o metanivel, el cual permite realizar operaciones sobre los modelos, que no serían posibles sin existir una forma explícita de descripción de los mismos.

El enfoque de los metamodelos es de gran utilidad en las etapas de análisis de requerimientos y modelado conceptual de la base de datos, en sistemas de información complejos. El metamodelo pasa a ser un fundamento de las aplicaciones dedicadas a

tareas de negocio y a los sistemas basados en flujo de trabajo, ya que este representa la base para el desarrollo de herramientas de Ingeniería de Software.

En la actualidad existen una gran cantidad de herramientas en el mercado, que utilizan los diccionarios de datos como componentes para la introducción de técnicas de análisis y diseño de aplicaciones. Cada una de estas posee una serie de características que representan un valor agregado para el objetivo fundamental de las mismas.

II.2.1.5. Comparación de herramientas CASE.

Las Herramientas CASE dedicadas al análisis y diseño de aplicaciones, han tenido un auge considerable en los últimos tiempos; por ello diversas compañías las han desarrollado, con la finalidad de incluirle además de las características básicas de este tipo de aplicaciones, una serie de funcionalidades propias del producto a comercializar.

A continuación se muestra una tabla comparativa de tres de las herramientas que actualmente poseen prestigio en el mercado:

Criterios de comparación	ERwin 3.0	ER/Studio 2.5	Power Designer 6.1
Características Generales.	Es una herramienta para modelar, que ayuda a diseñar bases de datos de alto desempeño para cliente/servidor y web intranet, así como aplicaciones de data warehousing.	Es una herramienta de modelado de datos, facil de usar y multinivel, para el diseño y construcción de bases de datos, a nivel físico y lógico.	Es una herramienta para crear bases de datos y aplicaciones cliente/servidor basadas o no en web. Exporta información del modelo físico y extiende atributos al diccionario de 4GL.
Diagramas	Los diagramas de modelos de datos, se usan para generar o actualizar bases de datos relacionales.	La creación de diagramas es clara y rápida. Tiene la posibilidad de crear subvistas, para separar y manejar áreas importantes.	Cuenta con herramientas para la creación y control de diagramas, como son: Off-page Connector, Business rules y CRUD matriz.
Esquema de la Base de Datos	Para diseñar un modelo de datos, ERwin proporciona propiedades específicas de DBMS y el servidor de editores, que permiten definir nombres físicos y propiedades para las tablas, columnas y relaciones que genera cuando se crea el esquema de la base de datos.	Las capacidades de diseño que contiene, ayudan a crear un diseño lógico que puede transformarse en cualquier número de diseños físicos. Como resultado, se puede mantener un diseño lógico normalizado mientras se desnormalizan los diseños físicos para su desempeño.	Proporciona capacidades de modelado de datos tradicional, incluyendo diseño de bases de datos, generación, mantenimiento, ingeniería reversa y documentación para arquitecturas de bases de datos. El diseño se realiza en dos niveles: el nivel conceptual (entidades, relaciones, dominios, etc.) y el lógico (tablas, columnas, dominios, etc.).
Metodología	Soporta metodologías para diagramas de entidad-relación, modelos de datos IDEF1 y Yourdon/DeMarco	Soporta metodologías de Yourdon, con diagramas entidad-relación y modelos IDEF1.	Soporta modelos funcionales y notaciones de diagramas de flujo, modelo funcional de objetos (OMT), Yourdon/DeMarco, Gane&Sarson y SSADM.

Tabla 2. Cuadro comparativo entre herramientas del mercado.
Fuente: Modificada por los autores.
Adaptada de Subdirección de Sistemas-DCCA- UNAM (2004).

Como se pudo observar en la tabla anteriormente expuesta, las herramientas CASE han adquirido una serie de funcionalidades que van de la mano con las tendencias tecnológicas que existen en la actualidad; por ello es factible mencionar algunas de las líneas de la evolución hacia las que se han encaminado este tipo de sistemas:

- CASE para sistemas bajo arquitectura Cliente/Servidor.
- CASE multiplataforma.
- CASE para ingeniería inversa y directa.
- CASE para trabajo en grupo.
- CASE para desarrollo de sistemas orientados a objetos.

Quizás a mediano o largo plazo se podrá incluir la utilización de tecnología multimedia, inteligencia artificial y sistemas de realidad virtual.

CAPÍTULO III. MARCO METODOLÓGICO

III.1. Modelo de Cascada.

El modelo de cascada propuesto por Royce (1970), toma las actividades fundamentales del proceso de especificación, desarrollo, validación y evolución del software y los representa como fases separadas.

Teóricamente una fase comienza cuando la otra ha finalizado, pero debido a las implicaciones de esta rígida concepción, fue desarrollada una variación del modelo en la que se solapan las fases permitiendo la transferencia de información entre las mismas; esta variación es conocida por el nombre de "Modelo cascada con fases solapadas" (Sommerville, 2002).

El modelo de cascada con fases solapadas fue seleccionado como metodología de desarrollo, debido a que a través de este se pueden abarcar todas las etapas del ciclo de vida de software. Además, el solapamiento de las fases, provee una ventaja significativa ya que permite iterar entre las fases, admitiendo el hecho de poder tomar en cuenta aspectos que vayan surgiendo durante el avance del proyecto.

En cada una de las fases de modelo, se incluyeron técnicas propias del diseño orientado a objetos, las cuales utilizan, en su mayoría, el lenguaje UML[8] para la representación de dichas técnicas (Ver apéndice C).

A continuación, se muestra el modelo propuesto como metodología de desarrollo (Figura 3):

[8] Unified Modeling Lenguage

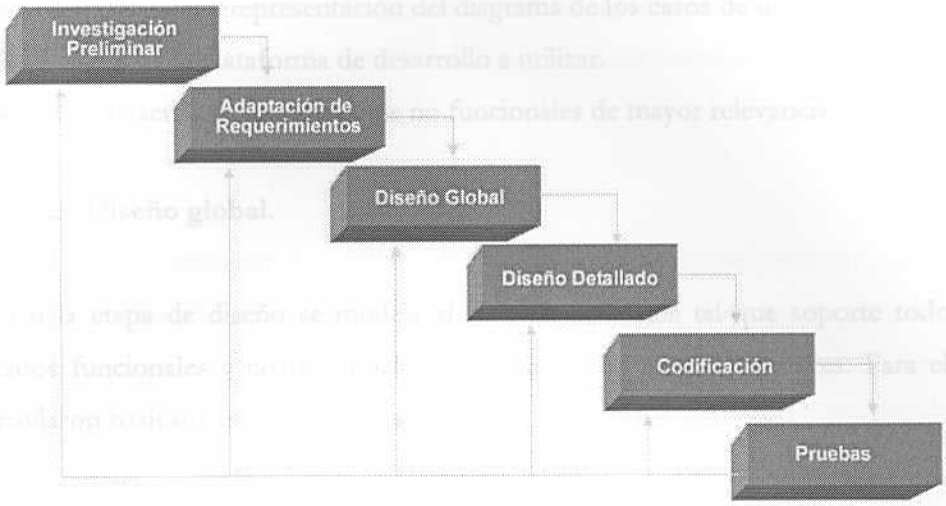

Figura 3. Modelo cascada con fases solapadas.
Modificada por los autores. Adaptada de (Sommerville 2002).

III.1.1. Investigación Preliminar.

En esta etapa se realiza todo lo referente al levantamiento de información, y el establecimiento del alcance y objetivos del proyecto. Las actividades propuestas para lograr obtener dicha información, son las siguientes:

- Entrevistas y reuniones con la tutora y profesores expertos en el área.
- Aplicación de encuestas a estudiantes.
- Revisiones bibliográficas.

III.1.2. Adaptación de requerimientos.

Esta fase utiliza la información lograda a través de la etapa anterior, como fuente para el análisis y la definición de los requerimientos funcionales y no funcionales del sistema.

Las actividades a realizar en esta etapa son las siguientes:

- Definición de las funciones básicas del sistema.

- Identificación y representación del diagrama de los casos de uso.
- Análisis de la plataforma de desarrollo a utilizar.
- Identificación de los requisitos no funcionales de mayor relevancia.

III.1.3. Diseño global.

En la etapa de diseño se modela el sistema de forma tal que soporte todos los requisitos funcionales y no funcionales identificados en etapas anteriores. Para ello se desarrollaron básicamente dos actividades:

- Elaboración de los diagramas de actividades para cada caso de uso.
- Definición de la arquitectura del sistema.

III.1.4. Diseño detallado.

A través de esta etapa se adecua el análisis a las características específicas del ambiente de Implantación y se completan las distintas aplicaciones del sistema con los modelos de control, interfaz o comunicaciones, según sea el caso.

Las actividades que se realizan en esta fase, son las siguientes:

- Descripción de los casos de uso, identificados en la etapa de adaptación de requerimientos.
- Diseño de la estructura de clases del sistema.
- Elaboración de los diagramas de secuencia.
- Descomposición en capas de la arquitectura del sistema.

III.1.5.Codificación.

En esta fase se implementa el sistema en términos de componentes, es decir, archivos de código fuente, scripts, ejecutables, etc. (Jacobson et al., 2000). El propósito

fundamental de este proceso es implementar las clases y los subsistemas encontrados durante la fase de diseño, con la finalidad de integrarlos para obtener el sistema deseado.

III.1.6.Pruebas.

Las pruebas verifican el resultado de la implementación, probando las estructuras internas y externas del sistema, con el fin identificar fallar y realizar evaluaciones que permitan verificar el cumplimiento de las expectativas y necesidades de los usuarios. Para lograr este objetivo se desarrollaron las siguientes actividades:

- Pruebas de caja negra de los módulos del sistema.
- Pruebas con usuarios.

CAPÍTULO IV. DESARROLLO

IV.1. Investigación preliminar

Esta etapa representa el inicio del ciclo de vida del presente Trabajo Especial de Grado. En ella se engloban todas aquellas actividades involucradas en los procesos inherentes al levantamiento de información y la captura de los requerimientos funcionales y no funcionales del sistema.

Debido a que el proyecto no se realizó siguiendo algún lineamiento establecido por ninguna organización o grupo de investigación específico, los requerimientos funcionales y no funcionales del sistema, se obtuvieron a través de reuniones y entrevistas con la tutora y profesores de la Escuela de Ingeniería Informática de la Universidad Católica Andrés Bello, así como también realizando actividades como encuestas e investigaciones, sobre herramientas similares.

Durante las investigaciones previas, realizadas acerca de las herramientas CASE que existen en la actualidad, se tomaron en cuenta diversos criterios, tales como:

- Funcionalidades que proveen.
- Grado de dificultad en su utilización.
- Tipo de usuarios hacia los que están dirigidas.
- Comportamiento en los procesos de desarrollo de aplicaciones.

Dichos criterios sirvieron de apoyo para el establecimiento del alcance y objetivo de la herramienta, los cuales fueron determinados durante una serie de entrevistas pautadas con la tutora, en las que además, se logró definir la funcionalidad general de la aplicación.

Analizando la visión obtenida mediante la definición del alcance y objetivo; la herramienta a desarrollar, estaría dirigida a usuarios con pocos conocimientos en el

desarrollo de aplicaciones, bajo un enfoque de sencilla usabilidad, tolerancia a fallas y asistencia al mismo durante todo el proceso de diseño de los modelos de datos.

A fin de validar las funcionalidades planteadas a través de las actividades descritas anteriormente, se realizaron encuestas en las cuales se definieron una serie de variables, las cuales fueron planteadas bajo la asesoría de estudiantes de psicología de la Universidad Católica Andrés Bello. Dichas variables son las siguientes:

- Conocimientos acerca de las herramientas CASE.
- Usos generales de las herramientas CASE.
- Características de las herramientas CASE.
- Opiniones y sugerencias.

Las preguntas formuladas fueron delimitadas en función de variables expuestas anteriormente, así como también los métodos utilizados para la tabulación de los resultados. Cabe destacar que la muestra seleccionada para la aplicación de la encuesta, estuvo compuesta por estudiantes de la Universidad Católica Andrés Bello que estaban cursando las asignaturas: "Sistema de Base de Datos II" y "Desarrollo de Software", ya que estos poseen un cierto grado de conocimiento en el manejo de este tipo de herramientas.

Los modelos de las encuestas y los resultados obtenidos a través de su aplicación se encuentran especificados en el apéndice A.

IV.2. Adaptación de requerimientos

Tomando en cuenta las funcionalidades identificadas durante la primera fase del desarrollo, se procedió a desglosar cada una de ellas, en función de obtener los requerimientos funcionales y no funcionales de la herramienta.

Debido a la naturaleza de la herramienta, las funciones del sistema están divididas en cinco grupos principales: funciones de almacenamiento (Ver Tabla 3), funciones de eliminación (Ver Tabla 4), funciones de recuperación (Ver Tabla 5), funciones de modificación (Ver Tabla 6) y funciones de generación (Ver Tabla 7). Cada una de estas funciones se clasificó según las siguientes categorías:

- **Evidente:** Debe realizarse, y el usuario debería saber que se ha realizado.
- **Oculta:** Debe realizarse, aunque no es visible para los usuarios.
- **Superflua:** Opcionales; su inclusión no repercute significativamente en el costo ni en otras funciones.

A continuación se especifican cada una de las funciones del sistema, agrupadas en tablas según el grupo al cual pertenecen.

Referencia #	Función	Categoría
R1.1	**Capturar la información introducida por el usuario.**	**Evidente**
R1.2	**Validar los datos introducidos por el usuario.**	**Evidente**
R1.3	**Almacenar los datos en las estructuras de memoria.**	**Oculta**
R1.4	**Almacenar los datos en archivos XML**	**Oculta**

Tabla 3. Funciones de almacenamiento.
Fuente: Elaborado por los autores.

Referencia #	Función	Categoría
R2.1	**Eliminar objeto de la memoria.**	**Oculta**
R2.2	**Remover el objeto seleccionado para la eliminación gráficamente.**	**Evidente**

Tabla 4. Funciones de eliminación.
Fuente: Elaborado por los autores.

Referencia #	Función	Categoría
R3.1	Recuperar el objeto de la memoria.	Oculta
R3.2	Mostrar el objeto gráficamente.	Evidente

Tabla 5. Funciones de recuperación.
Fuente: Elaborado por los autores.

Referencia #	Función	Categoría
R4.1	Modificar el objeto en memoria.	Oculta

Tabla 6. Funciones de modificación.
Fuente: Elaborado por los autores.

Referencia #	Función	Categoría
R5.1	Generar modelo lógico gráficamente.	Evidente
R5.2	Generar modelo físico.	Evidente
R5.3	Generar modelo lógico a partir de script SQL.	Evidente

Tabla 7. Funciones de generación.
Fuente: Elaborado por los autores.

El resto de los requisitos funcionales del sistema fueron capturados a través de la utilización de los casos de uso. En primer lugar se definieron los actores, los cuales no son más que terceros fuera del sistema que colaboran o interactúan con él. En este caso el único encargado del manejo y utilización de la herramienta es el usuario, por lo tanto solo fue definido un actor el cual se encarga de llevar a cabo todos y cada uno de los casos de usos que se describirán a continuación. Sin embargo vale la pena mencionar que fue contemplado un perfil adicional, el cual cumple con las funciones de administración de la herramienta, con el objetivo de que a futuro el sistema pueda ser controlado en un entorno multiusuarios.

Cabe destacar que debido al número de funcionalidades del sistema, fue necesario realizar una agrupación de las mismas, de acuerdo a ciertos criterios de similitud, lo cual

facilitó la realización del diagrama representativo de los casos de uso del sistema, que se muestran a través de las figuras (3,4,5 y6).

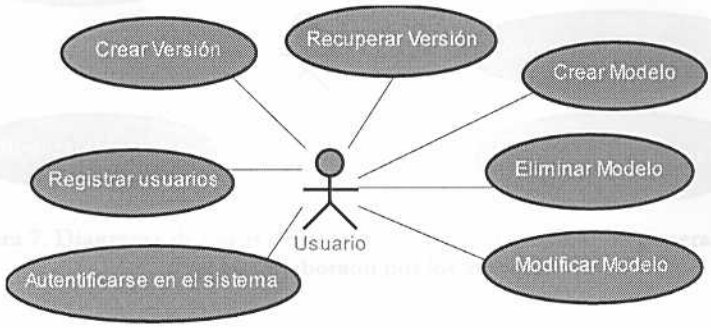

Figura 4. Diagrama de casos de usos del sistema (Opciones de inicio).
Fuente: Elaborado por los autores.

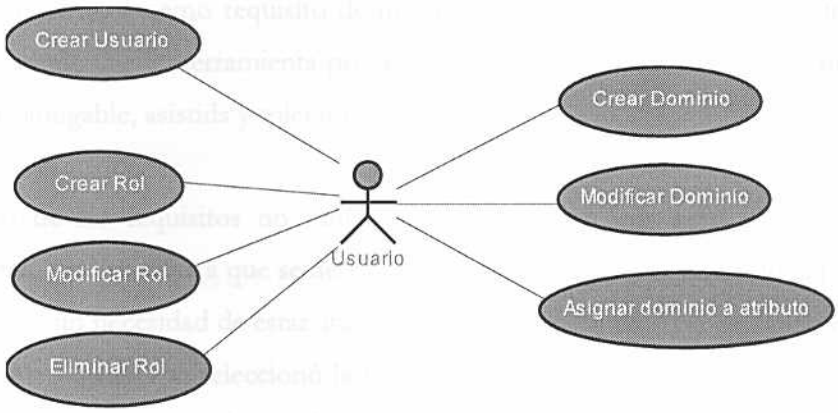

Figura 5. Diagrama de casos de usos del sistema (Opciones de uso general).
Fuente: Elaborado por los autores.

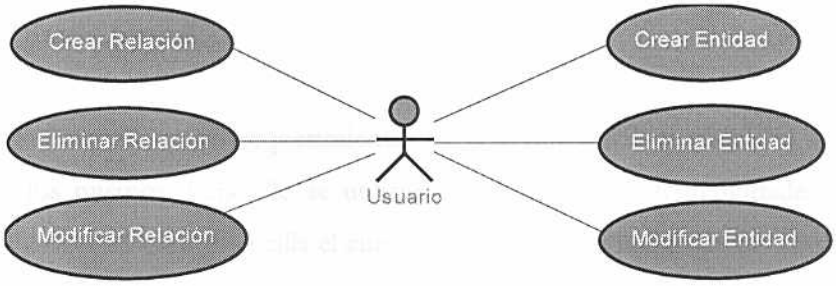

Figura 6. Diagrama de casos de usos del sistema
(Opciones para elaboración de modelos).
Fuente: Elaborado por los autores.

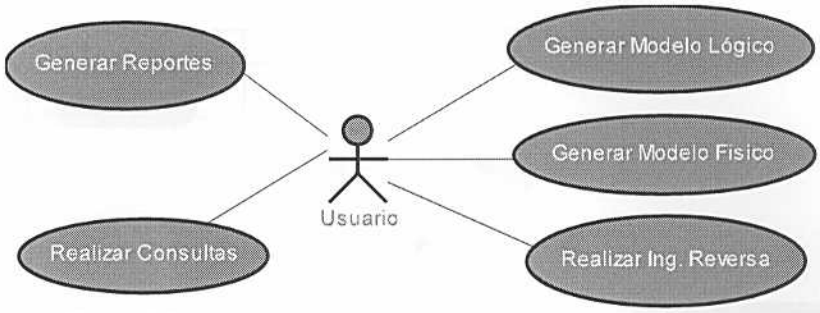

Figura 7. Diagrama de casos de usos del sistema (Opciones de generación). Fuente: Elaborado por los autores.

La mayoría de los requisitos no funcionales de la herramienta, giran en torno a la idea de que este sea dirigido a usuarios con poca experiencia en su utilización. Por lo tanto, se determinó como requisito de mayor relevancia, la presentación o interfaz, ya que se pretende que la herramienta provea al usuario la posibilidad de crear modelos de una forma amigable, asistida y tolerante a fallas.

Otro de los requisitos no funcionales que poseen una gran importancia es la portabilidad, esto se debe a que se desea que el sistema pueda ser utilizado bajo cualquier plataforma y sin necesidad de estar atado a algún manejador de base de datos. Para lograr alcanzar este objetivo se seleccionó la tecnología Java, ya que esta cuenta con una gran cantidad de componentes, los cuales cumplen con el paradigma orientado a objetos y proveen una independencia total de la plataforma de desarrollo (Ver Apéndice I).

IV.3. Diseño Global

Una vez obtenidos los requerimientos de la herramienta, se procedió a realizar un análisis de los mismos. Para ello se utilizaron los diagramas de actividades, los cuales muestran de manera clara y sencilla el curso secuencial de cada uno de los casos de uso.

La figura 8 muestra un diagrama de actividades, que corresponde a uno de los casos de usos nombrados anteriormente.

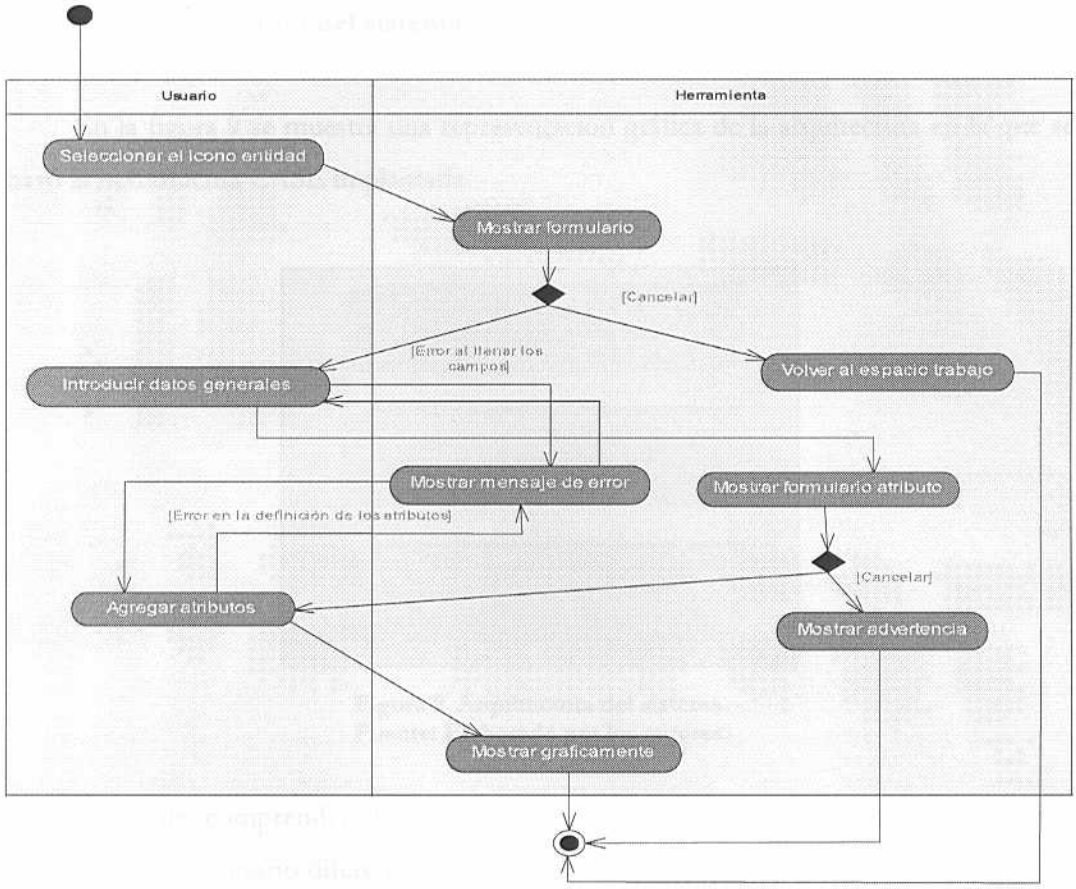

Figura 8. Diagrama de actividades (Crear entidad).
Fuente: Elaborado por los autores.

La especificación de los demás diagramas de actividades se presenta en el Apéndice E.

Una vez analizados los casos de uso junto con los diagramas de actividades, se procedió a concebir el modelo de la arquitectura que cumpliese con las especificaciones funcionales requeridas.

Considerando que el enfoque principal del sistema, establece la independencia de la plataforma de desarrollo de las aplicaciones; se diseñaron estrategias de almacenamiento y recuperación de datos, las cuales brindan la independencia y escalabilidad deseada. Basándose en lo descrito anteriormente, la arquitectura planteada, se fundamentó en un modelo de componentes a tres capas.

IV.3.1. Arquitectura del sistema

En la figura 9 se muestra una representación gráfica de la arquitectura en la que se basó la herramienta CASE implantada.

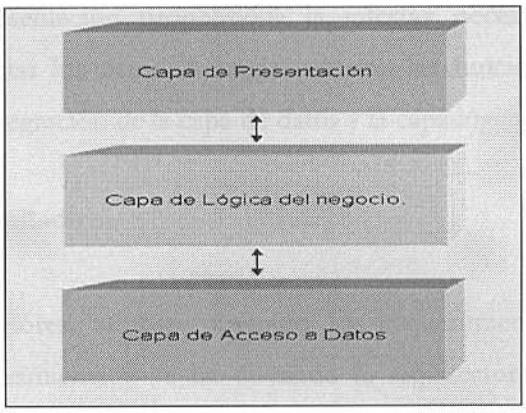

Figura 9. Arquitectura del sistema.
Fuente: Elaborado por los autores.

A fin de comprender de manera general el funcionamiento de la arquitectura implantada, es necesario diferenciarla por componentes.

IV.3.1.1. Capa de acceso a datos

Esta capa representa la base fundamental de la arquitectura, en esta se realizan todos los procesos referentes tanto al almacenamiento de los datos, como a las estrategias utilizadas para gestionar la información en la herramienta.

IV.3.1.2. Capa lógica del negocio

Representa la parte de la arquitectura en la que se estructura en módulos, aquellos procesos que utiliza la herramienta internamente para transformar la información entre las distintas fases por las que debe pasar el usuario, para lograr alcanzar la implantación final de una base de datos.

Dentro de esta capa se distinguen tres fases importantes: el diseño conceptual, el diseño lógico y el diseño físico de una base de datos (Navathe et al. 1997).

IV.3.1.3. Capa de presentación

La capa de presentación proporciona la interfaz necesaria para presentar la información y recolectar los datos. Además muestra, las funciones que se ofrecen al usuario mediante la integración de la capa de datos y la capa lógica de la arquitectura.

IV.4. Diseño Detallado

En etapas anteriores, se han obtenido los requerimientos funcionales y no funcionales de la herramienta y se ha diseñado la arquitectura bajo la cual ha sido desarrollada. En esta fase del diseño detallado, se procedió a preparar el camino para la implantación de los requerimientos obtenidos, mediante la especificación a fondo de cada uno de los componentes que conforman el sistema.

En la fase de adaptación de requerimientos se identificaron los casos de usos del sistema, sin especificar las actividades que se llevan a cabo en cada uno de estos. En esta etapa fue necesario realizar una descripción detallada de cada uno de los casos de usos, ya que a través de esta, se identificaron una serie de funcionalidades necesarias para realizar el diseño de la implementación del sistema.

A continuación se presenta en detalle uno de los casos de usos del sistema:

Caso de Uso #1:	Crear entidad
Actores:	**Usuario (iniciador).**
Propósito:	**Agregar una entidad al modelo o versión actual.**
Precondición:	**El usuario debe haber creado un modelo o estar trabajando sobre una versión existente.**

Resumen: **El usuario selecciona en la barra de herramientas, una entidad y llena los datos correspondientes.**

Referencias cruzadas: ***Funciones:*** **R1.1, R1.2, R1.3,R1.4.**

Casos de Usos: **el usuario debe haber terminado alguno de los casos de usos:** ***Crear modelo o Recuperar versión.***

Curso normal de los eventos:

Acción de los actores	Respuesta del sistema
1. El usuario escoge la opción entidad en la barra de herramientas.	**2. El sistema muestra un formulario, con campos necesarios para poder ingresar los datos generales de la entidad. Estos son:** - *Nombre:* **seudónimo con el que identifica la entidad.** - *Descripción:* **breve explicación del objetivo o la función de la entidad.**
3. El usuario introduce cada uno de los datos solicitados y presiona el botón "continuar".	**4. El sistema muestra una ventana, en donde el usuario podrá añadir atributos a la entidad. Para ello se mostrará una tabla en donde se permitirá agregar los siguientes datos de los atributos:** - Nombre: **seudónimo con el que identifica el atributo.** - Descripción: **breve explicación del objetivo o la función del atributo.** - Tipo de dato: **formato en el que**

	es almacenado, ejemplo: varchar, entero, etc. - *Opcionalidad:* ***indica si el atributo debe existir o no, obligatoriamente en la entidad.*** - Longitud: **cantidad de espacio que ocupa el atributo en memoria.** - Dominio: **rango de valores que puede poseer el atributo.** - Claves: **indica si el atributo es clave o no en la entidad.**
5. El usuario agrega los atributos que sean necesarios a la entidad, una vez introducidos los datos, presiona el botón "Finalizar".	**6. El sistema muestra gráficamente la entidad creada.**

Cursos alternos:

3.a) El usuario presiona el botón "Cancelar".

3.a.1) El sistema cierra la ventana "Entidad" y vuelve al espacio de trabajo del usuario.

3.b) El usuario deja de llenar alguno de los campos o los llena incorrectamente.

3.b.1) El sistema muestra un mensaje de error según sea el caso.

5.a) El usuario presiona el botón "Cancelar".

5.a.1) El sistema muestra un mensaje de advertencia, donde se le indica que de cancelar la operación perderá los datos de la entidad.

5.a.1.1) En caso de que este desee continuar con la operación de cancelación, el sistema cierra la ventana "Entidad" y vuelve al espacio de trabajo.

5.a.1.2) Si el usuario decide abortar la operación, el sistema vuelve a la ventana "Atributos".

5.b) El usuario no agrega ningún atributo y presiona el botón finalizar.

5.b.1) El sistema muestra un mensaje donde le indica al usuario que al menos debe de existir un atributo clave dentro de la entidad.

5.c) El usuario no selecciona ninguno de los atributos como clave.

5.c.1) El sistema muestra un mensaje de error indicando que al menos un atributo debe ser clave.

La especificación de los demás casos de usos se presenta en el Apéndice D.

Tomando en cuenta los requerimientos obtenidos a través de las distintas fases, se procedió a diseñar la estructura de clases del sistema (Figura 10), logrando como resultado su diagrama correspondiente. Dicho diagrama proporciona una vista estática, en la que muestran las clases del sistema y sus relaciones, sin ningún tipo de interacción entre ellas.

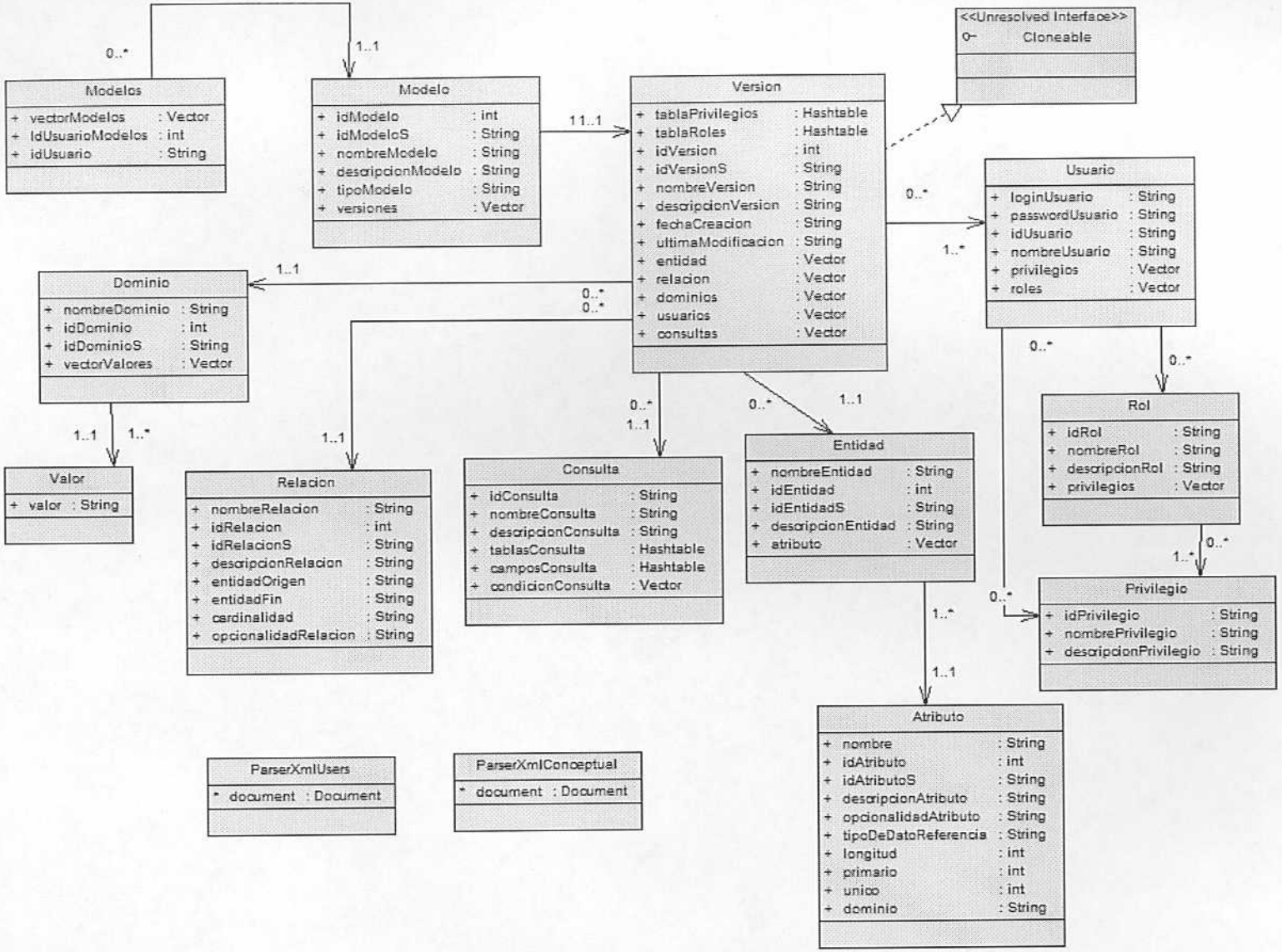

Figura 10. Diagrama de clases (Paquete conceptual).
Fuente: Elaborado por los autores.

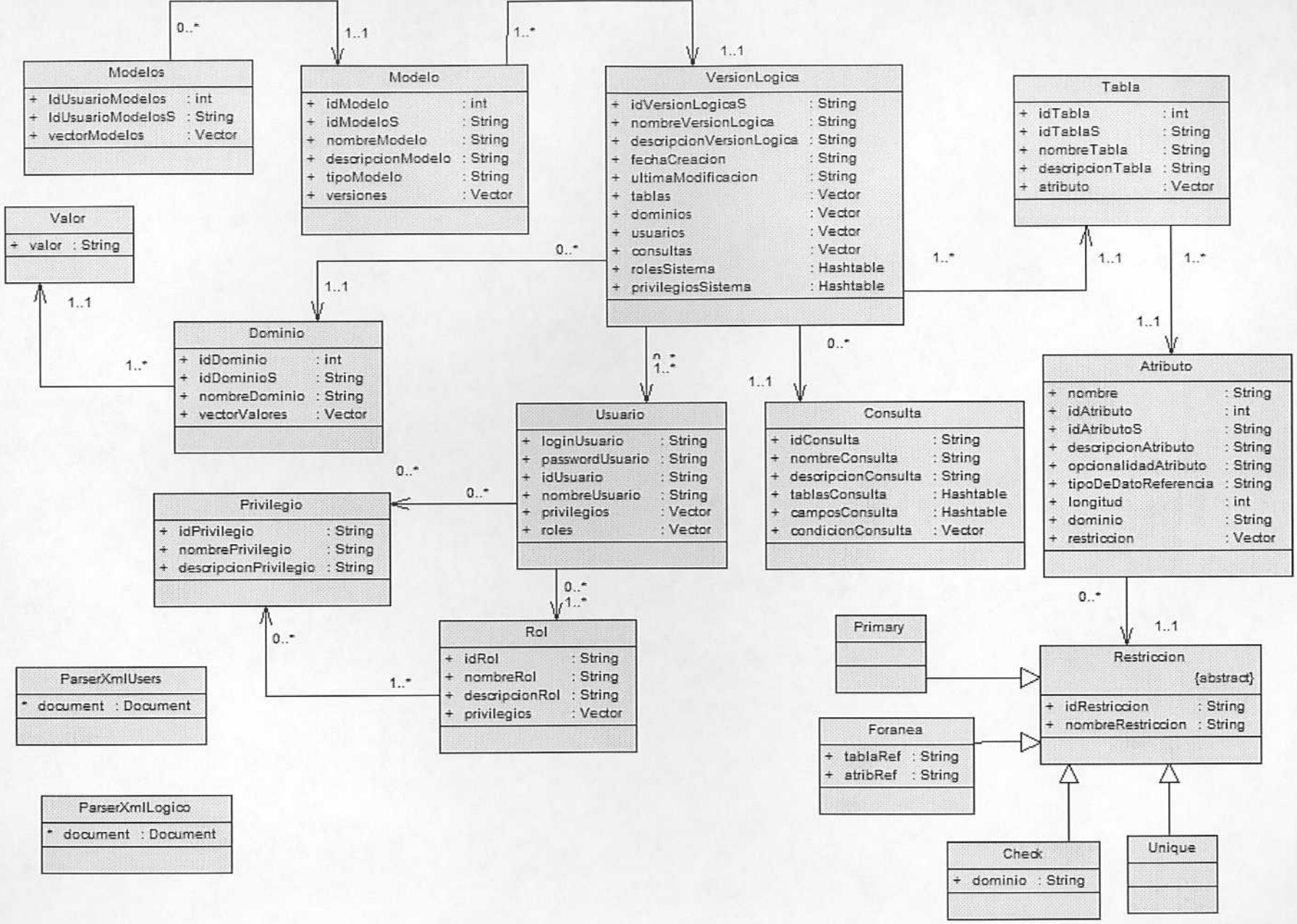

Figura 11. Diagrama de clases (Paquete lógico).
Fuente: Elaborado por los autores.

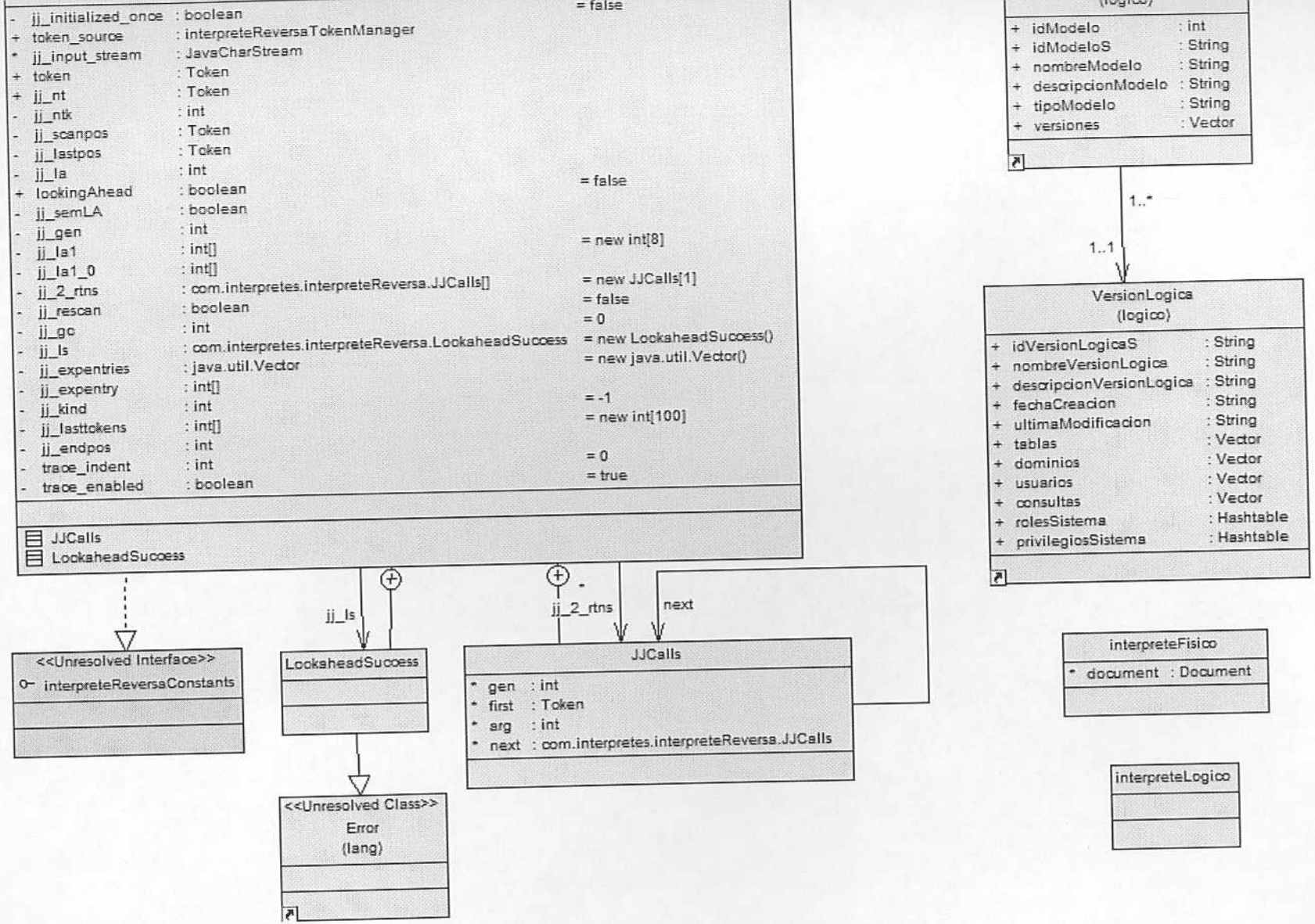

Figura 12. Diagrama de clases (Paquete de intérpretes).
Fuente: Elaborado por los autores.

Una vez estructuradas las clases pertenecientes a la aplicación, se procedió a realizar los diagramas de secuencia, los cuales reflejan la interacción de los objetos del sistema. En la figura 13 se muestra el diagrama de secuencia correspondiente al caso de uso *"Crear entidad"* descrito anteriormente.

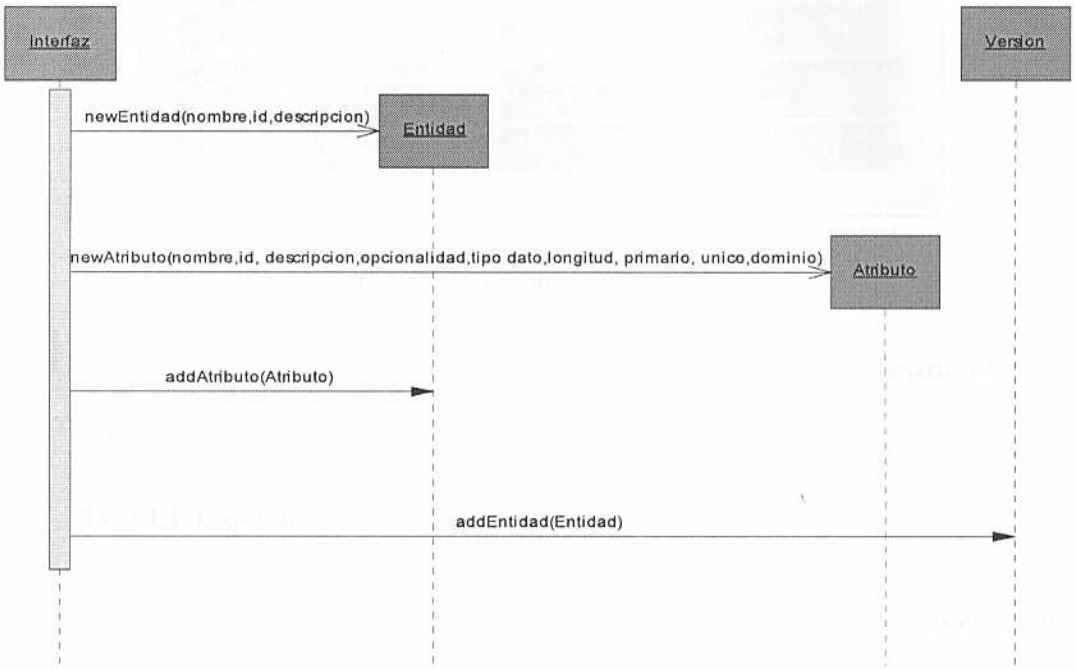

Figura 13. Diagrama de secuencia (Crear Entidad).
Fuente: Elaborado por los autores.

El resto de los diagramas de secuencia se muestran en el apéndice F.

IV.4.1. Descomposición de la arquitectura del sistema.

Habiendo diseñado la arquitectura global del sistema a implantar, se procedió a profundizar en cada una de las capas de la misma, obteniendo así un mayor nivel de detalle.

La descomposición realizada para cada una de las capas de la arquitectura se muestra en la figura 14.

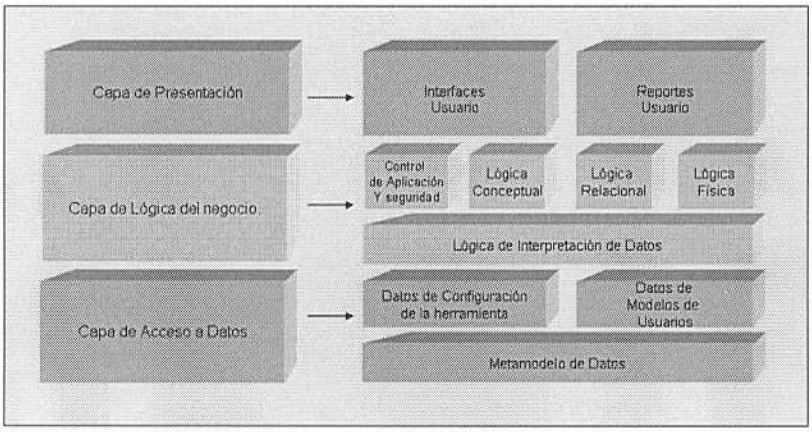

Figura 14 Arquitectura detallada del sistema.
Fuente: Elaborado por los autores.

Para una mejor comprensión de la arquitectura del sistema es necesario detallarla en componentes.

IV.4.1.1. Capa de acceso a datos.

A continuación se describirá de manera detallada las divisiones realizadas en la capa de acceso a datos del sistema implantado.

- **Metamodelo de datos.**

El metamodelo de datos representa la base de la aplicación desarrollada. Sobre este, reposan todas las estrategias de almacenamiento y recuperación de datos del sistema.

Para su diseño se tomó en cuenta la información de almacenamiento necesaria, según los requerimientos capturados en las fases previas, obteniendo como resultado un esquema conceptual representado a través de un diagrama entidad-relación, el cual se ilustra en la figura 15.

Para la construcción del metamodelo, se tomaron en cuenta las fases del diseño de una base de datos, permitiendo el almacenamiento de los datos, en momentos distintos del proceso de construcción de los modelos. Dentro de la estructura planteada se añadieron submodelos, los cuales representan distintos componentes de la herramienta, tales como: el sistema de archivos, el manejo de la seguridad, reportes, etc.

En el apéndice O, se encuentra detallado el diccionario de datos del metamodelo bajo el cual esta soportada la arquitectura de la herramienta.

Una vez completado el esquema conceptual del metamodelo de datos, se procedió a transformar dicho esquema a un esquema lógico, en el cual se especifican detalles de implementación, como: tipos de datos, restricciones de unicidad, claves primarias, foráneas entre otras.

En la figura 16 se muestra el esquema lógico del metamodelo implantado.

Cabe destacar que para la obtención del metamodelo en su versión final, fue necesario manejar avances parciales del mismo, debido a que en el análisis de cada uno de sus componentes, se iba afinando su estructura a fin de modelar de manera precisa cada uno de los requerimientos capturados previamente. Las versiones parciales del metamodelo diseñado se encuentran en el apéndice L.

La implantación de los objetos incluidos en el metamodelo de datos, fue realizada mediante la utilización de documentos XML[9], debido a que estos brindan la flexibilidad de poder organizar la información en el formato que mejor se adapte a los requerimientos. Para obtener más información referente a este metalenguaje, véase el apéndice G.

- **Lógica de Gestión de Datos.**

A fin de realizar una efectiva implantación de la herramienta, se diseñó un sistema de archivos, mediante el cual se almacena la información, de forma organizada y de fácil acceso. Para ello se planteó una arquitectura mediante la cual se presenta una estructura jerárquica de almacenamiento, en la que se diferencian y ordenan cada uno los componentes propios del sistema.

La figura 17 muestra un diagrama del sistema de archivos anteriormente mencionado.

[9] Xtensible Markup Language

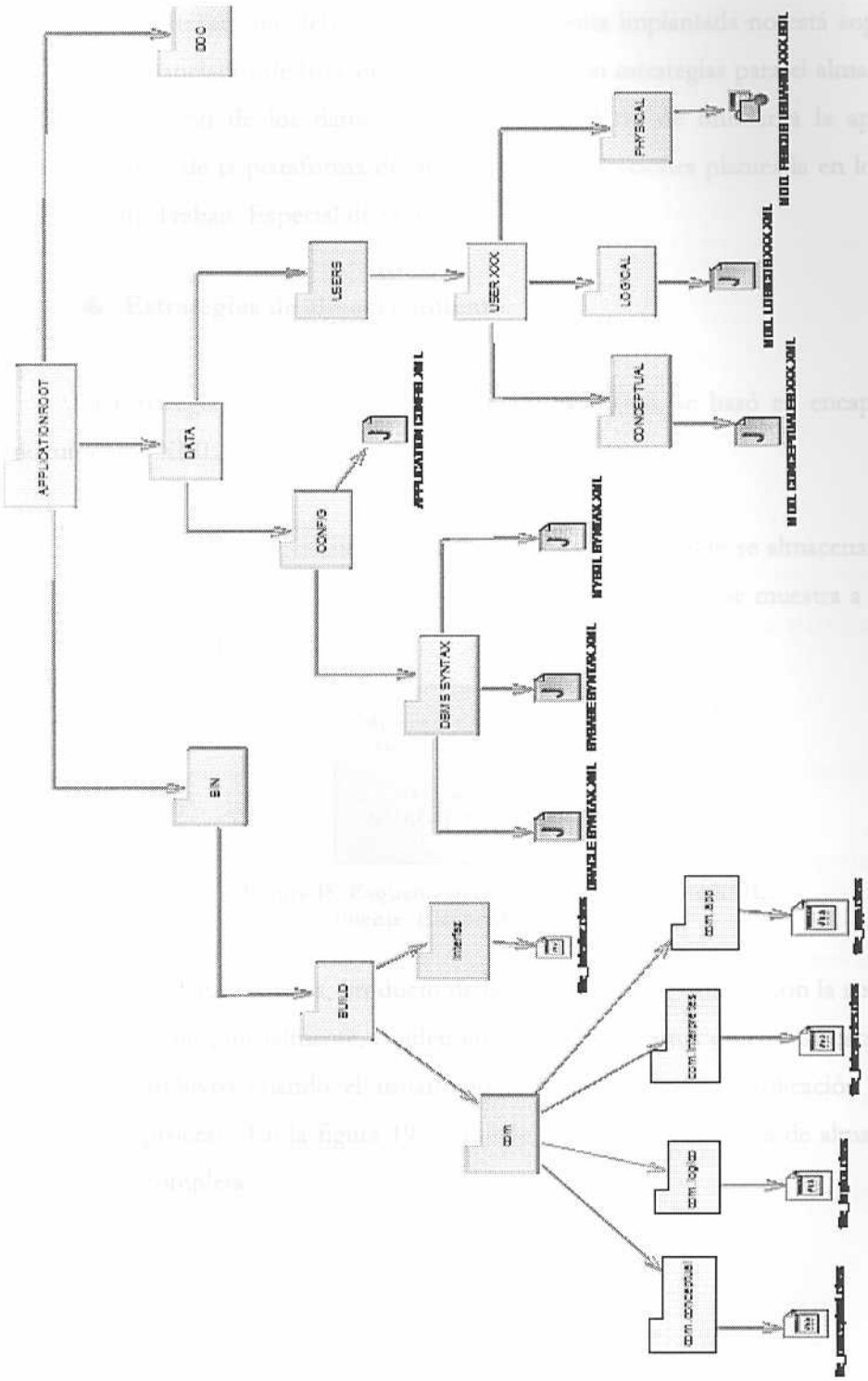

Figura 17. Sistema de archivos de la Herramienta.
Fuente: Elaborado por los autores.

Cabe destacar, que debido a que la herramienta implantada no está soportada por un sistema manejador de base de datos, se definieron estrategias para el almacenamiento y la recuperación de los datos. Todo esto con el fin de brindar a la aplicación, la independencia de la plataforma de desarrollo de aplicaciones planteada en los objetivos del presente Trabajo Especial de Grado.

- **Estrategias de almacenamiento.**

La estrategia para el almacenamiento de los datos, se basó en encapsularlos en documentos XML, siguiendo los siguientes lineamientos:

a) Se definió la estructura del documento XML donde se almacenan los datos. El esquema general de un documento de este tipo, se muestra a través de la figura 18.

```
<?xml version="1.0"?>
  <tagPadre>
     <tagHijo1 atributo="valor_1" />
     <tagHijo2 atributo="valor_2" />
  </tagPadre>
```

Figura 18. Esquema general de un documento XML.
Fuente: Elaborado por los autores.

b) Se crean objetos, producto de la interacción del usuario con la aplicación, los cuales, inicialmente, residen en la memoria. Estos objetos se escriben en los archivos cuando el usuario lo solicite o cuando la aplicación realice este proceso. En la figura 19, se puede observar la estrategia de almacenamiento completa.

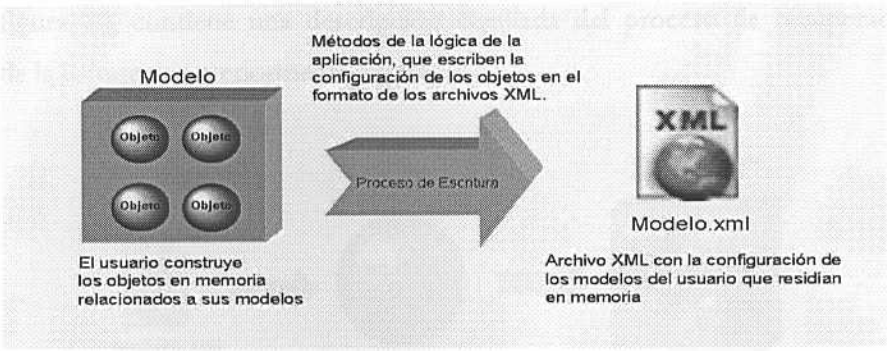

Figura 19. Estrategia de almacenamiento.
Fuente: Elaborado por los autores.

Estrategias de recuperación.

Así como fueron necesarias estrategias para el almacenamiento de los datos, también se requirió del diseño de estrategias de recuperación, de la información escrita en documentos XML.

Los pasos definidos para la recuperación de datos dentro de la herramienta fueron:

a) Inicialmente se realiza un proceso de lectura e interpretación de la información que se desea recuperar. Este se efectúa cargando el documento que contiene la información a la memoria utilizando estructuras DOM[10]. (Ver apéndice G)

b) Una vez almacenada la información en los objetos DOM, se manipula para realizar las operaciones que se deseen. Estos cambios se escriben de nuevo en el archivo XML, completando así, el proceso de recuperación y escritura de los datos.

[10] Document Object Model.

La figura 20, contiene una descripción detallada del proceso de recuperación y escritura de la estrategia anteriormente explicada.

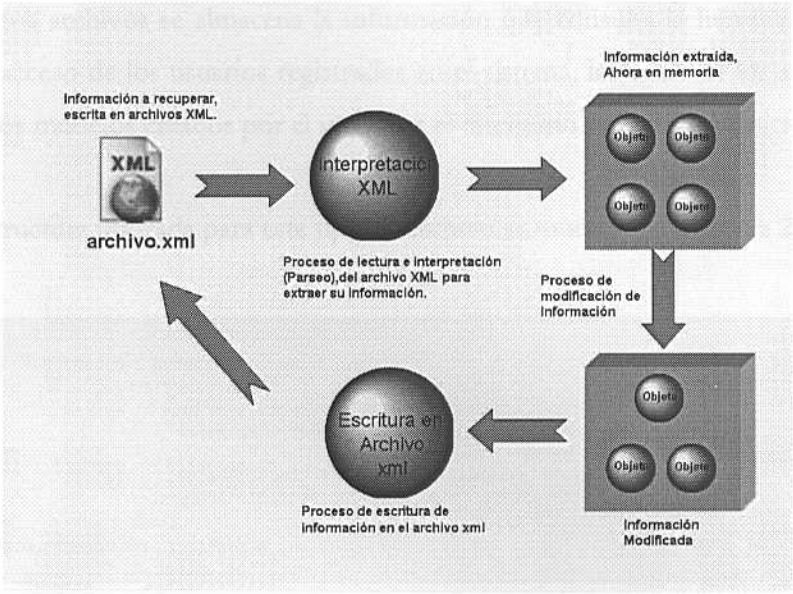

Figura 20. Estrategia de recuperación.
Fuente: Elaborado por los autores.

Las estrategias descritas en puntos anteriores, fueron utilizadas durante todo el desarrollo, con el fin de unificar los procesos de almacenamiento y recuperación de datos propios de la herramienta. Los detalles de la lógica e implantación de las estrategias diseñadas para la gestión de datos, se encuentran especificados posteriormente en la capa de lógica del negocio de la arquitectura.

- **Datos de configuración de la herramienta.**

La información asociada a la configuración del sistema engloba los datos necesarios para realizar las operaciones internas de la herramienta, tales como: control de acceso y seguridad, configuración del sistema de archivos, etc. Esta información se encuentra almacenada en dos tipos de archivos XML: los de configuración y los de sintaxis de los manejadores de base de datos.

Archivos de configuración de la herramienta.

En estos archivos se almacena la información que consulta la herramienta para el control de acceso de los usuarios registrados en el sistema, las rutas de los archivos que contienen los modelos creados por el usuario y el directorio raíz de la aplicación.

La estructura utilizada para este tipo de archivo se muestra en la figura 21:

```
<applicationConfig appRoot="C:/....." >
        <usersInfo>
            <appUser idUsuario="" nombreUsuario="" loginUsuario="" passwordUsuario="">
                <conceptualPath>"C:/....."</conceptualPath>
                <logicalPath>"C:/....."</logicalPath>
                <physicalPath>"C:/....."</physicalPath>
            </appUser>
        </usersInfo>
</applicationConfig>
```

Figura 21. Esquema general del archivo de configuración.
Fuente: Elaborado por los autores.

Archivos de Sintaxis de Sistemas Manejadores de Base de Datos.

Los archivos de sintaxis contienen la información asociada a las sentencias SQL[11] necesarias para la generación de los archivos de creación de bases de datos. Existe un archivo por cada manejador, para el presente Trabajo Especial de Grado, se contemplaron las sintaxis de los manejadores Oracle, Mysql y Sybase.

A continuación, en la figura 22, se presenta un fragmento del archivo de sintaxis correspondiente a la sentencia "CREATE TABLE" para el sistema manejador de base de datos "ORACLE".

[11] Structured Query Language.

```
<?xml version="1.0"?>
  <config nombreManejador="ORACLE">
    <sentencia>
      <create>
        <createTable>
            <abreTabla>CREATE TABLE %NOMBRE_TABLA%( </abreTabla>
            <columna> %NOMBRE_ATRIBUTO% %TIPO_DATO% (%LONGITUD%)  %OPCIONALIDAD% </columna>
            <cierraTabla>);</cierraTabla>
        </createTable>
            .
            .
            .
            .
            .
      </create>
    </sentencia>
  </config>
```

Figura 22. Fragmento del archivo de sintaxis de ORACLE.
Fuente: Elaborado por los autores.

➢ Datos y Modelos de Usuarios.

Son todos aquellos archivos que almacenan los modelos de datos creados por el usuario, durante todas las etapas del proceso de diseño de una base de datos. Cada usuario posee un espacio de trabajo, en el cual se encuentran todos sus modelos, con la finalidad de poder recuperarlos y modificarlos cuando lo requiera.

Los archivos asociados al espacio de trabajo de un usuario se almacenan como se ilustra en la figura 23.

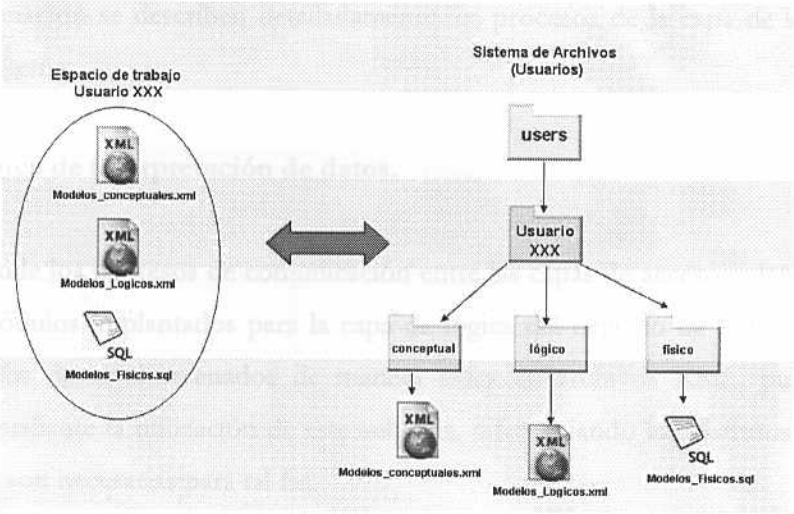

Figura 23. Estructura de almacenamiento de los espacios de trabajo de usuarios.
Fuente: Elaborado por los autores.

Como se pudo observar en la figura anterior, para cada usuario, se pueden crear 3 categorías de archivos, en relación a cada una de las fases del proceso de diseño de una base de datos: conceptual, lógica y física.

Para el tipo de archivo asociado a los modelos conceptuales de los usuarios, se almacenan datos como: entidades, atributos, relaciones, dominios, usuarios, roles, privilegios y consultas. En el apéndice H se encuentran los diagramas de la estructura XML, de cada una de las etapas de diseño de una base de datos.

IV.4.1.2. Capa lógica del negocio.

En esta capa de la arquitectura, se encuentran todas aquellas operaciones lógicas necesarias para la creación, recuperación y modificación de modelos de datos creados por el usuario, así como también los necesarios para la gestión de procesos internos de la herramienta. La funcionalidad implantada está representada por un API,[12] mediante el cual se proveen las operaciones requeridas para la manipulación y recuperación de datos dentro del metamodelo descrito en anteriores oportunidades.

A continuación se describen detalladamente los procesos de la capa de lógica del negocio del sistema.

- **Lógica de interpretación de datos.**

Comprende los procesos de comunicación entre las capas de acceso a datos y cada uno de los módulos implantados para la capa de lógica del negocio de la herramienta. Básicamente los datos almacenados de manera física en archivos XML, pueden ser recuperados mediante la utilización de esta subcapa, diferenciando los distintos tipos de funciones que son necesarias para tal fin.

---

[12] Application Program Interface.

Esto se realiza mediante la implementación de intérpretes o "parsers" los cuales traducen dicha información a objetos en memoria para su posterior manipulación (Ver apéndice B). En la figura 24, se muestra el proceso de construcción de los objetos a nivel de memoria.

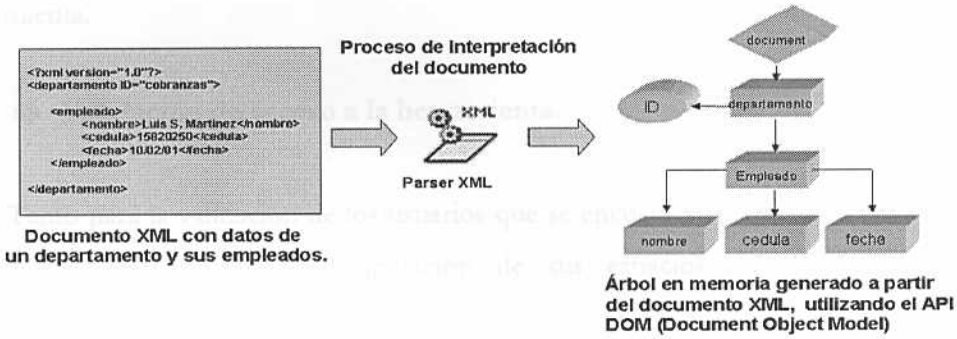

Figura 24. Proceso de interpretación y creación en memoria de árbol DOM.
Fuente: Elaborado por los autores.

Este proceso se realiza para obtener los datos tanto de las operaciones de la aplicación a nivel de control, como los de datos relacionados a los modelos creados por el usuario. Por tanto, luego de haber cargado la información en objetos a nivel de memoria, puede ser manipulada y escrita nuevamente en su archivo correspondiente. Se implementaron intérpretes para cada tipo de archivo diseñado (Figura 25).

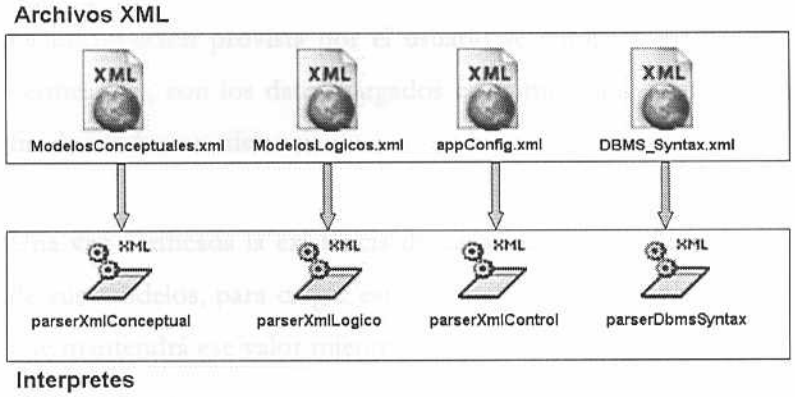

Figura 25. Interpretes implementados en la herramienta.
Fuente: Elaborado por los autores.

- **Control de aplicación y seguridad de datos.**

Mediante la capa de control de aplicación y seguridad, se implementan todas aquellas funcionalidades referentes a la gestión de usuarios, operaciones de validación de accesos, administración de espacios de trabajo y los procesos de instalación de la herramienta.

- **Validación de acceso a la herramienta.**

Tanto para la validación de los usuarios que se encuentran registrados dentro de la herramienta, como para la recuperación de sus espacios de trabajo se realiza internamente el siguiente proceso:

a) Inicialmente el usuario se encuentra registrado en el sistema, por lo que la información acerca de su espacio de trabajo, se encuentra almacenada físicamente en el documento XML correspondiente.

b) La aplicación internamente realiza un acceso al documento correspondiente, utilizando la estrategia de recuperación descrita y carga a nivel de memoria la información de configuración de los usuarios.

c) La información provista por el usuario se compara mediante funciones de verificación, con los datos cargados en la memoria en el paso anterior, con fin de validar su existencia.

d) Una vez verificada la existencia de usuario, se procede a ubicar los archivos de sus modelos, para cargar estos datos a nivel de memoria en una variable que mantendrá ese valor mientras el usuario se encuentre dentro de su sesión (Ver figura 26).

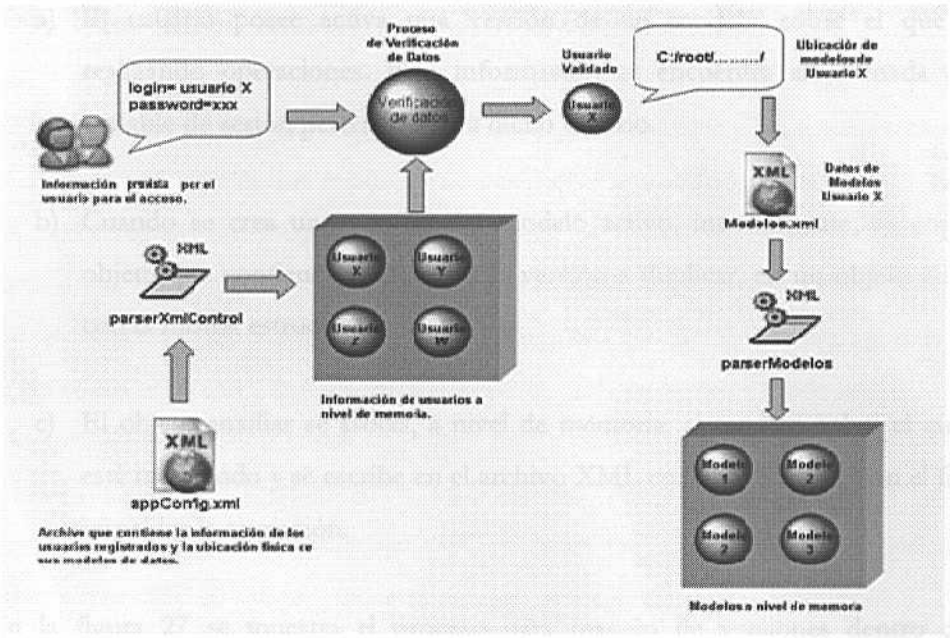

Figura 26. Proceso de validación de acceso a la herramienta.
Fuente: Elaborado por los autores.

- **Lógica conceptual.**

La lógica conceptual de la aplicación engloba todas aquellas operaciones necesarias para la construcción, recuperación y modificación de los modelos conceptuales de datos creados por los usuarios dentro de la herramienta. Permite manipular aquellos objetos pertenecientes a un modelo de datos a nivel conceptual como: entidades, atributos, relaciones, dominios, consultas, usuarios, roles y privilegios.

- **Manejo de versiones.**

Una consideración importante de la implantación de los modelos de datos, es el manejo de versiones. Estas, permiten mantener avances de los modelos almacenados con la finalidad de no perder la información inicial, previamente almacenada. La lógica para almacenar los datos de las versiones, se describe a continuación:

a) El usuario posee activa una versión de un modelo sobre el que está realizando operaciones. Esta información se encuentra almacenada en la variable de sesión perteneciente a dicho usuario.

b) Cuando se crea una versión del modelo activo, internamente, se copia el objeto que contiene los datos de la versión a duplicar, en un objeto auxiliar con la misma estructura.

c) El objeto auxiliar se asocia, a nivel de memoria, al modelo sobre el cual se está trabajando y se escribe en el archivo XML correspondiente, con el fin de respaldar la operación.

En la figura 27 se muestra el proceso para manejo de versiones dentro de la herramienta.

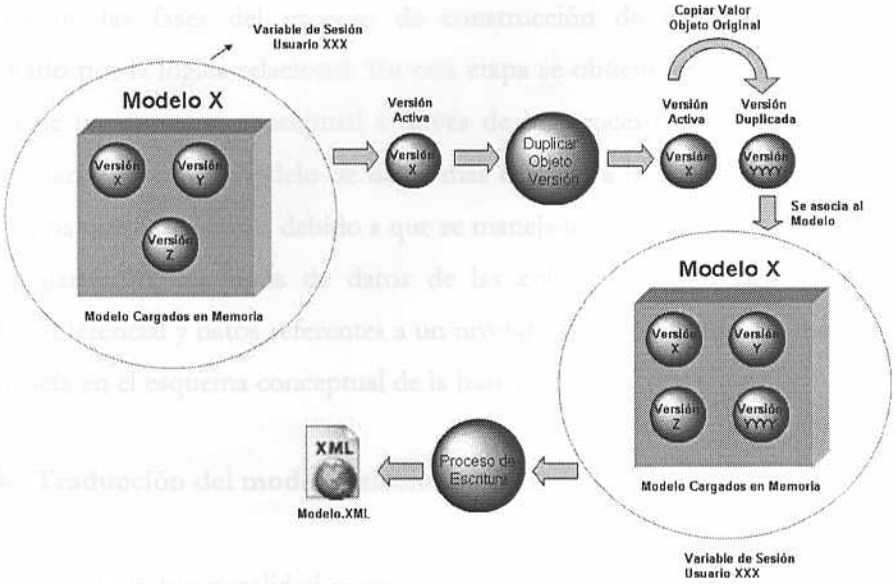

Figura 27. Manejo de versiones.
Fuente: Elaborado por los autores.

Traducción del modelo conceptual al modelo relacional.

Entre las funcionalidades implementadas para los modelos conceptuales de datos, se encuentra un intérprete que recibe como parámetro un modelo conceptual de datos y genera como salida un modelo relacional. Esto se realiza mediante una lógica de traducción que establece, aquellas reglas necesarias para transformar un modelo de alto nivel a un modelo relacional más cercano a la implementación de la base de datos. En líneas generales el proceso seguido para la implantación del intérprete conceptual, consta de transformar entidades a tablas, atributos claves en restricciones de clave primaria o unicidad, verificar la cardinalidad y opcionalidad de las relaciones para generar restricciones de integridad referencial o claves foráneas, dominios a restricciones de dominio, generación de índices, usuarios, roles y privilegios.

Lógica relacional.

Otra de las fases del proceso de construcción de una base de datos, está representado por la lógica relacional. En esta etapa se obtiene un modelo lógico, el cual se deriva de un esquema conceptual a través de un proceso de interpretación. A este nivel el usuario posee un modelo de datos más cercano a la implementación final de la base de datos que desea crear, debido a que se maneja información acerca de los espacios de almacenamiento, los tipos de datos de las columnas, volumen, restricciones de integridad referencial y datos referentes a un nivel de abstracción mucho mas bajo que el que se poseía en el esquema conceptual de la base de datos.

Traducción del modelo relacional al modelo físico.

Además de la funcionalidad correspondiente a esta etapa en el diseño de base de datos, también se incluye dentro de esta lógica un intérprete que posee como parámetro de entrada un modelo relacional generado por la herramienta y como salida genera un archivo de creación de la base de datos que se esta manipulando, dependiendo del sistema manejador de bases de datos que el usuario elija.

Para la traducción del modelo relacional al modelo físico de datos se diseño la siguiente lógica:

a) Una vez elegido el sistema manejador de base de datos para el que se desea generar el modelo físico, la herramienta consulta el archivo de sintaxis propio del mismo y lo carga en una tabla de dispersión en la memoria.

b) Los componentes del modelo relacional se verifican contra las plantillas de sintaxis, y se sustituyen con los valores correspondientes. Cada una de las sentencias generadas se escriben en el archivo SQL del modelo físico creado. (Ver figura 28).

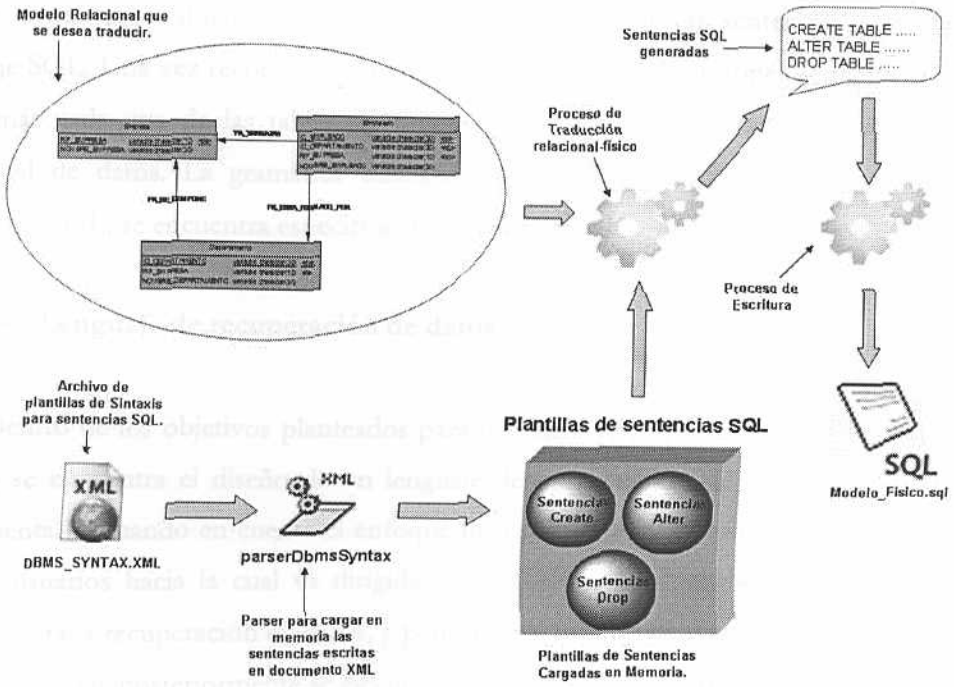

Figura 28. Traducción del modelo relacional al modelo físico.
Fuente: Elaborado por los autores.

- **Lógica física.**

Representa la última fase en el proceso de construcción de una base de datos, a este nivel se posee un archivo que contiene tanto, las sentencias de creación de la base de datos como toda la configuración de usuarios, roles y privilegios del sistema.

- **Proceso de ingeniería reversa del modelo físico al modelo relacional.**

Dentro de las funcionalidades de esta capa de la arquitectura, se incluyó un intérprete que realiza el proceso de ingeniería reversa a partir de un archivo SQL, obteniendo un modelo de datos relacional dentro de la herramienta.

Para realizar este proceso fue necesario implementar un analizador sintáctico y léxico utilizando la herramienta llamada JavaCC[13] (Ver apéndice I), mediante el cual se establecieron reglas sintácticas para reconocer cada una de las sentencias escritas en lenguaje SQL. Una vez reconocidas las sentencias se ejercieron acciones semánticas para almacenar cada una de las tablas, columnas, restricciones, etc., mediante un modelo relacional de datos. La gramática utilizada para el reconocimiento sintáctico de las sentencias SQL, se encuentra especificada en el apéndice K.

- **Lenguaje de recuperación de datos.**

Dentro de los objetivos planteados para la realización de este Trabajo Especial de Grado, se encuentra el diseño de un lenguaje de recuperación de datos, propio de la herramienta. Tomando en cuenta el enfoque bajo el cual se desarrolló la aplicación, y el tipo de usuarios hacia la cual va dirigida, el lenguaje a crear debía poseer sentencias sencillas para la recuperación de datos, y permitirle al usuario de manera intuitiva generar las consultas que posteriormente se ejecutarán sobre las bases de datos creadas.

---

[13] Java Compiler Compiler

Como consecuencia de lo descrito anteriormente se procedió a diseñar e implementar un lenguaje de recuperación con las siguientes características:

- Las sentencias para las consultas estarán escritas en un lenguaje de alto nivel, semejante al lenguaje natural.

- El lenguaje de recuperación sigue la siguiente estructura:

  Buscar *campo 1,campo 2,campo 3*
  En *tabla 1,tabla 2,tabla 3,..., tabla n*
  bajo las condiciones *cond 1, cond 2, cond 3*

- Las consultas se escriben bajo el lenguaje diseñado, y luego son interpretadas para generar las consultas en lenguaje SQL.

IV.4.1.3. Capa de presentación.

En la capa de presentación se encuentran todos aquellos medios por los cuales el usuario interactúa con la herramienta para ejecutar sus operaciones.

➢ **Interfaces de usuario.**

Las interfaces de usuario dentro de la herramienta, fueron diseñadas de tal manera que el usuario estuviese asistido durante todo el proceso de creación de sus modelos. Las operaciones que el usuario puede realizar, se encuentran bien especificadas dentro de las pantallas de la aplicación, ya que se diseñaron menús con iconos nemotécnicos de fácil asociación con las operaciones posibles dentro del sistema.

A continuación se presentan algunas pantallas (Figura 29 y 30), de la interfaz de usuario diseñada para la aplicación.

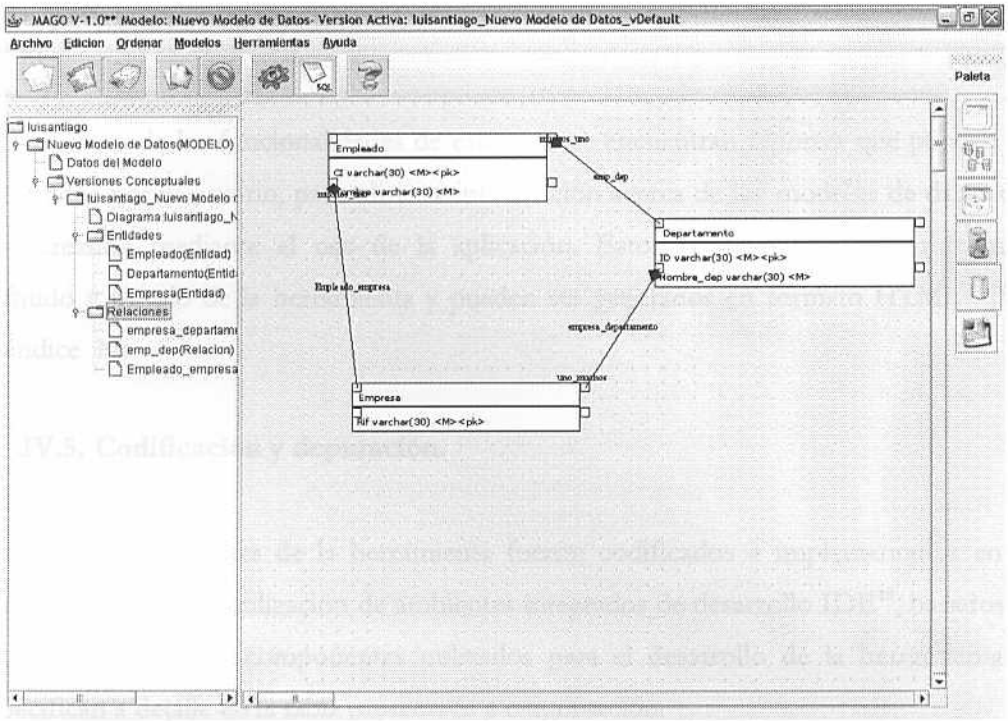

Figura 29. Pantalla principal de la Herramienta.
Fuente: Elaborado por los autores.

Figura 30. Pantalla para crear atributos a una entidad.
Fuente: Elaborado por los autores.

El resto de las pantallas se encuentran en el apédice M.

- **Reportes de usuario.**

Dentro de las funcionalidades de esta capa se encuentran reportes que pueden ser solicitados por el usuario, para obtener información acerca de los modelos de datos que esta creando mediante el uso de la aplicación. Estos reportes poseen un formato definido y propio de la herramienta y pueden ser generados en formato HTML[14] (Ver apéndice N).

IV.5. Codificación y depuración.

Los componentes de la herramienta fueron codificados e implementados en su mayoría mediante la utilización de ambientes integrados de desarrollo IDE[15], basados en tecnología Java. Los componentes utilizados para el desarrollo de la herramienta se especifican a detalle en la tabla presentada a continuación.

Ambiente de Programación	Descripción del ambiente.	Uso para la implementación.
JavaTM 2 Platform Standard Edition Development Kit (JDKTM).	Es un entorno de desarrollo para la construcción de aplicaciones y componentes mediante el uso del lenguaje de programación Java. Además de esto posee una amplia documentación de las componentes prediseñados para el desarrollo de aplicaciones API (Application Program Interface).	Se utilizó para todas las operaciones a realizar dentro de la herramienta, debido a que todas las capas de la arquitectura están codificadas en lenguaje Java.
Apache Ant 1.6	Es una herramienta de construcción, mediante la cual se pueden realizar procesos de compilación, basados en directivas almacenadas en documentos XML de configuración.	Se utilizó para los procesos de compilación de los paquetes que conformaban la lógica y presentación de la aplicación desarrollada.
Javacc 3.5	Es un generador de analizadores sintácticos descendentes, mediante la utilización de	Se usó para la construcción del analizador sintáctico y semántico utilizado para el proceso de

[14] Hyper Text Markup Language.

[15] Integrated Development Enterprise

	lenguaje Java.	ingeniería de reversa implantado en la capa de Lógica Física de la arquitectura.
JCreator Pro 3.5.0009	Es un entorno de desarrollo mediante el cual pueden ser editados, compilados y ejecutados programas escritos en Lenguaje Java soportado por el JDK (Java Development Kit).	Se utilizó para la codificación y depuración del código que fue escrito en Lenguaje Java.
NetBeansTM IDE 4.0	Es un ambiente integrado del desarrollo (IDE) para la edición, compilación y depuración, de programas basados en tecnología Java.	Se utilizo para el diseño de las interfaces de usuario, y para la integración entre las capas de lógica del negocio y presentación de la arquitectura propuesta.

Tabla 8. Ambientes de desarrollo.
Fuente: Elaborado por los autores.

IV.5.1. Estructura de paquetes de la herramienta.

La aplicación para su codificación fue dividida en 3 grandes paquetes con el fin de mantener la estructura propuesta dentro de su arquitectura. Por lo tanto fueron estructuradas según el diagrama que se muestra en la figura 29.

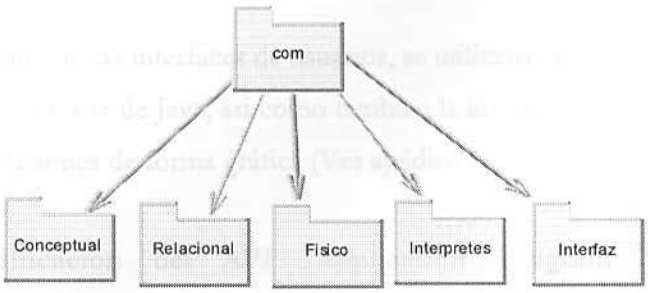

Figura 31. Estructura de paquetes de la herramienta.
Fuente: Elaborado por los autores

Las clases implementadas en cada uno de los paquetes interactúan entre ellas, debido a que por la dependencia planteada entre los paquetes se requieren métodos que tienen que ser aplicados a objetos de otras clases.

Los detalles de los estándares de codificación utilizados para el desarrollo se encuentran especificados en el apéndice J.

IV.5.2. Librerías utilizadas para la implementación de los componentes de la arquitectura.

Para la codificación de los componentes de la arquitectura se utilizaron diversas librerías que contienen los diferentes métodos necesarios para la programación de la lógica y presentación de la aplicación. En líneas generales dentro de las clases de la aplicación se utilizaron los paquetes comunes de Java para la manipulación de estructuras de almacenamiento como: vectores, arreglos, tablas de hash (*java. util. *)*, cadenas de caracteres (*java. lang . **), operaciones matemáticas, entrada y salida de datos (*java .io . *)*.

Para la implementación de los intérpretes (Parsers) para la lectura y carga de datos desde archivos XML, se utilizaron las librerías javax.xml.parsers y para la construcción en memoria de la estructura jerárquica del documento XML (DOM) se utilizó la librería org.w3c.dom.

Para la creación de las interfaces de usuarios, se utilizaron los componentes propios de las librerías swing y awt de Java, asi como tambien la librería GEF para la creación de las entidades y relaciones de forma gráfica (Ver apédice Q).

IV.5.3. Codificación del API (Application Program Interface) para manipulación del metamodelo.

Para realizar la codificación de los métodos pertenecientes al API de manipulación del metamodelo de datos, se tomaron en cuenta las funciones diseñadas para la manipulación de los objetos pertenecientes a cada clase de la herramienta. Los métodos implementados en principio fueron los métodos get (para obtener ese objeto) y set (para modificar ese objeto) de cada una de las clases y posteriormente, los métodos que ejecutaran las operaciones necesarias para el correcto funcionamiento de la herramienta.

A continuación se presenta como ejemplo, el método de la clase "Versión" perteneciente al paquete com.conceptual, en el que se obtiene un objeto de la clase "Entidad", a partir de una cadena de caracteres asociada a su nombre.

```
public Entidad getEntidadByName(String NombreEntidad)
{
for(int i=0;i<entidad.size();i++){
        Entidad ent=(Entidad)entidad.elementAt(i);
        if(NombreEntidad.equals(ent.nombreEntidad)){
        return ent;}
}
return null;
}
```

Para la obtención final del API, en un formato acorde para su presentación, se utilizaron técnicas de documentación propias de los estándares planteados por el JDK[16] para la posterior utilización del comando Javadoc, mediante el cual se genera un documento HTML con la información detallada de cada una de las clases que componen el sistema.

Los detalles de las clases, métodos y objetos que conforman el API de manipulación del metamodelo de datos construido, se encuentran especificados en el Apéndice Ñ.

IV.6. Pruebas.

Al haber completado la implementación de los módulos pertenecientes a la arquitectura de la herramienta, se procedió a efectuar las pruebas correspondientes que validan los resultados obtenidos mediante el proceso de desarrollo del sistema.

[16] Java Development Kit

Las pruebas realizadas para validar todos y cada uno de los componentes de la herramienta, se basaron principalmente en pruebas de caja negra mediante los cuales se verificaría las respuestas del sistema a cada una de las operaciones solicitadas y pruebas de usabilidad realizadas directamente con usuarios con el fin de validar los requerimientos planteados al inicio del desarrollo.

IV.6.1.Pruebas de caja negra.

Las pruebas de caja negra verifican las respuestas de la aplicación a solicitudes realizadas por los actores del sistema, y se basan en cada una de las acciones realizadas en los casos de uso planteados en etapas anteriores del desarrollo.

A continuación se presenta el caso de prueba caja negra para el caso de uso "Crear Modelo".

Caso de Prueba #4: Crear Modelo.

Prueba 4.1 Curso normal.

Descripción: Prueba que el usuario pueda agregar un modelo a su espacio de trabajo.

Entrada: El usuario presiona el icono correspondiente a la opción "Nuevo modelo" en la barra de herramientas e introduce los datos.

Salida: El sistema habilita un espacio de trabajo para el modelo.

Prueba 4.2 Curso alterno.

Descripción: Verifica que el sistema no cree el modelo al presionar el botón "Cancelar".

Entrada: El usuario presiona el botón "Cancelar" en la ventana "Modelo".

Salida: El sistema cierra la ventana "Modelo" y retorna al espacio de trabajo del usuario.

Prueba 4.3 Curso alterno:

Descripción: Verifica que el sistema muestre un mensaje de error en el caso que el usuario llene erróneamente los datos o deje algún campo vacío.

Entrada: El usuario al crear el modelo, introduce incorrectamente los datos solicitados o deja de ingresar alguno.
Salida: El sistema muestra un mensaje de error.

En el apéndice P se encuentran especificadas las de pruebas restantes del sistema.

IV.6.2.Pruebas con usuarios.

A fin de validar la funcionalidad implementada durante el desarrollo de la herramienta, se realizaron pruebas de usabilidad del sistema, tomando como muestra un grupo de estudiantes de la escuela de Ingeniería Informática de la Universidad Católica Andrés Bello.

Al igual que en el inicio del desarrollo, se plantearon variables que miden el grado de usabilidad y aceptación, por parte de los usuarios, del sistema implementado.
Dichas variables se pueden puntualizar en:

- Grado de usabilidad y que tan amigables son las interfaces implementadas.
- Tiempo de respuesta de las operaciones.
- Confiabilidad de las operaciones, en cuanto a pérdida de datos o errores en el funcionamiento de la herramienta.
- Asistencia durante el proceso de creación de modelos mediante el uso de la herramienta, midiendo así la tolerancia a fallas.
- Calidad de la presentación de los archivos generados por la herramienta.
- Opiniones y nivel general de satisfacción de los usuarios.

Mediante la aplicación de dichas pruebas se logró una receptividad positiva por parte de los estudiantes encuestados, así como también recomendaciones acerca de posibles mejoras o incrementos futuros en la funcionalidad de la herramienta.

En el apéndice P se encuentra la encuesta realizada a los usuarios luego de las pruebas de usabilidad, así como también los resultados obtenidos por parte de los estudiantes.

CAPÍTULO V. RESULTADOS

Mediante la utilización de la metodología Cascada con fases solapadas, se realizó de manera satisfactoria el desarrollo del sistema planteado para el presente Trabajo Especial de Grado. De esta manera, se alcanzaron todos los objetivos trazados al inicio del desarrollo, así como también se obtuvieron los resultados esperados para el cumplimiento todos los requerimientos.

Los resultados obtenidos mediante el desarrollo de la herramienta propuesto para este Trabajo Especial de Grado fueron los siguientes:

- La herramienta implementada le brinda al usuario la posibilidad de crear, recuperar y alterar modelos de datos, sin la necesidad de poseer conocimientos en el lenguaje de bajo nivel (SQL). Todo esto apoyado en la independencia brindada por la herramienta en cuanto a los sistemas de gestión de base de datos, permitiéndole crear modelos, sin necesidad de conocer la configuración o la sintaxis de las sentencias necesarias para la implementación de las bases de datos.

- Se diseñó e implementó una lógica de gestión de versiones de los modelos de datos creados por el usuario, permitiendo realizar modificaciones a dichos modelos, teniendo así la posibilidad de almacenar la información que el usuario requiera para el diseño y creación de la base de datos. Con esto se le brinda al usuario la posibilidad de mantener activa la información de sus modelos, conservando la compatibilidad entre los datos que contiene.

- La herramienta implementada posee funciones de gestión de seguridad, creación y administración de perfiles de usuarios, así como también de control de acceso mediante la autentificación de datos de usuarios, con el fin de resguardar la seguridad e integridad de los datos almacenados en cada espacio de trabajo. Esto permite poder mantener múltiples sesiones y evitar accesos indeseados al espacio

de trabajo de un usuario en específico. En cuando a la creación de perfiles de seguridad dentro de los modelos creados por el usuario, la herramienta provee la funcionalidad de asignación de roles y privilegios tanto sobre el esquema que se desea generar como sobre los objetos que lo componen.

- Para la implementación de la lógica de la herramienta, se construyó un API , que contiene todas aquellas funciones necesarias para la manipulación del metamodelo de datos construido. Mediante este, la herramienta realiza todas las gestiones referentes a los modelos de datos (conceptuales, relacionales y físicos), funciones de administración (administración de perfiles de usuarios, contraseñas, control accesos, lógica de transformación entre fases de modelos) y el manejo de interfaces de usuarios necesarias para la interacción del usuario con la herramienta.

- Se definió un lenguaje de recuperación de datos, basado en estrategias de recuperación y almacenamiento, utilizando el lenguaje extensible de marcas (XML). Para construir consultas a través de este lenguaje, el usuario solo debe utilizar la sintaxis de un lenguaje natural, que mediante el apoyo de la lógica de consultas de la herramienta, es capaz de generarlas en lenguaje SQL de forma transparente.

- Para cubrir con el objetivo general del presente Trabajo Especial de Grado, se diseñó e implantó un metamodelo de datos escalable, flexible e independiente de la plataforma de desarrollo, con el cual se pueden manipular los modelos de datos mediante una interfaz, así como también la lógica de gestión de procesos y datos.

Se implementó una herramienta bajo una arquitectura de capas a tres niveles, la cual posee en la parte superior la presentación con el usuario (reportes, interfaces, etc.), luego le sigue una capa que maneja la lógica del negocio para realizar la manipulación de los objetos de todos los procesos que realiza la

herramienta durante su funcionamiento, y por ultimo, en el nivel inferior, la capa de acceso a datos que representa la base de toda la aplicación.

La aplicación implantada como herramienta CASE, se clasifica dentro de Toolkit por su amplitud y Middlecase haciendo referencia al ciclo de vida que logra automatizar, las cuales incluyen las actividades relacionadas a el análisis, diseño e implementación de los modelos de datos.

CONCLUSIONES

- El nivel de conocimiento poseído por usuarios al desarrollar modelos de datos, es un factor determinante al momento de escoger herramientas que los ayuden en la automatización de los procesos correspondientes a las fases propuestas para su desarrollo. Dichos conocimientos, podrían ser cada vez menores, si se implementan herramientas que asistan al usuario en cada una de las fases de diseño.

- Durante el desarrollo del presente Trabajo Especial de Grado se ratificó que el manejo de versiones de modelos de datos dentro de una herramienta CASE, no resulta un tema trivial, debido a que implica una lógica de gestión que requiere de una gran cantidad de validaciones y operaciones que permita mantener consistencia a nivel de los datos, así como también la lógica de gestión de almacenamiento de todas y cada una las versiones generadas a partir de un modelo de datos.

- La gestión de perfiles de usuario y seguridad de datos tanto en la aplicación como en los modelos creados, representan una característica fundamental en el desarrollo de aplicaciones, debido a que los espacios de trabajo mantenidos para cada usuario, así como también su administración, se corresponden con dichos perfiles, en cuanto a que se realicen solo accesos autorizados y se mantenga la consistencia e integridad de los datos almacenados.

- La sintaxis asociada al lenguaje recuperación en sistemas de gestión de bases de datos, es estricta en cuanto a la construcción de sentencias de búsqueda. Esto representa para un usuario, con pocos conocimientos de las sentencias utilizadas, una limitación al momento de ubicar algún tipo de información específica dentro de una base de datos. Por tanto durante el desarrollo del presente Trabajo Especial de Grado se acercó al usuario a un lenguaje nemotécnico de búsquedas

con expresiones escritas prácticamente en lenguaje natural, permitiendo así asociar los objetos creados en sus modelos con la información que desee obtener.

- El sistema implementado brinda al usuario la posibilidad de crear, alterar y gestionar modelos de datos de manera sencilla, amigable y con operaciones asistidas, permitiéndole a este automatizar los procesos necesarios para la conclusión de cada una de las fases del diseño de una base de datos. Mediante las interfaces creadas, el usuario se encuentra asistido en todas las operaciones, reduciendo significativamente la posibilidad de cometer errores durante dicho proceso. Por otra parte el hecho de realizar operaciones de alto nivel mediante el uso de la herramienta, representa un impulso para que el usuario, profundice en las tecnologías existentes y logre automatizar, de manera progresiva, los procesos asociados al desarrollo de aplicaciones.

RECOMENDACIONES

Tomando en cuenta las conclusiones obtenidas, se presentan a continuación una serie de recomendaciones para el futuro incremento de las funcionalidades de la herramienta, así como también posibles mejoras a la misma:

- Realizar mejoras a las interfaces de usuario de la herramienta desarrollada, sería un punto importante en el incremento de la usabilidad y navegabilidad de la misma. Esto podría lograrse mediante estudios acerca de los estilos de presentación mas aceptados por los usuarios hacia los cuales esta dirigida principalmente la aplicación.

- Para incrementar la funcionalidad de la herramienta desarrollada, podrían construirse módulos que permitiesen al usuario la creación de diagramas, no solo de datos sino de otros tipos de notaciones como por ejemplo UML.

- Con el fin de aumentar el alcance de la aplicación desarrollada, podrían incorporarse nuevos archivos con la configuración de la sintaxis de las sentencias de los distintos sistemas manejadores de bases de datos, y así poder brindar al usuario mayor cantidad de opciones al momento de implementar las bases de datos generadas por la herramienta.

- Basándose en la tecnología existente acerca de hojas de estilo XML, podría implementarse el componente de la herramienta referido al espacio de trabajo del usuario (representado visualmente por el componente Jtree de Java del lado izquierdo de la ventana principal de la aplicación), mediante la utilización de dicha tecnología. Esto aumentaría de manera considerable la escalabilidad del componente, tomando en cuenta el posible crecimiento de la herramienta desarrollada.

- *Proveer al usuario con opciones de importación y exportación de diagramas* desde y hacia otras herramientas CASE, representaría una funcionalidad significativa para la portabilidad de los modelos creados mediante el uso de la aplicación.

- Tomando en cuenta las etapas que se deben completar en el diseño de bases de datos, la implementación de procesos de ingeniería reversa específicamente entre las fases lógica y conceptual, representarían un gran aporte a la funcionalidad que provee la herramienta desarrollada.

- Dentro de las funcionalidades que provee la herramienta desarrollada se brindan opciones de manipulación de los objetos pertenecientes a la fase conceptual del diseño de base de datos. Para incrementar el alcance de la aplicación, pudiesen desarrollarse los componentes necesarios para la manipulación de los objetos pertenecientes a la fase lógica del diseño, brindando la creación, modificación y eliminación de dichos objetos.

- Con la finalidad de proveer seguridad para los archivos de cada uno de los usuarios registrados en la herramienta, se podría incorporar algún algoritmo de cifrado simetrico, como DES, triple-DES, etc.

- Basándose en la idea de que la herramienta asista al usuario en cada una de la funciones a realizar. Sería una oportunidad significativa, agregar la posibilidad de crear relaciones M:N a través de un guia que indique al usuario cada uno de los pasos que debe realizar para culminar este proceso de manera correcta.

BIBLIOGRAFÍA

LIBROS

- Elmasri, R. y Navathe S. (1997). Sistemas de bases de Datos (2da. Ed.). México: Addison Wesley.
- Jacobson, I., Booch, G. y Rumbaugh J. (2000). *El proceso unificado de desarrollo del software.* (Trad. S. Sánchez) Madrid: Pearson Educación. (Original en inglés, 1999).
- Kendall y Kendall (1997). *Análisis y diseño de sistemas* (3ra. Ed.). México: Prentice Hall.
- Larman, C. (1999). *UML y patrones. Una introducción al análisis y diseño orientado a objetos y al proceso unificado* (1era. ed.). Madrid: Pearson Educación.
- Pressman R. (2002). *Ingeniería del software: un enfoque práctico* (5ta. Ed.) Madrid: McGraw-Hill.
- Sommerville, I. (2002). *Ingeniería de Software* (6ta. Ed.) México: Pearson Educación.

DOCUMENTOS ELECTRÓNICOS

- Ferré G, X y Sanchez S, M. (2004). *Desarrollo Orientado a Objetos con UML.* [en línea]. España: Univ. Politécnica de Madrid. Disponible en: http://www.clikear.com/manuales/uml/ [2005, Abril].
- Gálvez R, S. y Mora M.,M (2005). *Java a tope: Traductores y compiladores con LEX/YACC/JFLEX/CUP y JAVACC* [en línea]. España: Universidad de Málaga. Disponible en: http://www.lcc.uma.es/~galvez/ [2005, Abril].
- Instituto Nacional de Estadística e Informática (1999). *Herramientas CASE* [en línea]. Disponible en: http://www.pcm.gob.pe/portal_ongei/publicaciones/cultura/lib5103/Libro.pdf [2005, febrero].

- Subdirección de Sistemas-DCCA- UNAM (2004). *Herramientas CASE para el análisis y diseño estructurado* [en línea]. Disponible en: http://sistemas.dgsca.unam.mx/publica/pdf/casestru.pdf [2005, febrero].

ARTÍCULOS ELECTRÓNICOS

- DAEDALUS (2005). *El Enfoque sistémico* [en línea]. Disponible en: http://www.daedalus.es/AreasISEnfoque-E.php [2005, enero].
- García, María I. (2004). *Curso XML* [en línea]. Disponible en: http://geneura.ugr.es/~maribel/xml/introduccion/index.shtml [2005, abril].
- Lania (2000).*Introducción al diccionario de datos de Oracle* [en línea]. Disponible en: http://www.lania.mx/biblioteca/seminarios/basedatos/plsql/diccionario/ [2005, febrero].
- Marqués, María M. (2001). *Ciclo de vida de los sistemas de información* [en línea]. Disponible en: http://www3.uji.es/~mmarques/f47/apun/node66.html [2005, enero].
- Osmosis Latina (2005). *DOM, SAX, JDOM y JAXP* [en línea]. Disponible en: http://www.osmosislatina.com/xml/domsax.htm [2005, abril].
- Rodríguez, M. (2002). *Modelos y metamodelos* [en línea]. Disponible en: http://sensei.ieec.uned.es/~miguel/tesis/node27.html [2004, noviembre].
- Universidad Nacional de Colombia (2004). *Sistema de información* [en línea]. Disponible en: http://www.virtual.unal.edu.co/cursos/enfermeria/2002847/lecciones/tematica4/subtema4-1.html [2005, enero].

PÁGINAS WEB

- Reading XML Data into a DOM 2003. Disponible en: http://java.sun.com/xml/jaxp/dist/1.1/docs/tutorial/dom/1_read.html
- XML DOM - An Introduction to XML Document Object Models 2005. Disponible en: http://xmlfiles.com/dom/dom_intro.asp.
- Xtensible Markup Language. 1999. Disponible en: http://www.programacion.net/html/xml/htmdsssl/capitulo1/capitulo1.htm.

GLOSARIO

A

API (APPLICATION PROGRAMMING INTERFACE): Es una interfaz de programación de aplicaciones y representa un conjunto de especificaciones de comunicación entre componentes software. Representa un método para conseguir abstracción en la programación, generalmente entre los niveles o capas inferiores y los superiores del software.

B

BASE DE DATOS: Es un conjunto de datos que pertenecen al mismo contexto almacenados sistemáticamente para su uso posterior. En este sentido, una biblioteca puede considerarse una base de datos compuesta en su mayoría por documentos y textos impresos en papel e indexados para su consulta.

C

CASE: Acrónimo inglés de Computer Aided Software Engineering, que viene a significar Ingeniería de Software Asistida por Computadora. Son instrumentos o sistemas automatizados que brindan soporte a las actividades de producción de software.

D

DOM: Document Object Model (en inglés, *Modelo de Objetos de Documento*) es una forma de representar documentos estructurados (tales como una página web HTML o un documento XML) que es independiente de cualquier lenguaje orientado a objetos.

I

IDE (*Integrated Development Environment*): Un entorno (o ambiente) integrado de desarrollo, es un programa compuesto por un conjunto de herramientas para un programador. Puede dedicarse en exclusiva a un sólo lenguaje de programación o bien, poder utilizarse para varios.

J

JAVA: Es una plataforma de software desarrollada por Sun Microsystems, de tal manera que los programas creados en ella puedan ejecutarse de la misma forma en diferentes tipos de arquitecturas y dispositivos computacionales. La plataforma consta de cuatro partes fundamentales: el lenguaje de programación, la máquina virtual de que permite la portabilidad en ejecución, El API que representa una biblioteca estándar. El lenguaje mismo se inspira en la sintaxis de C++, pero su funcionamiento es más similar al de Smalltalk que a éste.

P

PARSER: Es un programa especializado que reconoce marcas SGML (Standard Generalized Markup Language o Lenguaje de Marcación Generalizado) en un documento. Un "Parser" puede ser construido en un editor para lenguajes de marcas (XML, HTML) para prevenir errores encontrados en etiquetas y realizar la verificación de todos los elementos que se encuentran en el documento.

S

SGML: Consiste en un sistema para la organización y etiquetado de documentos. El lenguaje **SGML** sirve para especificar las reglas de etiquetado de documentos y no impone en si ningún conjunto de etiquetas en especial.

SISTEMA: Un sistema es un conjunto de elementos organizados que interactúan entre sí y con su ambiente, para lograr objetivos comunes, operando sobre información, sobre energía o materia u organismos para producir como salida información o energía o materia u organismos. Un sistema aislado no intercambia ni materia ni energía con el medio ambiente.

SMBD (SISTEMA MANEJADOR DE BASE DE DATOS): Un Sistema Manejador de base de datos es un conjunto de programas que permiten crear y mantener una Base de datos, asegurando su integridad, confidencialidad y seguridad.

SOFTWARE: conjunto de programas procesados, reglados y documentación; relativos al funcionamiento de un grupo de tratamientos de información.

SQL: El Lenguaje de Consulta Estructurado (Structured Query Language) es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones sobre las mismas. Aúna características del álgebra y el cálculo relacional permitiendo lanzar consultas con el fin de recuperar información de interés de una base de datos, de una forma sencilla.

U

UML (Unified Modeling Language): Es un conjunto de notaciones graficas, soportadas por diagramas, que ayuda en la documentación y diseño de sistemas de software particularmente aquellos construidos bajo el paradigma orientado a objetos. UML entrega una forma de modelar cosas conceptuales como lo son procesos de negocio y funciones de sistema, además de cosas concretas como lo son escribir clases en un lenguaje determinado, esquemas de base de datos y componentes de software reutilizable.

X

XML es el acrónimo del inglés e**X**tensible **M**arkup **L**anguage (lenguaje de marcado ampliable o extensible) desarrollado por el World Wide Web Consortium (W3C). Es una versión simple de SGML. Su objetivo principal es conseguir una página web más semántica. Aunque una de las principales funciones con las que nace sería suceder al HTML, separando la estructura del contenido y permitiendo el desarrollo de vocabularios modulares, compatibles con cierta unidad y simplicidad del lenguaje (objetivo que se viene desarrollando a través de la especificación XHTML), tiene otras aplicaciones entre las que destaca su uso como estándar para el intercambio de datos entre diversas aplicaciones o software con lenguajes privados como en el caso del SOAP.