Universidad de las Ciencias Informáticas
Facultad 1

XCAD, Generador de Capas de Acceso a Datos

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autor: Irving Deither Cao Cabrera.

Tutor: Jorge Landrian García.

Ciudad de la Habana, 20 de junio de 2008

Año 50 de la Revolución

“Yo he preferido hablar de cosas imposibles, porque de lo posible se sabe demasiado.”

Silvio Rodríguez Domínguez.

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

_____ _____

Irving Deither Cao Cabrera Jorge Landrian García

OPINIÓN DEL TUTOR DEL TRABAJO DE DIPLOMA

Título: XCAD, Generador de capas de acceso a datos.

Autor: Irving Deither Cao Cabrera.

Ing. Jorge Landrian García.

Firma	Fecha

AGRADECIMIENTOS

Quiero agradecer en primer lugar a mi tutor, Jorge Landrian García por sus tantas horas de atención. Su dedicación fue decisiva para lograr que este trabajo finalizara en tiempo y con calidad. Darle infinitas gracias por dejarme aprender de él en muchos sentidos, por su paciencia y por ayudarme a pensar de una manera más directa y profesional. Su esfuerzo y saber, fueron y serán incomparables.

A mis padres, mami, hermano, tías y primos, por brindarme lo que necesité en cualquier momento, su apoyo constante impactó en los ánimos para seguir siempre adelante con esta tesis, aún en situaciones difíciles y sin esperanzas.

Gracias a: Alina, Yadier, Dayana, Arnaldo, Adonis, Jenny, Yuliesky, Odeimy, Erik, Reynier, Anié, Yuni, Yilian, Any, Jofman, Dayron (Pa), José Alfredo y a todos mis compañeros del proyecto por la preocupación, conocimientos y los buenos momentos que casi siempre me brindaron.

A mis amigos de aula y apartamento por estos cinco años de tantas cosas: El Barry, Masa, Fabio, Yano, Líber, Dony, Lyon, César, Omar, Pedry, Miche, Yorley, Indira, Yaris.

A mi Universidad por todo lo que puso en mis manos, sobre mi cabeza y bajo mis pies.

DEDICATORIA

A la memoria de mi abuela Josefa.

A mi mamá, papá y hermano.

A mi familia, especialmente a tía Pilpi y Adrián.

A mi hijito Yanko.

RESUMEN

Durante la primera fase del proyecto Identidad, se lograron automatizar los principales procesos de una de las instituciones más importantes de Venezuela, la ONIDEX. Durante el período de implementación de estos procesos, se evidenció la necesidad de algunas herramientas, principalmente para el trabajo con las bases de datos por la gran demanda que tenía para los desarrolladores. Se necesitaba una herramienta masiva que agilizara los cambios que se pudieran necesitar en las capas de acceso a datos, producto de nuevas modelaciones en la base de datos, para permitir que cada desarrollador avanzara en su cronograma de trabajo con mayor eficiencia y rapidez.

Las capas de acceso a datos se generan con el Tier Developer 4.0 (con costo ascendente a los mil dólares y aplicable sólo para productos propietarios). Es una herramienta eficiente aunque no en un 100% si se contempla el factor tiempo.

En este trabajo de diploma se presenta una herramienta con el fin de estandarizar el acceso a datos en el proyecto Identidad y de ser necesario en otros proyectos de la UCI. Con el uso de ésta en las distintas soluciones, se simplificará la cantidad de horas/hombre para generar capas de acceso a datos en el proyecto, logrando mayor productividad en todos los procesos que lo requieran y por otra parte, un ahorro monetario considerable a la hora de adquirir herramientas de este tipo, pues es libre de licencias y pagos.

PALABRAS CLAVES:

ONIDEX: Oficina Nacional de Identificación y Extranjería.

UCI: Universidad de las Ciencias Informáticas.

Tier Developer 4.0: Herramienta utilizada para generar el acceso a datos.

ÍNDICE DE CONTENIDOS

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

La ONIDEX es una institución de mucha importancia en la sociedad venezolana, todos los ciudadanos del país dependen de ella para obtener su identificación, para gestionar sus trámites migratorios o su condición personal dentro del estado venezolano, entre otras cosas. Lograr automatizar la ONIDEX, significaba contar con la eficiencia necesaria para que miles de venezolanos obtuvieran una identidad real. Por otra parte, se lograrían minimizar los problemas en las fronteras terrestres de Venezuela con Brasil y Colombia debidos a ingresos o salidas ilegales de personas vinculadas a delitos que atentan contra la seguridad nacional.

Esta institución carecía de tecnología en el desarrollo de sus procesos, no funcionaba como debía, en ocasiones por el mal diseño en su flujo de trabajo y en otras por la corrupción dentro de su personal. Era inminente informatizar los complejos servicios que presta la ONIDEX.

Toda gran solución es parte de un proceso iterativo e incremental, por esta razón en muchas ocasiones durante el desarrollo del sistema los desarrolladores necesitaban crear, actualizar, borrar algunos de sus procedimientos almacenados o modificar las tablas de sus modelos de datos. Todos estos cambios en la base de datos, obligaban a modificar las capas de acceso a datos que se utilizaban en las soluciones, con la mayor rapidez posible para no frenar el constante desarrollo de la solución general.

La herramienta que se utiliza (Tier Developer 4.0) para crear el acceso a datos en el proyecto no es considerada eficiente si la demanda de cambios necesarios es alta, pues es costosa en tiempo. Varios miembros del proyecto implementaron plantillas para adaptar esta herramienta porque no era 100% factible. En la UCI sólo se contaba con una licencia genuina, válida para una sola computadora, ubicada en uno de sus laboratorios de producción. Mientras se estuvo trabajando en Venezuela fueron utilizadas dos licencias “especiales” (de 30 días cada una), para que la herramienta funcionara en varias computadoras. Cada 60 días (cuando vencían las dos) era necesario formatear el equipo (es obligatorio hacerlo de esta manera porque no es posible reconfigurar la herramienta), y proceder a su completa

reinstalación. Muchas veces estas licencias vencían en medio de un día de trabajo, lo cual demoraba considerablemente la labor de varios desarrolladores.

Para la segunda fase de Identidad, existe la posibilidad de una migración a otro lenguaje de programación. Es conocido que el Tier Developer4.0 no genera capas de acceso a datos para lenguajes de programación libres. Esto implicaría la necesidad de comprar una herramienta (por segunda vez) que genere acceso a datos para lenguajes libres.

Partiendo de estos acontecimientos se define una **Situación Problémica**, para la cual es necesario dar una solución: El desarrollo de la fase dos demanda cambios necesarios en el mecanismo de generar las capas de acceso a datos. Por esta razón, surge la necesidad de crear una herramienta común que permita el acceso a datos con distintos tipos de patrones de acceso a datos y para distintos lenguajes de programación.

De aquí la interrogante en la que nos enfocamos para plantear el **Problema Científico**.

¿Cómo estandarizar el acceso a datos de manera que aumente la productividad en el desarrollo dentro del proyecto Identidad?

Para dar solución al problema existente se ha tomado como **Objetivo general**:

- Desarrollar una herramienta que permita incorporar patrones de acceso a datos y lenguajes de programación en plantillas.

Y, **específicamente los objetivos**:

- Realizar el análisis, diseño e implementación de un generador de capas de acceso a datos, denominado XCAD.
- Garantizar que el modelo de negocio sea independiente del lenguaje en el que se desarrolla para posibilitar que la herramienta sea independiente del producto para el cual se desarrollan las capas de acceso a datos.

Para dar solución a la interrogante tenemos la siguiente **Hipótesis**:

Con la implementación de una herramienta que genere (de forma masiva) capas de acceso a datos, sin depender del lenguaje de programación que utilicen los desarrolladores, se

logrará estandarizar el proceso de generación de capas de acceso a datos, disminuyendo las horas/hombre necesarias para este rol y de esta manera, aumentar la productividad.

Métodos Investigativos
Métodos Empleados:

Métodos Teóricos: Hipotético-deductivo, Análisis Histórico-lógico.

- **Histórico-Lógico:** Este método posibilita el análisis histórico del proceso de gestión de información. Se emplea durante el proceso de diseño y mejoramiento de las funcionalidades que satisfacen o podrían satisfacer las demandas de la herramienta.
- **Hipotético-Deductivo:** Se emplea partiendo de la hipótesis, siguiendo la lógica de deducción obtenida para arribar a nuevos conocimientos y predicciones. Los resultados se someten a verificaciones. Este método se emplea en cada fase en que se obtenga un resultado o se analice sobre la base de algún conocimiento.

Métodos empíricos: Describen y explican las características fenomenológicas del objeto. Representan un nivel de la investigación cuyo contenido procede de la experiencia y se somete a cierta elaboración racional. Este tipo de método es empleado durante todo el proceso de implementación de la herramienta.

Observación: Se emplea en todo momento. Con este método se analiza cada fase del proceso y se toma experiencia de cada tarea, para aplicarse en otras.

Variables:

- **Independiente**: Herramienta que genere capas de acceso a datos, sin depender del lenguaje de programación que se utilice por los desarrolladores.
- **Dependiente**: Disminuir las horas/hombre necesarias para generar capas de acceso a datos. Aumentar la productividad.

Operacionalización de las Variables:

Variable Conceptual	Dimensión	Indicadores	Unidad de medida
Herramienta que genere capas de acceso a datos, sin depender del lenguaje de programación que se utilice por los desarrolladores	Factibilidad	Tiempo de desarrollo	▪ Extenso
			▪ Moderado
			▪ Breve
		Costo	▪ Costoso
			▪ Moderado
			▪ Barato
		Esfuerzo	▪ Alto
			▪ Moderado
			▪ Despreciable
Disminuir las horas/hombre necesarias para generar capas de acceso a datos. Aumentar la productividad.	Mejor rendimiento	Complejidad	▪ Alta
			▪ Media
			▪ Baja
		Control	▪ Bueno
			▪ Deficiente
		Organización del trabajo	▪ Bueno
			▪ Deficiente
		Importancia	▪ Alta
			▪ Media
			▪ Baja
		Tiempo de ejecución	▪ Rápido
			▪ Moderado
			▪ Lento

Los capítulos que conforman este documento quedan expuestos a continuación:

Capítulo 1: Fundamentación Teórica: Este capítulo muestra el estudio de herramientas que implementan su administración, además se hace referencia a las tendencias y tecnologías actuales que se utilizaron.

Capítulo 2: Modelo de Negocio: Se muestran los principales conceptos a través de clases, objetos, sus relaciones y una descripción de cada una de ellos.

Capítulo 3: Características de la herramienta: Se muestran los requerimientos de la herramienta, los diagramas de clases del análisis y del diseño para cada caso de uso del sistema, junto con los correspondientes diagramas de interacción.

Capítulo 4: Análisis y Diseño de la herramienta: Se muestran los diagramas de clases del análisis y del diseño para los casos de uso del sistema, junto con los correspondientes diagramas de interacción.

Capítulo 5: Implementación: Se muestra el modelo de implementación, y las técnicas usadas durante la implementación además del diagrama de despliegue.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

En este capítulo se hace un estudio de algunas herramientas que manejan conceptos y funcionalidades similares a la herramienta XCAD (propuesta para el trabajo con generación de código y acceso a datos) aunque carecen de otras características que este debe tener en cuenta. Además, se exponen las tecnologías usadas para el desarrollo del proyecto.

1.2 Análisis de otras soluciones existentes

En el estudio realizado sobre las herramientas que se utilizan a nivel mundial para generar acceso a datos, se analizaron características favorables o no, de las más notables. En su gran mayoría vienen diseñadas para un solo lenguaje de programación o son para trabajar en web, lo que limita el alcance de la herramienta y la posibilidad de crear versiones del producto ajustadas a otros tipos de requerimientos del negocio. También vale señalar que muchas, aunque no siempre sean las más usadas, poseen licencia comercial. Algunas de ellas se analizan a continuación:

Tier Developer 4.0 (Versión utilizada en el proyecto Identidad)

Esta herramienta de mapeo O/R (Object-to-Relational) es muy completa. Genera código para .NET, el patrón de persistencia que utiliza maneja las tablas que mapea de la BD (Base de datos) como objetos. Tiene la desventaja de poseer licencia comercial y aunque genera código obteniendo datos desde varios gestores de BD, sólo lo hace en dos lenguajes de programación: C#.NET y VB.NET. Esta versión genera el acceso a datos para cuatro fuentes de datos: Oracle, SQL Server, IBM DB2 (DB2 era el antiguo nombre del DBMS [Database Management System], muchos consideran que fue uno de los primeros productos en usar SQL) y Microsoft Access. Es limitada, porque no es aplicable para algunos gestores de BD y lenguajes libres. Actualmente es la herramienta que se utiliza dentro del proyecto, con previo pago (ascendente a mil (1000) dólares americanos) por su licencia.

Hibernate

Hibernate construye el diseño O/R (Objeto-Relacional) mapeando la capa a partir de su definición en un fichero XML. Hibernate es la infraestructura total de la capa de acceso a datos, pero sólo una pequeña porción de la infraestructura es el generador. A pesar de tener gran capacidad para abarcar las necesidades de sus usuarios, esta herramienta está diseñada para software libre, no incluye integración con la plataforma de .NET por lo que puede llegar a ser descartada en algunos proyectos que se desarrollen sobre .NET. Es una herramienta libre. (1)

NHibernate

Este software es considerado un ORM (Object/Relational/Mapping). Es una herramienta libre muy poderosa. Fue la conversión de Hibernate de lenguaje Java a C# posibilitando la integración de Hibernate con la plataforma .NET. El código generado en NHibernate también es válido en Mono. Se considera la más factible de todas por una serie de características claves, que amplían su versatilidad, algunas de ellas, se presentan a continuación: (2)

- Soporta lenguajes de programación orientada a objeto; herencia, polimorfismo, composición y las colecciones de los frameworks de .NET, incluyendo las colecciones genéricas.
- Los API de NHibernate usan los idiomas y conversiones de .NET.
- Posee una rica variedad de mapeos para colecciones de objetos dependientes**.**
- No hay generación de código adicional o pasos extras de procesamiento en el proceso de compilación.
- Personalización de SQL Especifica el SQL exacto que NHibernate debe usar para persistir los objetos.
- Es un software libre. NHibernate está certificado bajo el LGPL (Lesser GNU Public License)

FireStorm/DAO

FireStorm/DAO puede importar definiciones de esquemas de la base de datos a partir de scripts SQL o de la misma base de datos vía JDBC (Java Database Conection) y puede entonces generar el código persistiendo lo que esté basado en el diseño del patrón del DAO (Objeto de acceso a datos). El código generado es bien diseñado, consistente y está comentariado. Esta herramienta es limitada, pues no genera código para .NET, genera para las siguientes tecnologías: (3)

- Java Data Objects (JDO).
- Enterprise JavaBeans (EJB).
- Hibernate (Novedosa en Firestorm/DAO Release 3.0).

1.3 ¿Por qué la necesidad de una herramienta para generar acceso a datos en el proyecto Identidad?

Algunas de las herramientas descritas anteriormente, proporcionan un conjunto de funcionalidades que pudieran ser aplicables al proyecto, si este se ajustase a ellas, pero la necesidad real es adaptar el producto que generan, a nuestro trabajo, en una proporción mayor para lograr ser competentes ante la demanda de trabajo. También existía la necesidad de variar los lenguajes para los que se genera acceso a datos, con el fin de dar cumplimiento a nuevos requerimientos similares, en diferentes lenguajes de programación. Partiendo de estos conceptos y precondiciones es que se identifica la necesidad de una herramienta de acceso a datos, extensible a otros lenguajes de programación, portable en varios ordenadores simultáneamente, libre y con un patrón de persistencia diferente, que permita reconocer los mismos o nuevos requerimientos modelados desde otra perspectiva.

1.4 Tendencias tecnológicas

Siempre que se presente un reto en el mundo de la informática, todo creador debe plantearse un método de solución y seleccionar las herramientas que puedan simplificar el trabajo. Es casi seguro que la complicidad de la solución y su costo en tiempo de desarrollo disminuyan. Por tal razón se consideraron las siguientes tendencias tecnológicas:

UML como lenguaje de modelación visual

El **Lenguaje Unificado de Modelado** (en inglés *Unified Modeling Language o UML*) es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad; está apoyado en gran manera por el OMG (Object Management Group). Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema de software. UML ofrece un estándar para describir un software o parte de él a través de modelos gráficos, incluyendo aspectos conceptuales tales como procesos de negocios y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes de software reutilizables. (4)

RUP como metodología de desarrollo

El **Proceso Racional Unificado** (**RUP**, por sus siglas en inglés *Rational Unified Process*) es un proceso de desarrollo de software y junto con el Lenguaje Unificado de Modelado UML, constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos. RUP es en realidad un refinamiento realizado por Rational Software del más genérico Proceso Unificado. Proporciona una guía en el orden de las actividades de un equipo, dirige las tareas individuales de los desarrolladores, especifica qué productos deberían ser desarrollados y ofrece criterios para monitorear y medir los productos y actividades del proyecto. (5)

Sus principales características son:

- Organización en asignar tareas y responsabilidades (quién hace qué, cuándo y cómo).

- Pretende implementar las mejores prácticas en Ingeniería de Software:
  - Desarrollo iterativo.
  - Administración de requisitos.
  - Uso de arquitectura basada en componentes.
  - Control de cambios.
  - Modelado visual del software.
  - Verificación de la calidad del software.

UModel 2007 de Altova

Es el punto de entrada para el desarrollo de software exitoso. Permite crear e interpretar diseños de software mediante la potencia del estándar UML 2.1. Dibuja el diseño de la aplicación y genera código Java o C# a partir de sus planos. Es una herramienta que permite realizar la ingeniería inversa de programas existentes, a diagramas UML claros y precisos para abarcar rápidamente la arquitectura. Además puede corregir el código generado sincronizando con el modelo y viceversa.

Plataforma .NET

Microsoft.NET es el conjunto de nuevas tecnologías en las que Microsoft ha estado trabajando durante los últimos años con el objetivo de obtener una plataforma sencilla y potente para distribuir el software en forma de servicios que puedan ser suministrados remotamente y que puedan comunicarse y combinarse unos con otros de manera totalmente independiente de la plataforma, lenguaje de programación y modelo de componentes con los que hayan sido desarrollados.

El Framework de .Net es una infraestructura sobre la que se reúne todo un conjunto de lenguajes y servicios que simplifican enormemente el desarrollo de aplicaciones. Mediante esta herramienta se ofrece un entorno de ejecución altamente distribuido, que permite crear aplicaciones robustas y escalables. Organiza toda la funcionalidad del sistema operativo en un

espacio de nombres jerárquico de forma que a la hora de programar resulta bastante sencillo encontrar lo que se necesita. (6)

Posee un conjunto de ventajas entre las que se destacan:

- *Código administrado*: El Tiempo de Ejecución del Lenguaje Común (CLR, por sus siglas en inglés Common Language Runtime) realiza un control automático del código para que este sea seguro, es decir, controla los recursos del sistema para que la aplicación se ejecute correctamente.
- *Interoperabilidad multilenguaje*: El código puede ser escrito en cualquier lenguaje compatible con .Net ya que siempre se compila en código intermedio o Microsoft Intermediate Lenguaje (MSIL).
- *Compilación just-in-time*: El compilador JIT (Just In Time, nombre que recibe ese tipo de compilación porque se realiza en tiempo de ejecución) incluido en el Framework compila el código intermedio (MSIL) generando el código máquina propio de la plataforma. Se aumenta así el rendimiento de la aplicación al ser específico para cada plataforma.
- *Despliegue:* Por medio de los ensamblados resulta mucho más fácil el desarrollo de aplicaciones distribuidas y el mantenimiento de las mismas. El Framework realiza esta tarea de forma automática mejorando el rendimiento y asegurando el funcionamiento correcto de todas las aplicaciones.

Lenguaje de programación C#

Aunque para la plataforma .NET es prácticamente posible programar en cualquier lenguaje, el C# es el lenguaje de propósito general diseñado por Microsoft para ser utilizado en ella, por lo que programarla usando C# es mucho más sencillo e intuitivo que hacerlo con cualquiera de los otros. (7)

Entre sus principales características se destacan:

- *Sencillez*: C# elimina muchos elementos que otros lenguajes incluyen y que son innecesarios en .NET. Por ejemplo:
    - El código escrito en C# es auto contenido, lo que significa que no necesita de ficheros adicionales al propio fuente tales como ficheros de cabecera.
    - El tamaño de los tipos de datos básicos es fijo e independiente del compilador, sistema operativo o máquina para quienes se compile, lo que facilita la portabilidad del código.

- *Orientación a componentes*: La propia sintaxis de C# incluye elementos propios del diseño de componentes que otros lenguajes tienen que simular mediante construcciones más o menos complejas. Es decir, la sintaxis de C# permite definir cómodamente propiedades (similares a campos de acceso controlado), eventos (asociación controlada de funciones de respuesta a notificaciones) o atributos (información sobre un tipo o sus miembros).
- *Eficiente*: En principio, en C# todo el código incluye numerosas restricciones para asegurar su seguridad y no permite el uso de punteros. Sin embargo, y a diferencia de Java, en C# es posible saltarse dichas restricciones manipulando objetos a través de punteros. Para ello basta marcar regiones de código como inseguras (modificador unsafe) y podrán usarse en ellas punteros de forma similar a como se hace en C++, lo que puede resultar vital para situaciones donde se necesite una eficiencia y velocidad de procesamiento muy grandes.

Oracle como Sistema Gestor de Base de Datos

De acuerdo a las características del proyecto Identidad, es Oracle, quien responde como gestor a nuestros requerimientos de trabajo. Por tal razón es que XCAD se diseña para Oracle, aunque por su estructura de herramienta extensible fácilmente podría emplearse en otros gestores de bases de datos.

Oracle es considerado uno de los Sistemas Gestores de Bases de Datos más prestigiosos en la esfera mundial, representa una opción factible y potente para la información que se manejará en un centro de datos, para aquellos sistemas que generan un gran volumen de información por presentar y requerir altos rendimientos. Es altamente escalable permitiendo grandes demandas de usuarios y manteniendo una alta capacidad de adaptarse a cambios bruscos de demanda.

El Oracle 10gR2 permite la utilización de los Clúster de Aplicaciones Reales (RAC), con la tecnología de disco compartido. Está diseñada para satisfacer las demandas actuales de la propuesta de arquitectura. Los recursos, servidores, y almacenamiento pueden ser administrados como una entidad única dentro del ambiente del clúster. A medida que se agregan recursos, el Clúster de Aplicaciones Reales puede utilizarlos, esto asegura un costo total de propiedad más bajo, de esta forma no se necesita comprar nuevo hardware con los requerimientos necesarios.

Otras de sus principales características son: su gran capacidad de almacenamiento y de réplica, máxima seguridad, administración simplificada, soporte de transacciones y facilidades en las tareas de recuperación y respaldo. (8)

XML

XML, siglas en inglés de *E**x**tensible **M**arkup **L**anguage* (traducido: “lenguaje de marcas extensible”), es un metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C[1]). Es una simplificación y adaptación del SGML (Standard Generalized

[1] El **World Wide Web Consortium**, abreviado **W3C**, es un consorcio internacional que produce estándares para la World Wide Web. Se las principales tecnologías en las que se basa son: URL (*Uniform Resource Locator*,

Markup Language o Lenguaje estándar genérico por etiquetas) y permite definir la gramática de lenguajes específicos (de la misma manera que HTML es a su vez un lenguaje definido por SGML). Por lo tanto XML no es realmente un lenguaje en particular, sino una manera de definir lenguajes para diferentes necesidades. Algunos de estos lenguajes que usan XML para su definición son XHTML, SVG, MathML.

XML no ha nacido sólo para su aplicación en Internet, sino que se propone como un estándar para el intercambio de información estructurada entre diferentes plataformas. Se puede usar en bases de datos, editores de texto, hojas de cálculo y casi cualquier cosa imaginable.

XML es una tecnología sencilla que tiene a su alrededor otras que la complementan y la hacen mucho más grande, incluso con posibilidades mucho mayores. Tiene un papel muy importante en la actualidad, pues permite la compatibilidad entre sistemas para compartir la información de una manera segura, fiable y fácil.

XSL

XSL (siglas en inglés de Extensible Stylesheet Language, traducido: “lenguaje extensible de hojas de estilo”) es una familia de lenguajes basados en el estándar XML que permite describir cómo la información contenida en un documento XML cualquiera, debe ser transformada o formateada para su presentación en un medio. (9)

XSLT

XSLT o **Transformaciones XSL** es un estándar de la organización **W3C** que presenta una forma de transformar documentos XML en otros e incluso a formatos que no son XML. Las hojas de estilo **XSLT** (aunque el término de hojas de estilo no se aplica sobre la función directa del XSLT) realizan la transformación del documento utilizando una o varias reglas de plantilla: unidas al documento fuente a transformar, esas reglas de plantilla alimentan a un

Localizador Uniforme de Recursos), HTTP (*HyperText Transfer Protocol*, Protocolo de Transferencia de HiperTexto) y HTML (Lenguaje de Marcado de HiperTexto).

procesador de **XSLT**, el cual realiza las transformaciones deseadas colocando el resultado en un archivo de salida o, como en el caso de una página web, directamente en un dispositivo de presentación, como el monitor de un usuario.

Actualmente, **XSLT** es muy usado en la edición web, generando páginas HTML o XHTML. La unión de XML y **XSLT** permite separar contenido y presentación, aumentando así la productividad. (10)

XSLT es usado para transformar un documento XML en otro documento XML u otro tipo de documento (como HTML o XHTML) que sea reconocido por un explorador. XSLT normalmente hace eso transformando cada elemento XML en un elemento XHTML.

Con XSLT se puede adicionar o quitar elementos y atributos desde o hacia el fichero de salida. Se pueden reagrupar y ordenar elementos, personalizar pruebas y tomar decisiones acerca de cada elemento que se mostrará o no. Para esto cuenta con una especie de navegador de página llamado XPath.

Una manera común de describir el proceso de transformación es decir que XSLT transforma un árbol con fuente XML en un árbol con resultado XML.

A través de las transformaciones XSL es que se crean las plantillas del lenguaje para el que se decide generar el acceso a datos, de esta manera al cargar las plantillas el núcleo generador de la herramienta genera en el código la información que lee de las plantillas XML del lenguaje y se obtienen las clases que conforman la solución del acceso a datos.

XPath

XPath:

- Es una sintaxis para definir partes de un documento XML.
- Utiliza expresiones de camino (path expressions) para navegar en los documentos XML.
- Contiene una librería de funciones estándares.

- Es el elemento más importante en XSLT.
- Es un estándar de la W3C.

XPath es un lenguaje para encontrar información en un documento XML, se utiliza para navegar a través de elementos y atributos en un documento XML. Un buen entendimiento de XPath posibilita un uso avanzado de XML.

Expresiones de camino de XPath (Path Expressions)

XPath emplea expresiones de camino para seleccionar nodos o conjuntos de nodos en un documento XML. Estas expresiones de camino son muy parecidas a las expresiones que se ven al trabajar con un sistema tradicional de ficheros de computadora (ej: C:\Archivos de programa\Altova\XMLSpy2007).

Funciones Estándares de XPath

XPath incluye más de 100 funciones integradas. Contiene funciones para cadenas, valores numéricos, comparaciones de fecha y hora, manipulación de nodos, trabajo con secuencias y valores booleanos entre otras.

XMLSpy2007 de Altova

XMLSpy es un editor de XML. Algunas de sus características especiales son:

- Fácil de utilizar.
- Completamiento automático de etiquetas (tags).
- Chequeo automático de buen formato.
- Colores de las sintaxis y buena impresión.
- Validación acoplada de DTD y/o XML Schema.
- Cambio fácil entre las vistas de texto y cuadrículas.

- Editor gráfico de esquemas XML integrado.
- Potentes utilidades de conversión.
- Importaciones y exportaciones de base de datos.
- Plantillas integradas para varios tipos de documentos XML.
- Analizador XPath 1.0/2.0 integrado.
- Editor, profile y debugger de XSLT 1.0/2.0.
- Generación de código en Java, C++, y C#.
- Soporte para Office 2007 / OOXML.

1.5 Conclusiones

Teniendo en cuenta la utilización que tendrá la herramienta XCAD dentro del proyecto Identidad y a su vez, la envergadura de este, se analizaron un conjunto de recientes tecnologías, con el fin de ser aplicadas en el desarrollo de la solución. Además se estudiaron algunas de las herramientas más utilizadas que proporcionan acceso a datos y se justifica la necesidad de crear una nueva, que mejore los aspectos negativos de las estudiadas y mantenga los positivos.

CAPÍTULO 2: MODELO DEL NEGOCIO

2.1 Introducción

En este capítulo se analizan las necesidades de la herramienta especificándolas mediante los requerimientos funcionales y los requerimientos no funcionales. Además, se hace un estudio del negocio en que se enmarca la herramienta y se llega a la conclusión de que sería mejor hacer una modelación del dominio, identificando los conceptos principales que se manejan y las relaciones entre ellos.

2.2 Modelo de dominio

Después de haber hecho el análisis correspondiente, se llegó a la conclusión de que no se modelaría el negocio a través del diagrama de casos de uso del negocio, debido a que no se pudieron identificar procesos de negocios bien claros, sólo elementos conceptuales. Por tal motivo es que se hace el Modelo de Dominio y se representa mediante conceptos relacionados entre sí. Este modelo deja bien claro cómo funciona el entorno en el cual está enmarcada la herramienta. El modelo de dominio representa conceptos del mundo real y para poder identificar los conceptos se hace necesario investigar el dominio del problema. (11)

En el siguiente modelo de dominio se especifican las relaciones que existen entre los principales conceptos que se gestionan en el desarrollo de la herramienta. Los conceptos serán tomados en su mayoría de la librería Lenguaje que funciona como base de XCAD. Dicha librería representa con sus clases y relaciones todos los conceptos que se manejarán al utilizar la herramienta.

2.3 Diagrama de clases del Modelo de Dominio

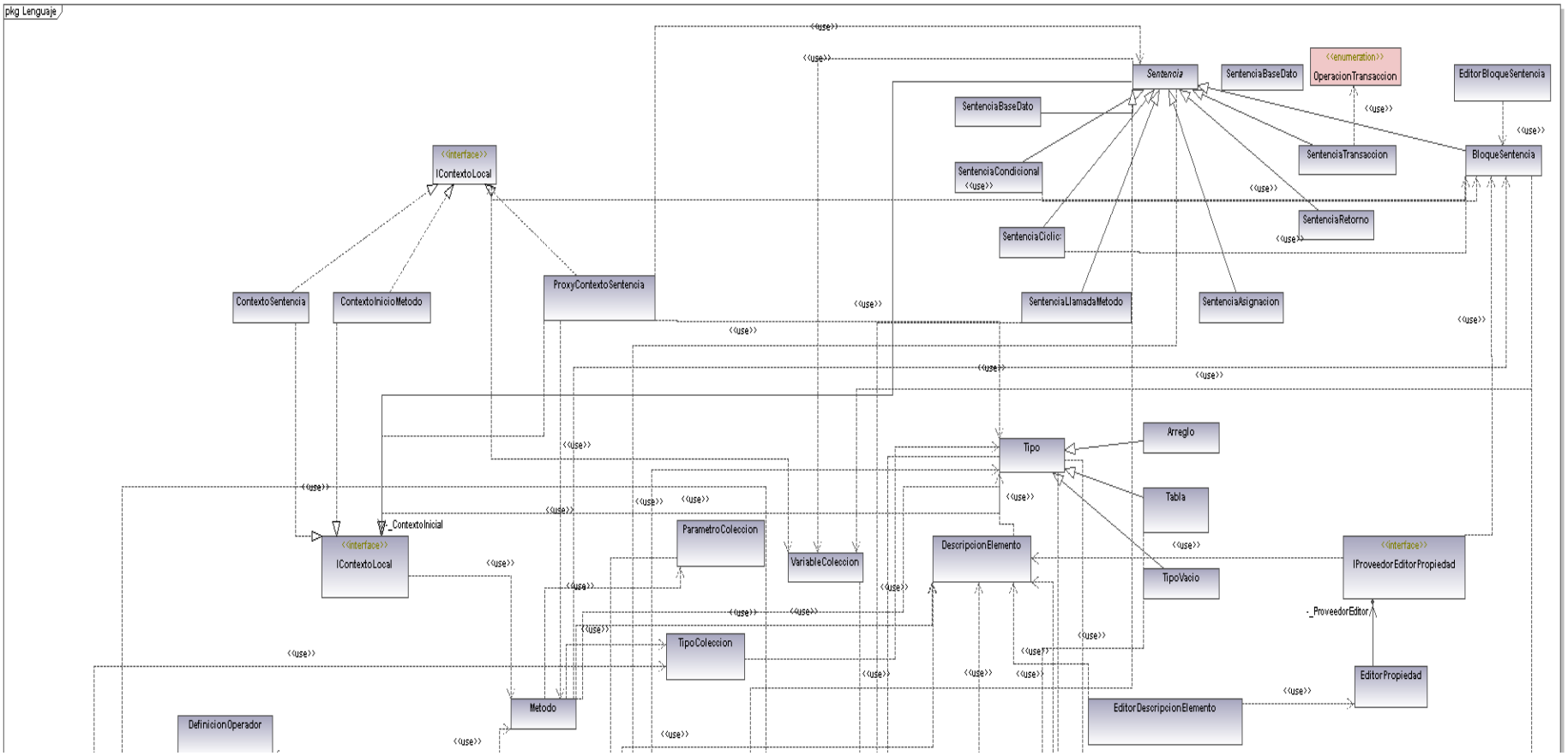

Figura 2.1 Porción superior del Diagrama del Modelo de Dominio.

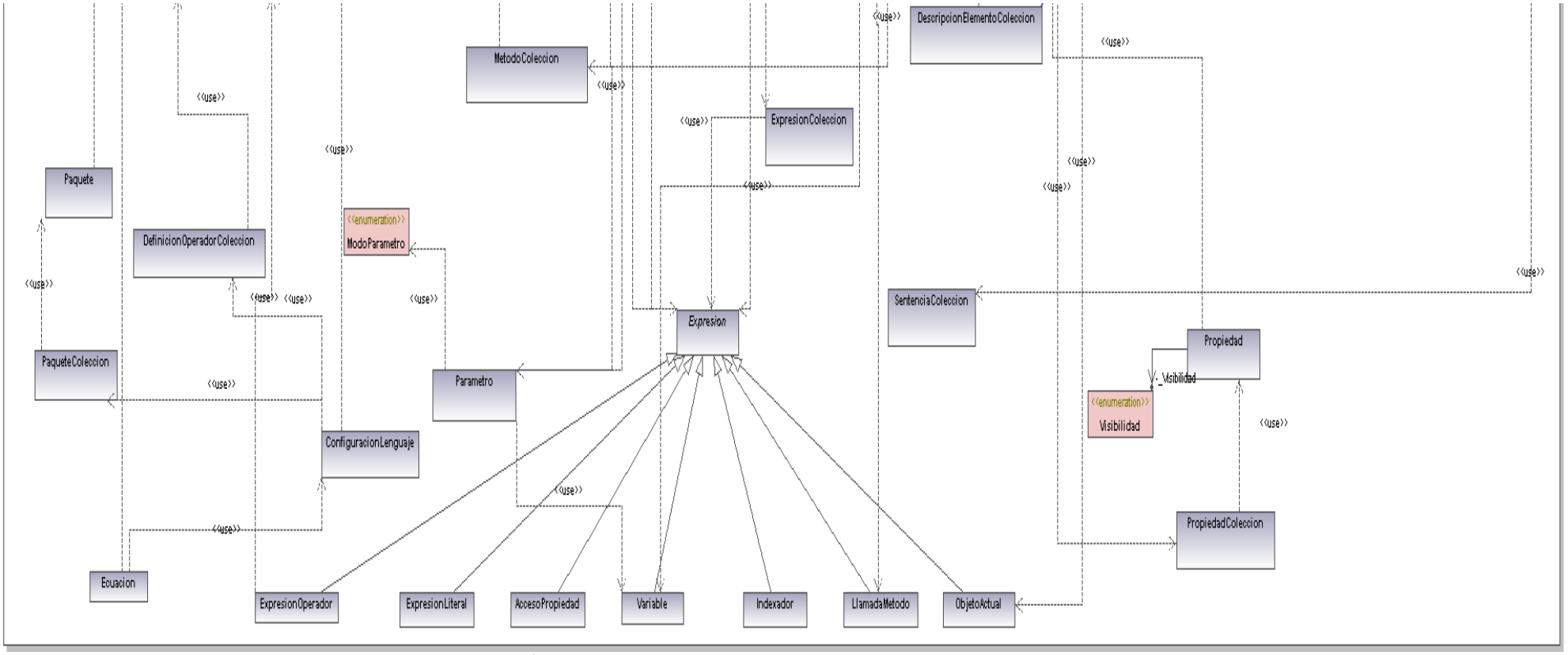

Figura 2.2 Porción inferior del Diagrama del Modelo de Dominio.

2.4 Glosario ordenado alfabéticamente de conceptos del Modelo de Dominio

- **Acceso a Propiedad:** Es un tipo de *Expresión* que tiene el lenguaje, se utiliza para representar el acceso a una *Propiedad* que tenga el objeto para el que se está modelando el acceso a datos.
- **Arreglo:** Es un *Tipo.* Concepto del sistema para representar el tipo de dato “arreglo” de los lenguajes de programación.
- **Bloque de Sentencias:** Es una *Sentencia* o grupo de ellas (véase descripción de *Sentencia*).
- **Configuración del Lenguaje:** Este concepto del sistema permite manejar los operadores, sus definiciones y los paquetes que se controlen.
- **Contexto de Inicio de Método:** Es uno de los conceptos más importantes dentro del dominio de la herramienta, pues se encarga de modelar el espacio donde se desarrolla cada método. Maneja información de *Tipos*, *Métodos* y V*ariables*. Cada contexto de inicio de método es una interfaz *IContextoLocal*.
- **Contexto Sentencia:** El contexto de una sentencia es en síntesis, el ambiente en que se va creando esta. Se deben tener en cuenta los *Tipos, Métodos y Variables* a controlar.
- **Definición de Operador:** Representa las definiciones de todas las propiedades de un operador. Tal y como está definido en varios lenguajes de programación, un operador tiene tipos de operando (izquierdo y derecho), precedencia, su identificador, entre otras propiedades para XCAD, funciona de la misma manera el lenguaje que soporta la herramienta por dentro, y este concepto define ese comportamiento.
- **Descripción Elemento:** Todo modelo se basa en elementos. Sus tipos y nombres son las informaciones que se controlan con este concepto dentro del ambiente de la herramienta.

- **Ecuación:** En un bloque de sentencias, es posible realizar casi cualquier operación de las que normalmente se realizan en cualquier lenguaje de programación. La ecuación es una expresión o varias expresiones, que se manejan dentro de un contexto. Este concepto permite modelar dentro del sistema el conjunto de variables, expresiones y la configuración del lenguaje que se engloban en una ecuación.
- **Editor del Bloque de Sentencias:** Concepto del sistema que se encarga de la edición dentro de un bloque de sentencia.
- **Editor Descripción Elemento:** Concepto del sistema que se encarga de la edición de un elemento.
- **Editor Propiedad:** Mediante este concepto es que se controla la edición a las propiedades. La edición de propiedad contempla los cambios en la descripción del elemento y los bloques de sentencias.
- **Expresión:** Es un concepto del sistema que se especializa en englobar los principales tipos de expresiones que puede tener un lenguaje de programación. Controla el tipo de retorno y las colecciones de expresiones.
- **Expresión Literal:** Es una *Expresión. C*ontrola la información al crearse una expresión literal dentro de un contexto o bloque de sentencias.
- **Expresión Operador:** Al definirse un operador, pueden estar vinculadas varias expresiones. Generalmente un operador contiene elementos a la izquierda y a la derecha, por tanto este concepto del sistema permite crear las expresiones de operadores manejando dos expresiones: Operando Izquierdo y Operando Derecho, y la descripción del operador.
- **Contexto Local:** Esta interfaz básicamente lo que contiene es la información del contexto donde se esté ubicado. Para todo contexto se debe saber el *Tipo*, *Método* y *Variables* del entorno.
- **Indexador:** Concepto del sistema para representar el trabajo con arreglos e índices. Su funcionamiento es similar a los indexadores de los lenguajes de programación.

- **Proveedor Editor Propiedad:** Concepto del sistema que se encarga de proveer la interfaz de edición para las propiedades.
- **Llamada a Método:** Es una *Expresión.* Este concepto del sistema permite trabajar con colecciones de *expresiones* e incorporar un receptor, encargado de almacenar el resultado de la llamada a un método.
- **Método:** Con este concepto se almacena la información que tienen normalmente los métodos de los lenguajes a programación, dígase descripción, parámetros y bloques de sentencias.
- **Objeto Actual:** Es un tipo de *Expresión* y un *Tipo* dentro del sistema.
- **Paquete:** Concepto del sistema que contiene todo lo modelado dentro de la herramienta. Para esto almacena la información de los *Tipos* y las descripciones.
- **Parámetro**: Es un concepto dentro de la herramienta, para representar a las condiciones adicionales específicas que se le pasan al procedimiento almacenado.
- **Propiedad:** Concepto del sistema que representa el comportamiento de una propiedad, con el mismo funcionamiento que en cualquier lenguaje de programación. De las propiedades almacena su visibilidad *(*véase descripción de *Visibilidad)* y descripción (véase *Descripción de Elemento*).
- **Proxy Contexto Sentencia:** Este concepto del sistema permite condicionarle un contexto local (véase descripción de *Contexto Local*) a una sentencia, desde cualquier entorno.
- **Sentencia:** Es un concepto del sistema que funciona similar en los lenguajes de programación. Representa una unidad de la cual heredan varias estructuras sintácticas del lenguaje. Este concepto representa las operaciones que están definidas dentro del lenguaje base de XCAD.
- **Sentencia de Asignación:** Es un tipo de *Sentencia*. Como su nombre sugiere, este concepto del sistema representa esa unidad estructural que se aplica en todo lenguaje al asignar un valor o expresión.
- **Sentencia Cíclica:** Concepto del sistema para representar las estructuras cíclicas dentro del lenguaje definido como base. Es un tipo de *Sentencia*.

- **Sentencia Condicional:** Concepto del sistema para representar las estructuras condicionales dentro del lenguaje definido como base. Es un tipo de *Sentencia*.
- **Sentencia de Llamada a Método:** Concepto del sistema para representar la llamada a un método en una sentencia dentro del lenguaje definido como base. Es un tipo de *Sentencia*.
- **Sentencia de Retorno:** Concepto del sistema para representar el proceso de retorno al ejecutar una sentencia dentro del lenguaje definido como base. Es un tipo de *Sentencia*.
- **Sentencia de Transacción:** Concepto del sistema para representar las transacciones en las operaciones dentro del lenguaje definido como base. Es un tipo de *Sentencia*.
- **Tabla:** Es un *Tipo* dentro del dominio de la herramienta que controla la información de una colección de elementos.
- **Tipo**: Es un concepto dentro de la herramienta que ayuda a englobar los objetos. Todo en XCAD es un Tipo (Conectores, tipos de datos, objetos). Controla colecciones de propiedades, métodos y clasifica los Tipos en simples o no, para operaciones posteriores.
- **Tipo Vacio:** Es un *Tipo* y dentro del dominio de la herramienta que representa los elementos nulos.
- **Variable**: Es un tipo de *Expresión*. Este concepto del sistema, representa el comportamiento de toda variable en un lenguaje de programación. En este concepto se controla el nombre y *Tipo* que se asigna dentro del entorno de la herramienta.

2.5 Especificación de los requerimientos del software

2.5.1 Requerimientos funcionales

Los requerimientos funcionales especifican acciones que el sistema debe ser capaz de realizar, sin tomar en consideración ningún tipo de restricción física. Es decir, especifican el

comportamiento de entrada y salida del sistema y surgen de la razón fundamental de la existencia del producto. (12)

La herramienta debe:

RF-1. Crear nuevo conector.
- 1.1. Nombrar conector.
- 1.2. Adicionar tipos.
  - 1.2.1. Importar tipos desde una librería de enlace dinámico (DLL: Dinamic Link Language).
  - 1.2.2. Seleccionar si es un tipo simple o no.

RF-2. Adicionar Métodos
- 2.1. Nombrar método.
- 2.2. Adicionar bloque de sentencia.
  - 2.2.1. Mostrar los tipos de sentencias.
    - 2.2.1.1. Seleccionar sentencia no de base de datos.
      - 2.2.1.1.1. Construir expresiones.
      - 2.2.1.1.2. Adicionar expresiones.
        - 2.2.1.1.2.1. Seleccionar operadores.
        - 2.2.1.1.2.2. Adicionar variables.
          - 2.2.1.1.2.2.1. Nombrar variable.
          - 2.2.1.1.2.2.2. Asignarle un tipo.
        - 2.2.1.1.2.3. Adicionar propiedades ya existentes.
        - 2.2.1.1.2.4. Adicionar objeto actual.
        - 2.2.1.1.2.5. Adicionar indexador.
          - 2.2.1.1.2.5.1. Asignar valor.
    - 2.2.1.2. Seleccionar sentencia de base de datos.
      - 2.2.1.2.1. Mostrar controles para conexión a la base de datos.
      - 2.2.1.2.2. Configurar proveedor.
        - 2.2.1.2.2.1. Suministrar nombre de la base de datos, usuario y contraseña.

2.2.1.2.2.2. Probar conexión.

2.3. Adicionar parámetros.

2.3.1. Nombrar parámetro.

2.3.2. Seleccionar modo parámetro.

RF-3. Adicionar Propiedades.

3.1. Nombrar propiedad.

3.2. Establecer la visibilidad de la propiedad.

RF-4. Limpiar controles visuales.

RF-5. Eliminar conector.

5.1. Seleccionar conector.

RF-6. Eliminar tipo.

6.1. Seleccionar tipo.

RF-7. Eliminar propiedad.

7.1. Seleccionar propiedad.

RF-8. Eliminar método.

8.1. Seleccionar método.

RF-9. Eliminar parámetro.

9.1. Seleccionar parámetro.

RF-10. Seleccionar la configuración de operadores del lenguaje.

RF-11. Generar capa de acceso a datos en XML

11.1. Nombrar el fichero de salida.

11.2. Crear ubicación.

RF-12. Cargar capa de acceso a datos en XML.

RF-13. Generar acceso a datos en lenguajes de programación

13.1. Seleccionar la plantilla XSLT del lenguaje.

RF-14. Salir de la herramienta.

2.5.2 Requerimientos no funcionales

Los requerimientos no funcionales son las propiedades o cualidades que el producto debe tener para que este sea atractivo, usable, rápido y confiable. (13)

- Apariencia o interfaz externa
  - Interfaces de Usuarios: La interfaz debe ser amigable para el usuario y lo más explícita posible.
- Usabilidad
  - El software debe ser fácil de usar por personas que tengan una mínima experiencia en trabajo con bases de datos y programación del acceso a datos.
- Portabilidad
  - El software está construido con código totalmente portable para la plataforma Mono, de software libre.
- Ayuda y documentación.
  - Entregar documentos técnicos y las guías de usuario, que incluyen:
    - Presentaciones realizadas en cada tema.
  - Manual de usuario.

2.6 Conclusiones

En este capítulo se han relacionado todos los conceptos del modelo de dominio y se ha argumentado por qué la decisión de hacer una modelación del negocio a través de un modelo de dominio. Además de esto se especificaron cada uno de los requerimientos, funcionales y no funcionales necesarios para desarrollar el software.

CAPÍTULO 3: CARACTERÍSTICAS DE LA HERRAMIENTA

3.1 Introducción

En este capítulo se realiza una modelación de la herramienta propuesta. Se analizan sus características a través de casos de usos críticos o arquitectónicamente significativos del sistema, actor que interviene y las descripciones gráficas o textuales detalladas de ellos. Además se especifica el comportamiento de la solución en cada uno de los procesos que se proponen.

3.2 Modelación de la herramienta

3.2.1 Actor del Sistema

Actor del sistema	Descripción
Desarrollador de Software	Es quien utilizará la herramienta de generación en el desarrollo de un proyecto.

Tabla 1. Descripción de los actores del sistema.

3.3.2 Diagrama de Casos de Uso del Sistema

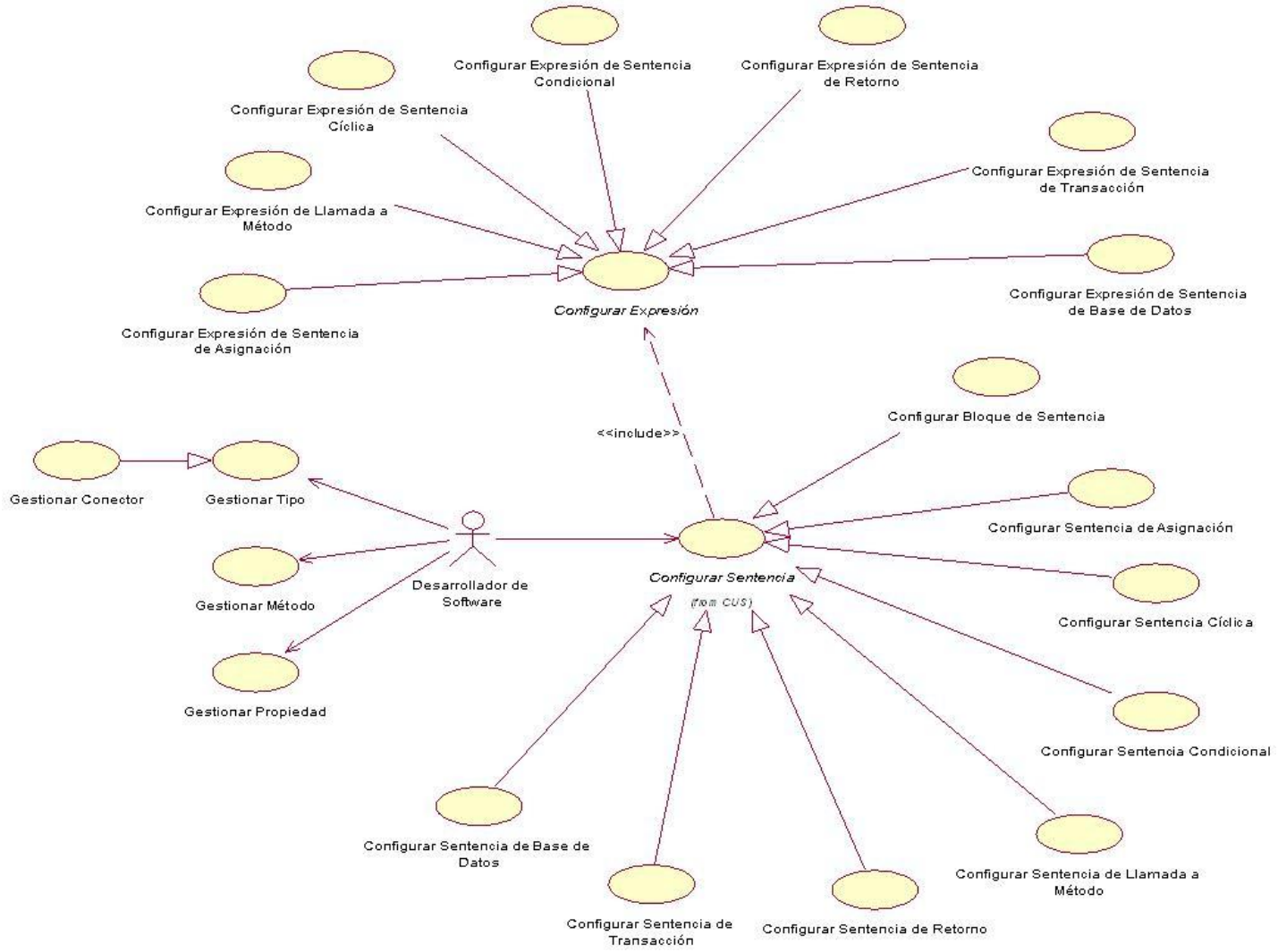

Figura 3.1 Diagrama de casos de uso del sistema.

3.3.3 Descripción de los Casos de Uso del Sistema

Caso de uso:	Gestionar Conector.
Actores:	Desarrollador de software.
Propósito:	Crear, modificar y eliminar conectores.
Resumen:	El caso de uso inicia cuando el desarrollador presiona en el botón Nuevo y crea un conector.
Referencias:	RF-1 (1.1)
Precondiciones:	El desarrollador debe tener la aplicación en ejecución.
Flujo Normal de los Eventos	
Acción del Actor	Respuesta del Sistema
1.- El desarrollador selecciona el botón de la aplicación que permite crear nuevos conectores.	2.- La herramienta muestra la interfaz para crear un nuevo conector y asignarle un nombre.
	3.- Al crear el conector el sistema debe mostrarlo en la lista de los conectores.
Sección 1: Nuevo.	
Acción del Actor	Respuesta del Sistema
3.1.- El desarrollador selecciona el conector.	3.2.- La herramienta muestra el identificador del conector, las opciones para modificar el tipo, su colección de métodos y su colección de propiedades.
Sección 2: Modificar.	
Acción del Actor	Respuesta del Sistema
3.3.- El desarrollador selecciona el conector a modificar.	
3.4.- El desarrollador introduce los datos del conector que desea cambiar (nombre de conector, tipo, método, propiedades).	3.5.- La herramienta actualiza las modificaciones.
Sección 3: Eliminar.	

Acción del Actor	Respuesta del Sistema
3.6.- El desarrollador selecciona el conector que desea eliminar y selecciona el botón Aceptar.	3.7.- La herramienta elimina el conector y toda su información.
	3.8.- La herramienta elimina de la lista de conectores el conector seleccionado.
Poscondiciones:	Sección 1: Se adiciona un conector nuevo. Sección 2: Se modifica un conector. Sección 3: Se elimina un conector.
Prioridad:	Crítico.

Tabla 2. Descripción del caso de uso: Gestionar Conector.

Caso de uso:	Gestionar Tipo.
Actores:	Desarrollador de software.
Propósito:	Crear, modificar y eliminar tipos.
Resumen:	El caso de uso inicia cuando el desarrollador se ubica en la pestaña Tipo que contiene las opciones para crear o importar los Tipos.
Referencias:	RF-1.2 (1.2.1 , 1.2.2)
Precondiciones:	El desarrollador debe tener la aplicación en ejecución.
Flujo Normal de los Eventos	
Acción del Actor	Respuesta del Sistema
1.- El desarrollador selecciona la pestaña Tipo, para ubicarse en el entorno de Tipo.	2.- La herramienta muestra la interfaz para crear un nuevo Tipo y asignarle un nombre.
2.1.- El desarrollador nombra el nuevo Tipo y selecciona el botón Aceptar	3.- La ventana para nombrar el Tipo es la misma que para crear los conectores porque los conectores son Tipos también. Se adiciona el nuevo Tipo a la lista de tipos.

3.1. El desarrollador puede importar los tipos desde una librería seleccionando el botón Importar.	3.2. – La herramienta contiene un botón Importar que abre una ventada de diálogo con el usuario permitiéndole moverse por Windows en busca de la librería a importar.
3.3.- El desarrollador selecciona la librería de tipos a importar.	3.4.- La herramienta carga todos los Tipos que tiene la librería y permite marcar los deseados.
3.5.- El desarrollador selecciona los Tipos a importar y presiona el botón Aceptar.	3.5.- La herramienta muestra en la lista de Tipos, los nuevos importados por el desarrollador.
Sección 2: Modificar.	
Acción del Actor	Respuesta del Sistema
3.6.- El desarrollador selecciona el Tipo a modificar.	3.7.- Ver descripción del caso de uso: Gestionar Conector.
Sección 3: Eliminar.	
Acción del Actor	Respuesta del Sistema
3.8.- El desarrollador selecciona el Tipo que desea eliminar y selecciona el botón Aceptar.	3.9.- Ver descripción del caso de uso: Gestionar Conector.
Poscondiciones:	Sección 1: Se adiciona un tipo nuevo. Sección 2: Se modifica un tipo. Sección 3: Se elimina un tipo.
Prioridad:	Crítico.

Tabla 3. Descripción del caso de uso: Gestionar Tipo.

Caso de uso:	Gestionar Método.
Actores:	Desarrollador de software.
Propósito:	Crear, modificar y eliminar métodos.
Resumen:	El caso de uso inicia cuando el desarrollador selecciona el botón

	para mostrar la colección de métodos de un conector o tipo.
Referencias:	RF-2(2.1, 2.3, 2.3.2)
Precondiciones:	El desarrollador debe haber creado algún conector o tipo.
Flujo Normal de los Eventos	
Acción del Actor	Respuesta del Sistema
1.- El desarrollador selecciona el botón de la aplicación que permite ubicarse en el entorno de Método.	2.- La herramienta muestra la interfaz para adicionar un nuevo método (Nombre, Tipo, Parámetros).
Sección 1: Adicionar.	
Acción del Actor	Respuesta del Sistema
3.- El desarrollador selecciona el botón Adicionar.	3.1.- La herramienta muestra la interfaz para crear Método.
3.2.- El desarrollador edita el nombre, puede seleccionar la lista de parámetros o editar el bloque de sentencias que tendrá el método.	3.3.- La herramienta crea un método con el nombre suministrado y lo adiciona a la lista de métodos.
3.4.- El desarrollador selecciona el botón Adicionar.	3.5.- La herramienta lo ubica en el entorno de Parámetros, donde puede adicionar o quitar los que ya estén. Además le permite editar el Modo del parámetro (Entrada, salida o por referencia.)
Sección 2: Modificar.	
Acción del Actor	Respuesta del Sistema
3.6.- El desarrollador selecciona el método a modificar.	3.7.- La herramienta muestra los elementos que contiene el método.
3.8.- El desarrollador introduce los datos del método que desea cambiar (nombre de método, parámetros, bloque de sentencias).	3.9.- Si es bloque de sentencia, ver Descripción del caso de uso: Configurar Sentencia (**Anexo 1**). 4.- Si es el nombre, se muestra una ventana para editarlo.

	4.1.- Si es parámetro, se muestra la colección de parámetros.
4.2.- El desarrollador selecciona el parámetro a modificar.	4.3.- La herramienta muestra los elementos del parámetro (Descripción y Modo).
4.4.- El desarrollador selecciona el botón Aceptar.	4.5.- La herramienta aplica los cambios.
Sección 3: Eliminar.	
Acción del Actor	Respuesta del Sistema
4.6.- El desarrollador selecciona el método que desea eliminar y presiona el botón Aceptar.	4.7.- La herramienta elimina el método y toda su información.
	4.8.- La herramienta elimina de la lista de métodos el método seleccionado.
Poscondiciones:	Sección 1: Se adiciona un método nuevo. Sección 2: Se modifica un método. Sección 3: Se elimina un método.
Prioridad:	Crítico.

Tabla 4. Descripción del caso de uso: Gestionar Método.

Caso de uso:	Gestionar Propiedad.
Actores:	Desarrollador de software.
Propósito:	Crear, modificar y eliminar propiedades.
Resumen:	El caso de uso inicia cuando el desarrollador presiona el botón para mostrar la colección de propiedades de un conector o tipo.
Referencias:	RF-2(2.1, 2.3, 2.3.2)
Precondiciones:	El desarrollador debe haber creado algún conector o tipo.
Flujo Normal de los Eventos	
Acción del Actor	Respuesta del Sistema
1.- El desarrollador selecciona el botón de la aplicación que permite ubicarse en el	2.- La herramienta muestra la interfaz para adicionar una nueva propiedad

entorno de Propiedad.	(Descripción, Visibilidad).
Sección 1: Adicionar.	
Acción del Actor	Respuesta del Sistema
3.- El desarrollador presiona en el botón Adicionar.	3.1.- La herramienta muestra la interfaz para crear la propiedad.
3.2.- El desarrollador teclea el nombre y/o la visibilidad.	3.3.- La herramienta crea una propiedad con el nombre y/o visibilidad suministrados y la adiciona a la lista de propiedades.
Sección 2: Modificar.	
Acción del Actor	Respuesta del Sistema
3.6.- El desarrollador selecciona la propiedad a modificar.	3.7.- La herramienta muestra los elementos que contiene la propiedad.
3.8.- El desarrollador introduce los datos de la propiedad que desea cambiar (nombre de la propiedad, visibilidad: pública, protegida, privada).	3.9.- Se muestra una ventana en dependencia de la opción a editar.
4.- El desarrollador selecciona el botón Aceptar.	4.1.- La herramienta aplica los cambios.
Sección 3: Eliminar.	
Acción del Actor	Respuesta del Sistema
4.6.- El desarrollador selecciona la propiedad que desea eliminar y presiona en Aceptar.	4.7.- La herramienta elimina la propiedad y toda su información.
	4.8.- La herramienta elimina de la lista de propiedades la propiedad seleccionada.
Poscondiciones:	Sección 1: Se adiciona una propiedad nueva. Sección 2: Se modifica una propiedad. Sección 3: Se elimina una propiedad.
Prioridad:	Crítico.

Tabla 5. Descripción del caso de uso: Gestionar Propiedad.

Caso de uso:	Configurar Sentencia.
Actores:	Desarrollador de software.
Propósito:	Configurar cualquiera de las sentencias que seleccione el actor.
Resumen:	El caso de uso inicia cuando el desarrollador presiona en el botón Bloque de Sentencias y levanta el Editor del Bloque de Sentencias.
Referencias:	RF-2.2(2.2.1, 2.2.1.1, 2.2.1.1.1, 2.2.1.1.2, 2.2.1.1.2.1, 2.2.1.1.2.2, 2.2.1.1.2.2.1, 2.2.1.1.2.2.2, 2.2.1.1.2.2.3, 2.2.1.1.2.2.4, 2.2.1.1.2.2.5, 2.2.1.1.2.2.5.1)
Precondiciones:	El desarrollador debe seleccionar el método para el que va a crear, modificar o eliminar la sentencia.
Flujo Normal de los Eventos	
Acción del Actor	Respuesta del Sistema
1.- El desarrollador selecciona el botón de la aplicación que permite crear un bloque de sentencia.	2.- La herramienta muestra la interfaz del Editor del Bloque de Sentencia.
3.- El desarrollador selecciona el botón Adicionar(A).	3.1.- La herramienta debe mostrar los tipos de sentencias.
3.2.- El desarrollador debe seleccionar la sentencia que desea adicionar.	3.3.- Si la sentencia es de Asignación, ver Descripción del caso de uso: Configurar Sentencia de Asignación (**Anexo 1**). 3.4.- Si la sentencia es Condicional, ver Descripción del caso de uso: Configurar Sentencia Condicional (**Anexo 1**). 3.5.- Si la sentencia es Cíclica, ver Descripción del caso de uso: Configurar Sentencia Cíclica (**Anexo 1**). 3.6.- Si la sentencia es Llamada a Método, ver Descripción del caso de uso:

	Configurar Sentencia de Llamada a Método (**Anexo 1**). 3.7.- Si la sentencia es de Retorno, ver Descripción del caso de uso: Configurar Sentencia de Retorno (**Anexo 1**). 3.8.- Si la sentencia es de Transacción, ver Descripción del caso de uso: Configurar Sentencia de Transacción (**Anexo 1**). 3.9.- Si la sentencia es de Bases de datos, ver Descripción del caso de uso: Configurar Sentencia de Bases de datos. (**Anexo 1**).
Sección 1: Adicionar Sentencia.	
Acción del Actor	Respuesta del Sistema
4.- El desarrollador presiona en el botón Aceptar.	4.1.- La herramienta muestra en el Editor del bloque de sentencias, la sentencia seleccionada lista para completarse.
Sección 2: Insertar Sentencia.	
Acción del Actor	Respuesta del Sistema
4.2.- El desarrollador selecciona la sentencia a desplazar.	4.2.1- La herramienta le cambia el color a la sentencia donde se sitúe el desarrollador.
4.3.- El desarrollador selecciona el botón Insertar(I)	4.3.1.- La herramienta muestra las sentencias que puede insertar.
4.4.- El desarrollador selecciona la nueva sentencia y presiona en el botón Aceptar.	4.5.- La herramienta desplaza en el Editor, la Sentencia sombreada, y pone en su lugar la nueva que seleccionó para insertar.
Sección 3: Modificar Sentencia	
Acción del Actor	Respuesta del Sistema

4.5.1.- El desarrollador selecciona la sentencia a modificar.	4.5.2.- La herramienta sombrea la sentencia seleccionada.
4.5.3.- El desarrollador presiona en la sección de la sentencia.	4.5.4.- La herramienta muestra las opciones para modificar.
4.5.5.- El desarrollador modifica la sentencia.	4.5.6.- La herramienta aplica y muestra los cambios.
Sección 4: Eliminar Sentencia.	
Acción del Actor	Respuesta del Sistema
4.6.- El desarrollador selecciona la sentencia en el Editor del bloque de sentencias.	4.7.- La herramienta sombrea de un color distinto la sentencia a eliminar.
4.7.- El desarrollador da clic Derecho.	4.8.- La herramienta muestra la opción Limpiar.
4.9.-El desarrollador selecciona la opción Limpiar.	5.- La herramienta elimina la sentencia del bloque de sentencias.
Poscondiciones:	Sección 1: Se adiciona una nueva sentencia. Sección 2: Se inserta una nueva sentencia. Sección 3: Se modifica una sentencia. Sección 4: Se elimina una sentencia.
Puntos de extensión:	Punto de extensión 1: Caso de uso: Configurar Expresión.
Prioridad:	Crítico.

Tabla 6. Descripción del caso de uso: Configurar Sentencia.

Caso de uso:	Configurar Expresión.
Actores:	Desarrollador de software.
Propósito:	Configurar cualquiera de las expresiones que seleccione el actor dentro de las sentencias.
Resumen:	El caso de uso inicia cuando el desarrollador da clic derecho en el Bloque de Sentencias y se muestran las expresiones

	correspondientes al cuerpo de la sentencia en que se esté desarrollando.
Referencias:	RF-2.2.1.1.1, RF- 2.2.1.1.2
Precondiciones:	El desarrollador debe seleccionar el método para el que va a crear, modificar o eliminar la sentencia.

Flujo Normal de los Eventos	
Acción del Actor	Respuesta del Sistema
1.- El desarrollador da clic derecho en el cuerpo de la sentencia.	2.- La herramienta muestra las opciones que corresponden al tipo de sentencia en que esté desarrollando el actor.
3.- El desarrollador selecciona la expresión.	3.1.- Si es una expresión de Sentencia de Asignación, ver Descripción del caso de uso: Configurar Expresión de Sentencia de asignación (**Anexo 1**). 3.2.- Si es una expresión de Sentencia de Llamada a Método, ver Descripción del caso de uso: Configurar Expresión de Llamada a Método (**Anexo 1**). 3.3.- Si es una expresión de Sentencia Cíclica, ver Descripción del caso de uso: Configurar Expresión de Sentencia Cíclica (**Anexo 1**). 3.4.- Si es una expresión de Sentencia Condicional, ver Descripción del caso de uso: Configurar Expresión de Sentencia Condicional (**Anexo 1**). 3.5.- Si es una expresión de Sentencia de Retorno, ver Descripción del caso de uso: Configurar Expresión de Sentencia de Retorno (**Anexo 1**). 3.6.- Si es una expresión de Sentencia de

	Transacción, ver Descripción del caso de uso: Configurar Expresión de Sentencia de Transacción (**Anexo 1**). 3.7.- Si es una expresión de Sentencia de Base de Datos, ver Descripción del caso de uso: Configurar Expresión de Sentencia de Base de Datos (**Anexo 1**).
Sección 1: Adicionar Expresión.	
Acción del Actor	Respuesta del Sistema
3.8.- El desarrollador da clic derecho sobre el cuerpo de la sentencia.	3.9.- La herramienta muestra las opciones de expresiones que tiene para la sentencia donde está marcando.
4.- Selecciona la opción expresión.	4.1.- Muestra la expresión en el lugar indicado por el desarrollador.
4.2.- El desarrollador selecciona el botón Adicionar(A)	4.3.- Se crea una nueva expresión.
Sección 2: Modificar Expresión.	
Acción del Actor	Respuesta del Sistema
4.2.- El desarrollador selecciona la expresión a modificar.	4.3.- La herramienta muestra las mismas opciones que para una nueva expresión.
4.4.- El desarrollador selecciona la nueva expresión o el valor nuevo para la que ya esté en el bloque de sentencias.	4.5.- La herramienta muestra los cambios antes de aplicar.
4.6.- El desarrollador presiona en el botón Aceptar.	4.7.- La herramienta aplica los cambios.
Sección 3: Eliminar Expresión.	
Acción del Actor	Respuesta del Sistema
4.8.- El desarrollador selecciona la expresión en el Editor del bloque de sentencias.	4.7.- La herramienta sombrea de un color distinto la expresión seleccionada.

4.7.- El desarrollador presiona el botón Eliminar (X).	4.8.- La herramienta borra la expresión seleccionada y mueve a su lugar, la siguiente en caso de que exista más de una expresión.
Poscondiciones:	Sección 1: Se adiciona una nueva expresión. Sección 2: Se modifica una expresión. Sección 3: Se elimina una expresión.
Prioridad:	Crítico.

Tabla 7. Descripción del caso de uso: Configurar Expresión.

Caso de uso:	Configurar Bloque de Sentencia.
Actores:	Desarrollador de software.
Propósito:	Configurar el bloque de sentencias de cualquiera de las sentencias que seleccione el actor.
Resumen:	El caso de uso inicia cuando el desarrollador levanta un Editor de Bloque de Sentencias, al seleccionar el botón correspondiente dentro de los métodos del conector o tipo.
Referencias:	RF-2.2(2.2.1, 2.2.1.1, 2.2.1.1.1, 2.2.1.1.2, 2.2.1.1.2.1, 2.2.1.1.2.2, 2.2.1.1.2.2.1, 2.2.1.1.2.2.2, 2.2.1.1.2.2.3, 2.2.1.1.2.4, 2.2.1.1.2.5, 2.2.1.1.2.5.1, 2.2.1.2, 2.2.1.2.1, 2.2.1.2.2, 2.2.1.2.2.1)
Precondiciones:	El desarrollador debe seleccionar el método para el que va a crear, modificar o eliminar el bloque de sentencias.
Flujo Normal de los Eventos	
Acción del Actor	Respuesta del Sistema
1.- El desarrollador presiona el botón que agrupa las sentencias (+).	2.- La herramienta muestra las sentencias que están creadas.
Sección 1: Adicionar Bloque de Sentencias.	
Acción del Actor	Respuesta del Sistema
3.- El desarrollador da presiona el botón	4.-Ver descripción del caso de uso:

Adicionar(A).	Configurar Sentencia Sección 1.
Sección 2: Insertar Bloque de Sentencias.	
Acción del Actor	Respuesta del Sistema
5.- El desarrollador presiona el botón Insertar (I).	6.- Ver descripción del caso de uso: Configurar Sentencia Sección 2.
Sección 3: Modificar Bloque de Sentencias.	
Acción del Actor	Respuesta del Sistema
7.- El desarrollador selecciona el bloque de sentencias a modificar.	8.- Ver descripción del caso de uso: Configurar Sentencia Sección 3.
Sección 4: Eliminar Bloque de Sentencias.	
Acción del Actor	Respuesta del Sistema
9.- El desarrollador selecciona el bloque de sentencias a eliminar.	10.- Ver descripción del caso de uso: Configurar Sentencia Sección 4.
Poscondiciones:	Sección 1: Se adiciona un bloque de sentencias. Sección 2: Se inserta un bloque de sentencias. Sección 3: Se modifica un bloque de sentencias. Sección 4: Se elimina un bloque de sentencias.

Tabla 8. Descripción del caso de uso: Configurar Bloque de Sentencia.

Demás descripciones de casos de uso del sistema, **ver Anexo 1**.

3.4 Conclusiones

En este capítulo se presentaron los casos de uso más importantes del sistema, sus descripciones y actores del sistema (para nuestra solución es sólo uno, el desarrollador de software). Con estos elementos es que se modela cómo debe ser la interacción con la herramienta, su comportamiento y resultados. Mediante la descripción de los casos de uso del sistema se pueden ir demostrando las condiciones y cualidades que tendrá la herramienta propuesta.

CAPÍTULO 4: ANÁLISIS Y DISEÑO DE LA HERRAMIENTA

4.1 Introducción

En el siguiente capítulo se tratarán de comprender perfectamente los requisitos de la herramienta mediante la modelación de las clases de análisis y se precisará, a través de la modelación de las clases del diseño, cómo se implementará la solución. Se presentarán además, diagramas de interacción de los casos de uso críticos del sistema para una mejor comprensión.

4.2 Análisis

4.2.1 Modelo de clases del análisis

4.2.1.1 Diagramas de clases del análisis

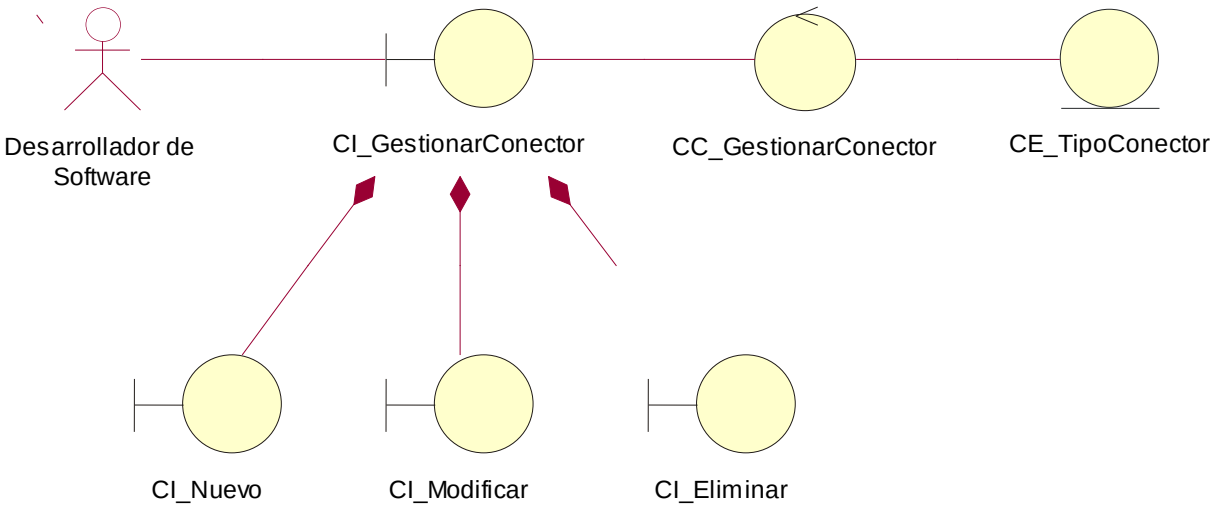

Figura 4.1 Diagrama de clases del análisis para el caso de uso Gestionar Conector.

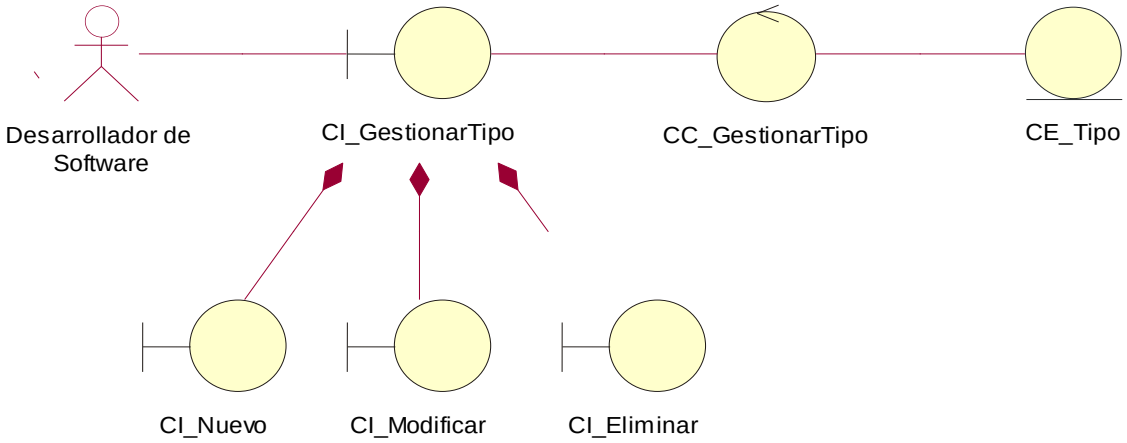

Figura 4.2 Diagrama de clases del análisis para el caso de uso Gestionar Tipo.

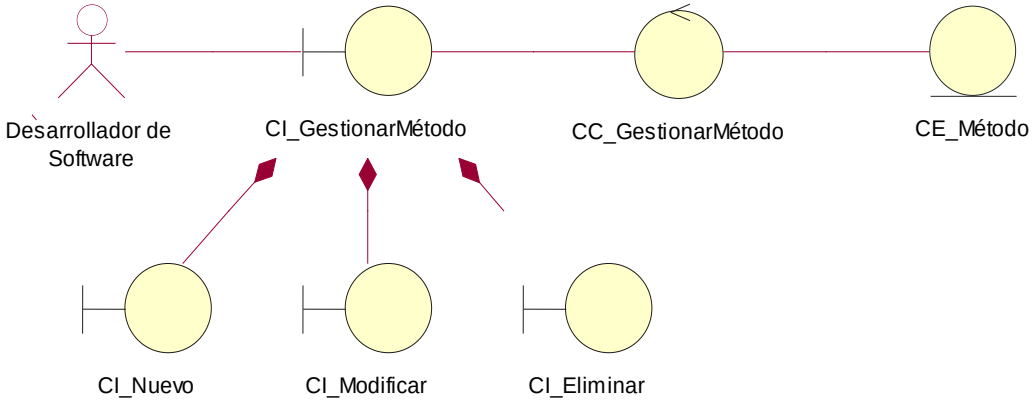

Figura 4.3 Diagrama de clases del análisis para el caso de uso Gestionar Método.

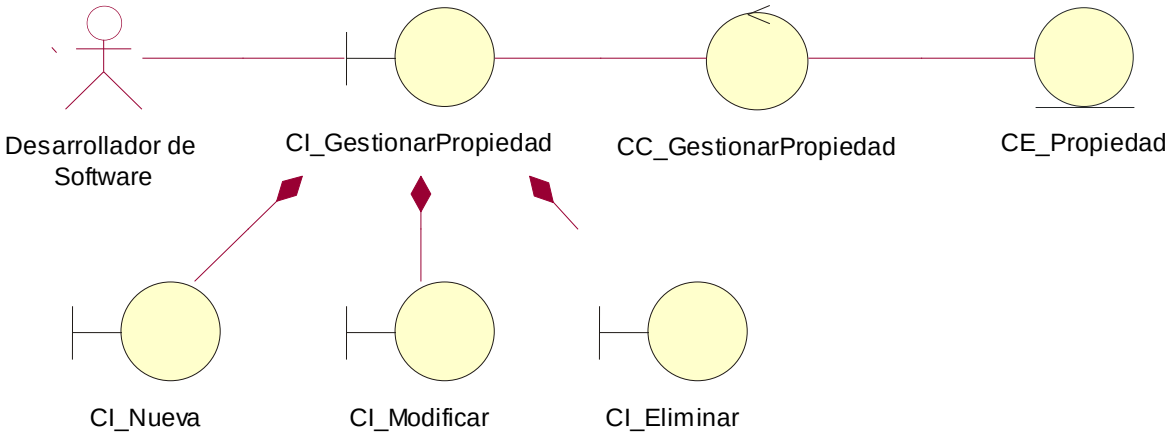

Figura 4.4 Diagrama de clases del análisis para el caso de uso Gestionar Propiedad.

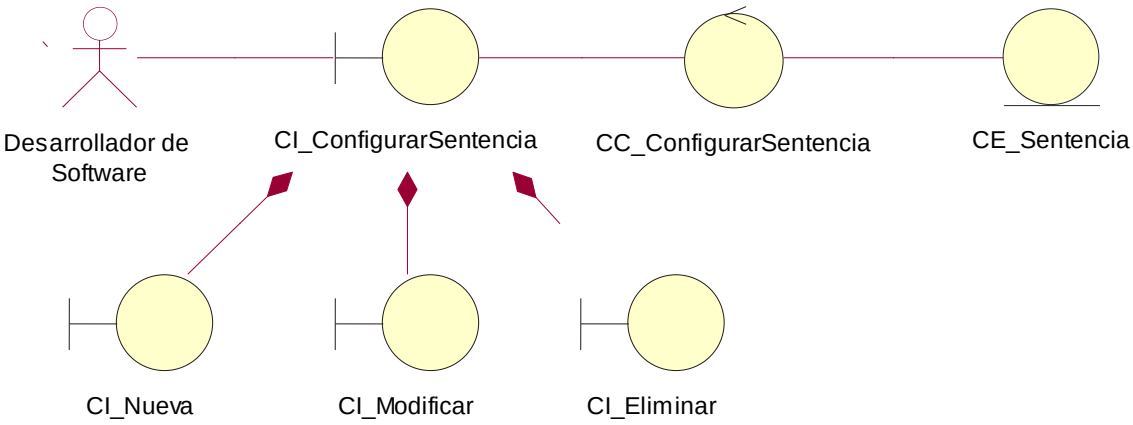

Figura 4.5 Diagrama de clases del análisis para el caso de uso Configurar Sentencia.

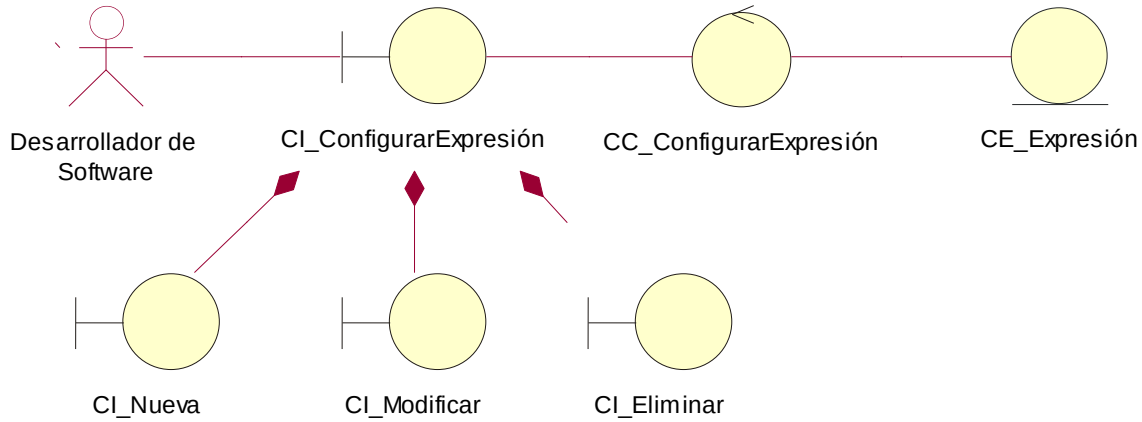

Figura 4.6 Diagrama de clases del análisis para el caso de uso Configurar Expresión.

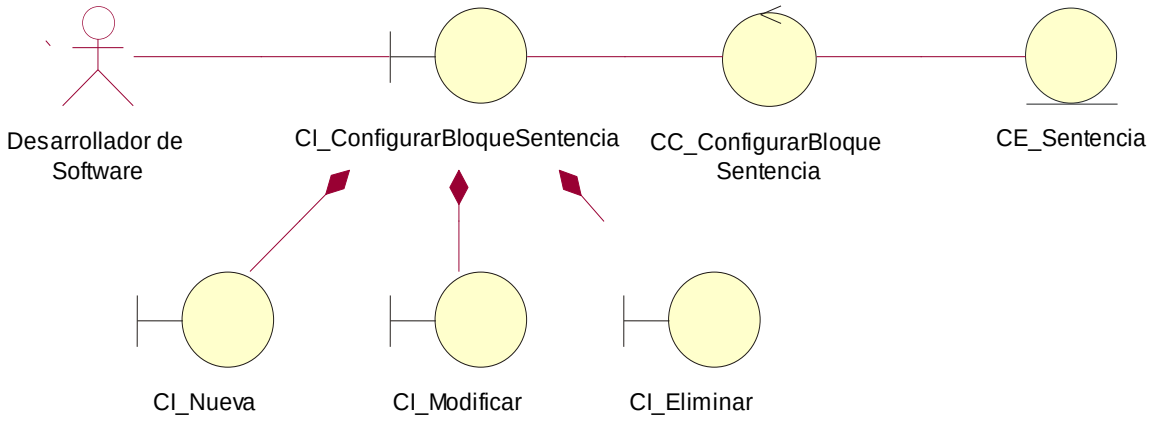

Figura 4.7 Diagrama de clases del análisis para el caso de uso Configurar Bloque de Sentencia.

4.3 Diseño

4.3.1 Descripción de la arquitectura de la aplicación

Para la interacción con el usuario la aplicación está diseñada mediante la implementación del patrón de arquitectura MVC, (Model View Controler o Modelo Vista Controlador). Este

patrón permite un desacople entre las capas de la aplicación, marcando la separación entre la vista o interfaz de usuario, el modelo de diseño y otras funcionalidades como por ejemplo la gestión de datos.

Para la implementación de la solución propuesta se utilizó un DOM (Document Object Model o Modelo de Objetos del Documento) que funciona como base de todas las operaciones que se manejen en la herramienta. Todo en XCAD es convertido al DOM para su procesamiento de manera transparente al usuario.

4.3.2 Definición del DOM Lenguaje

El DOM, es la interfaz que permite acceder y manipular, mediante la programación, los contenidos de un entorno. Proporciona una representación estructurada, orientada a objetos, de los elementos individuales y el contenido de un objeto, con métodos que recuperen y fijen sus propiedades. Proporciona además, métodos para agregar y eliminar los objetos, permitiendo crear contenido dinámico.

Este importante paquete (***Lenguaje***), es la base de XCAD. Lo que un desarrollador modele a nivel de interfaz es transformado al DOM y sus funcionalidades permiten que la aplicación haga invisibles al usuario complejos procesos.

4.3.3 Modelo de clases del diseño

4.3.3.1 Diagramas de clases del diseño

Figura 4.8 Diagrama de clases del diseño para el caso de uso Gestionar Conector.

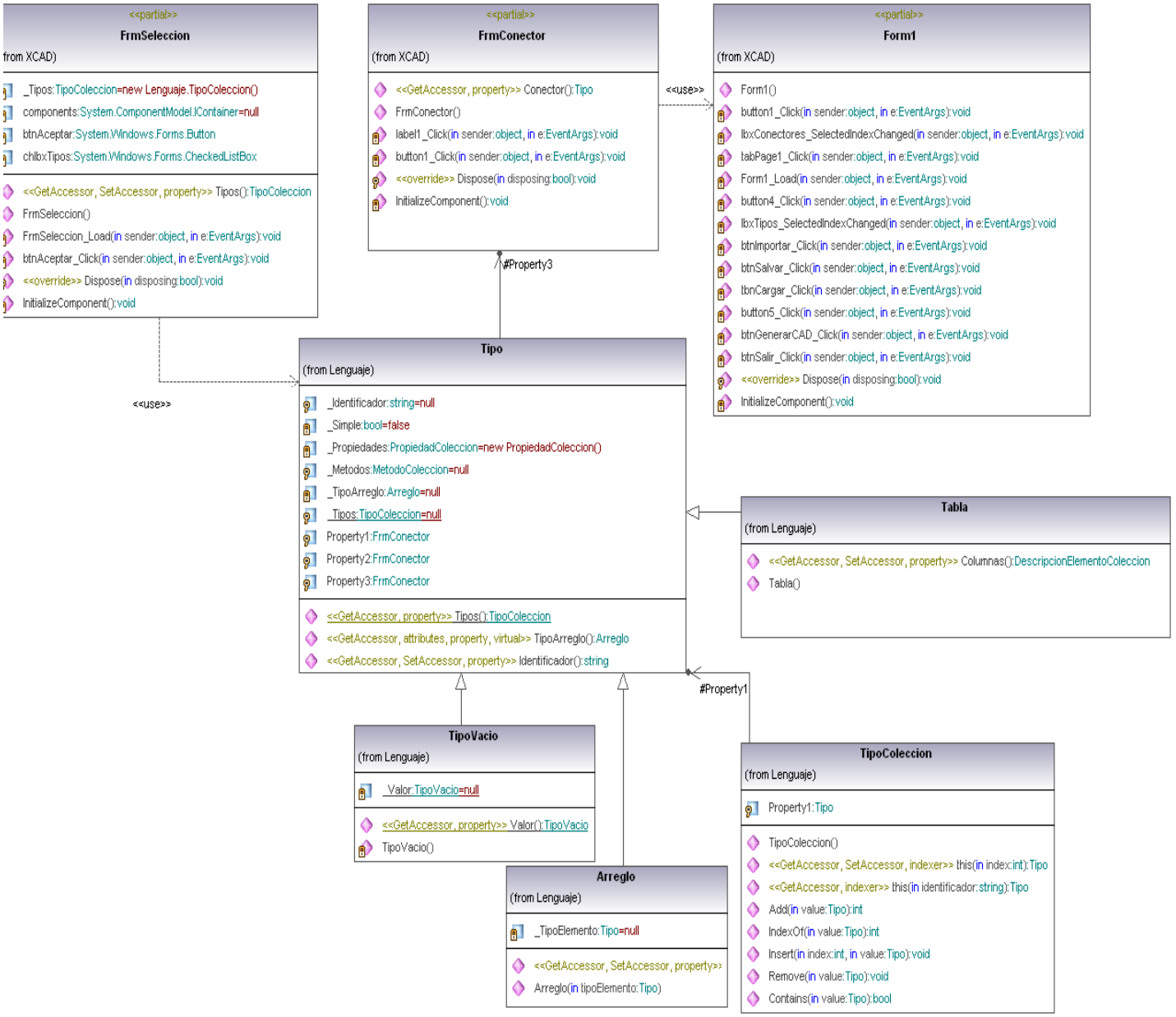

Figura 4.9 Diagrama de clases del diseño para el caso de uso Gestionar Tipo.

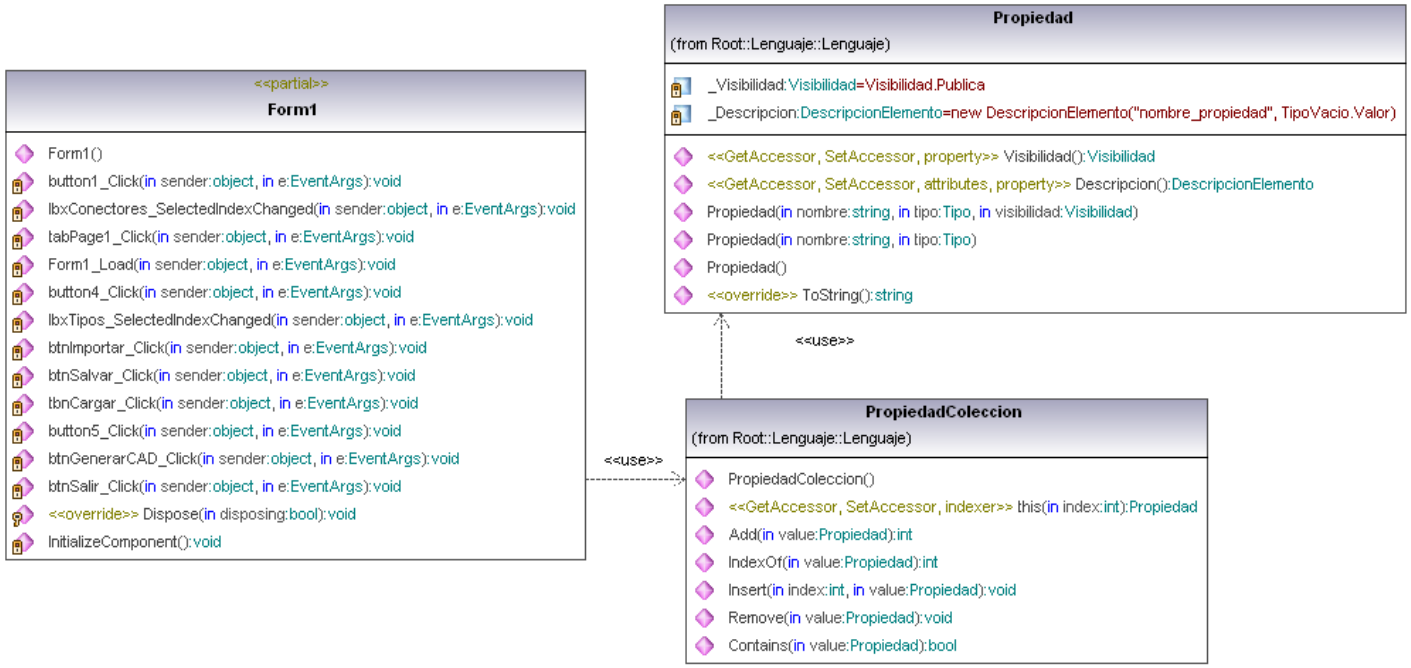

Figura 4.10 Diagrama de clases del diseño para el caso de uso Gestionar Propiedad.

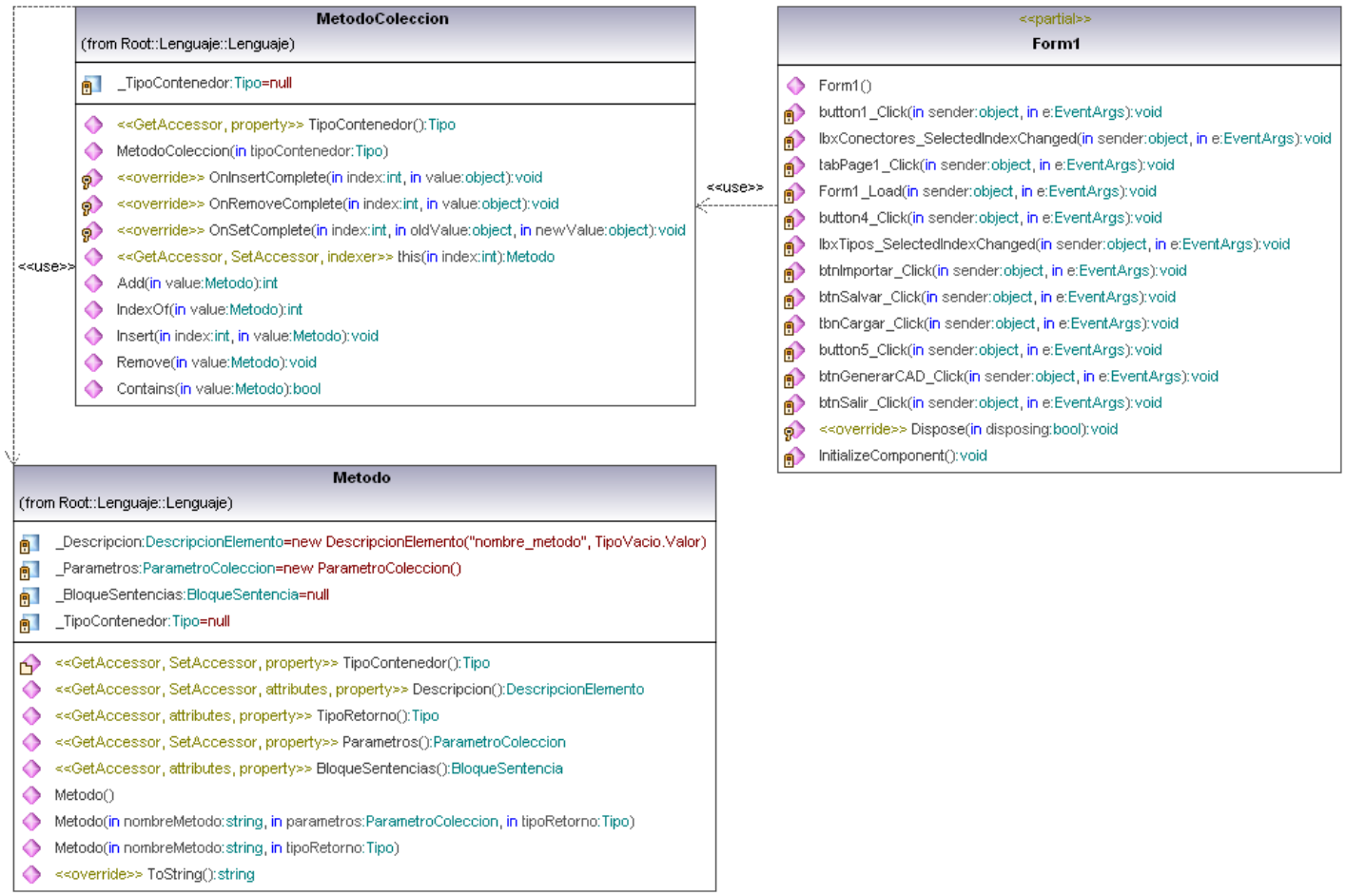

Figura 4.11 Diagrama de clases del diseño para el caso de uso Gestionar Método.

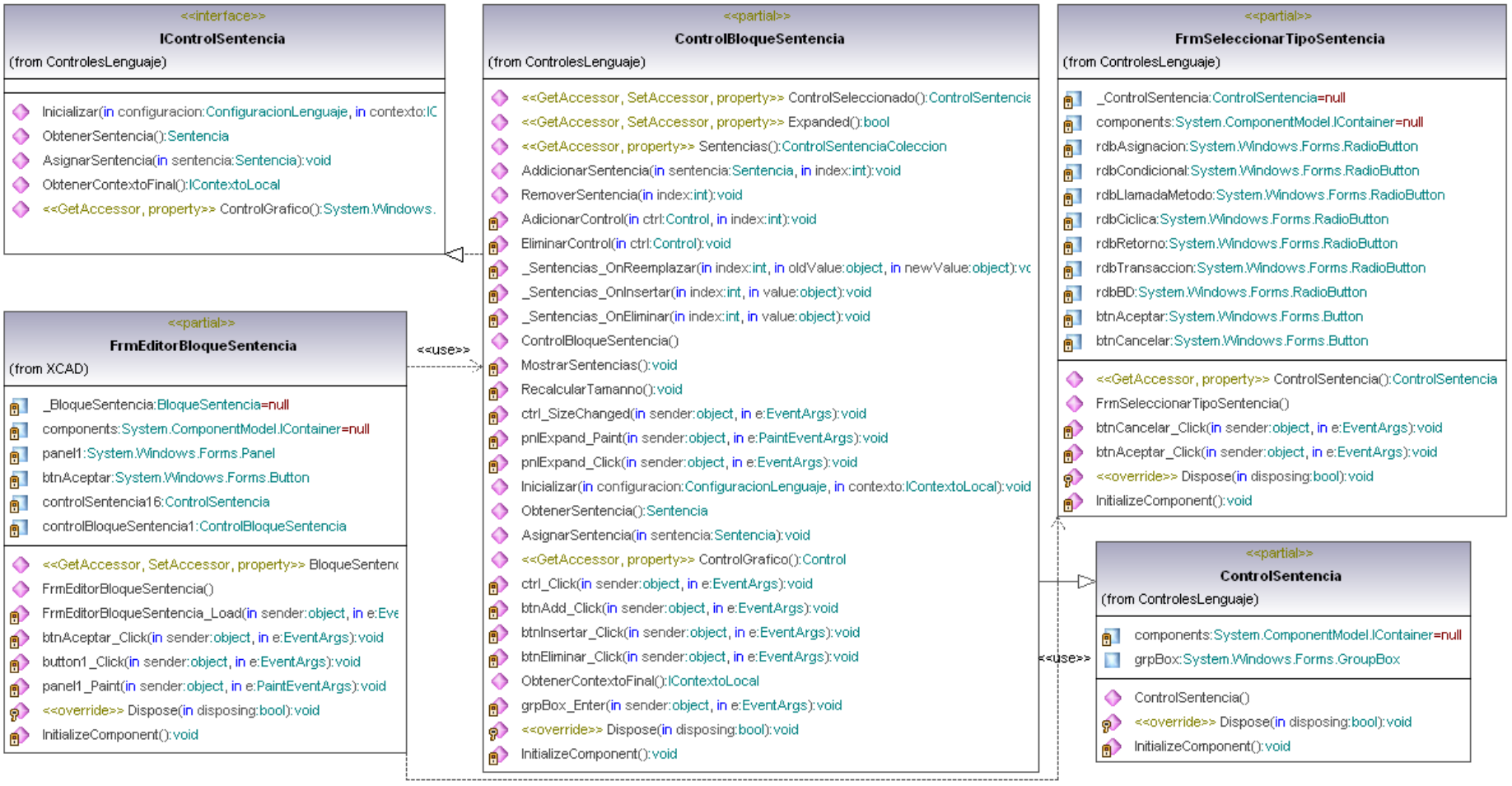

Figura 4.12 Diagrama de clases del diseño para el caso de uso Configurar Sentencia.

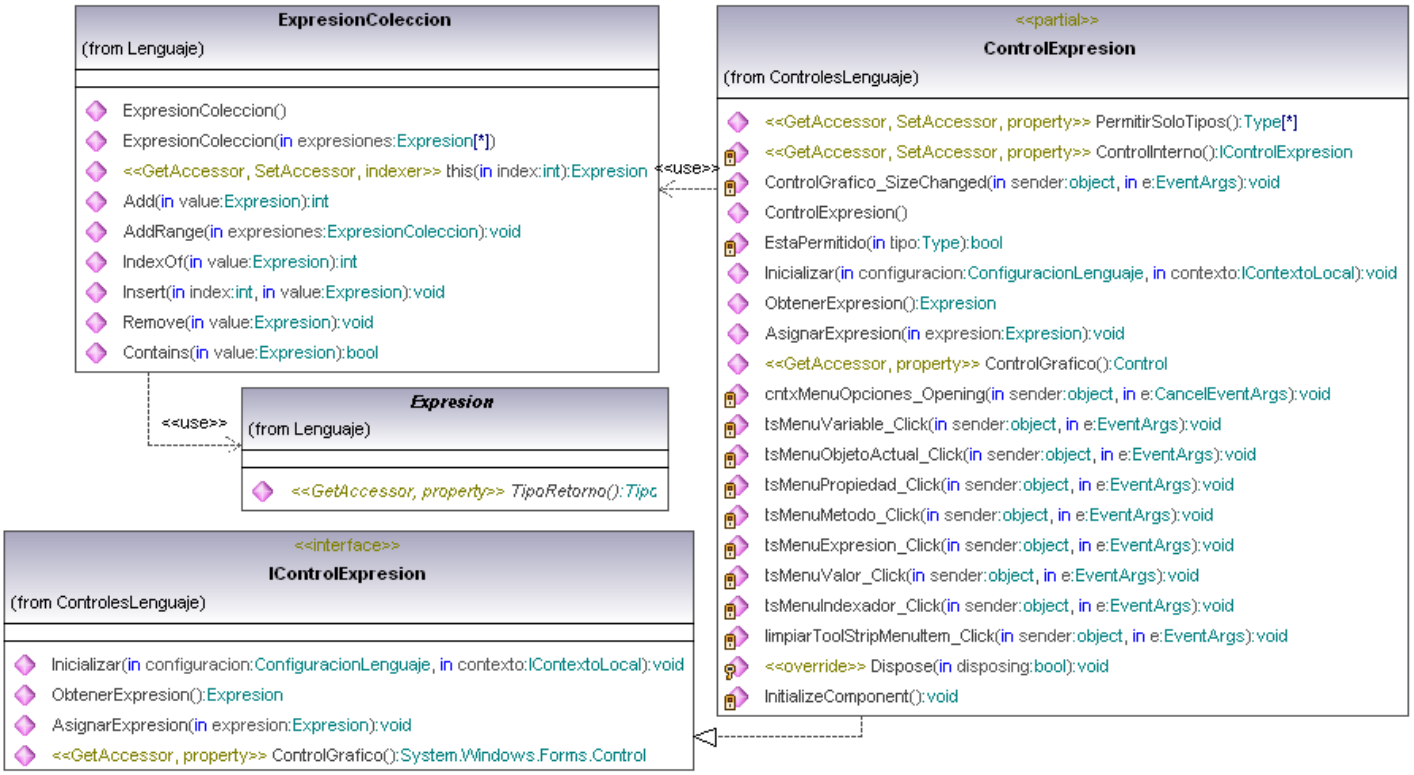

Figura 4.13 Diagrama de clases del diseño para el caso de uso Configurar Expresión.

Figura 4.14 Diagrama de clases del diseño para el caso de uso Configurar Bloque de Sentencia.

4.3.4 Diagramas de Interacción (Secuencia)

Por cada escenario de caso de uso se diseñó un diagrama de interacción (específicamente un diagrama de secuencia) donde se modelen los aspectos dinámicos del sistema y se destaque un conjunto de objetos y sus relaciones que deben incluir el ordenamiento temporal de los mensajes que se puedan generar entre ellos.

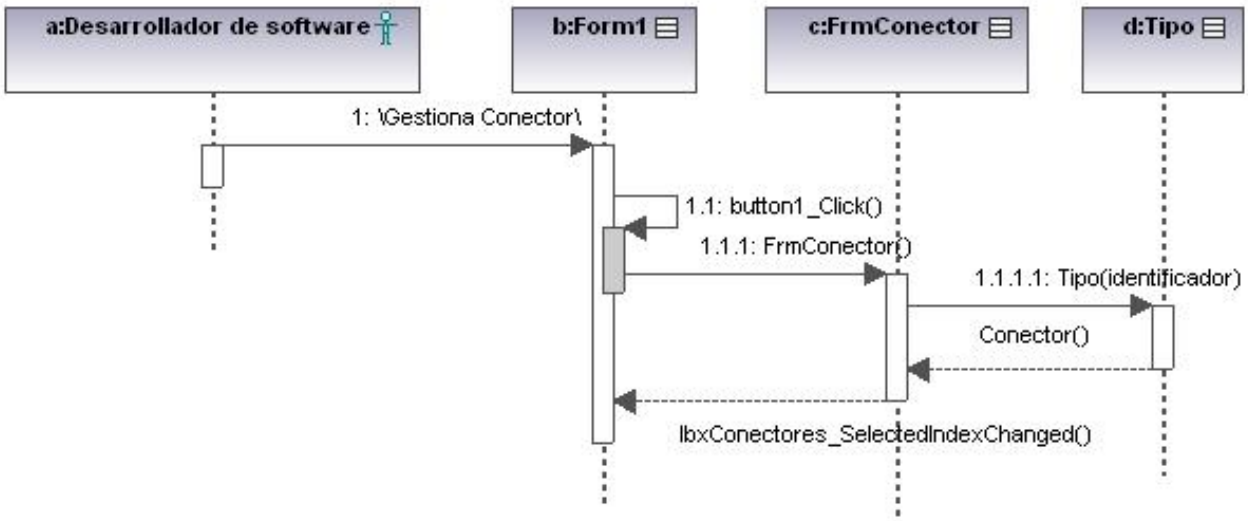

Figura 4.15 Diagrama de secuencia para el caso de uso Gestionar Conector.

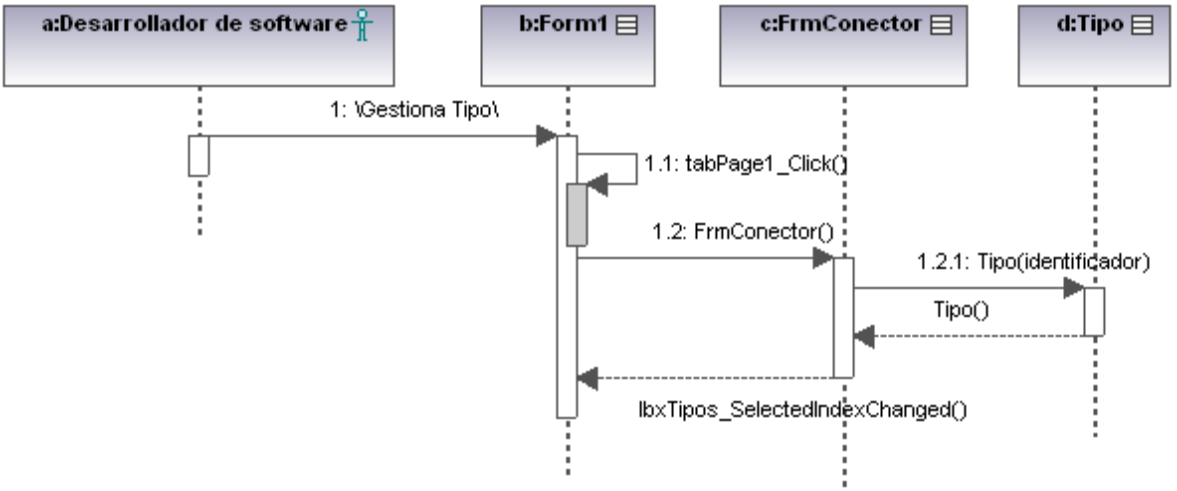

Figura 4.16 Diagrama de secuencia para el caso de uso Gestionar Tipo.

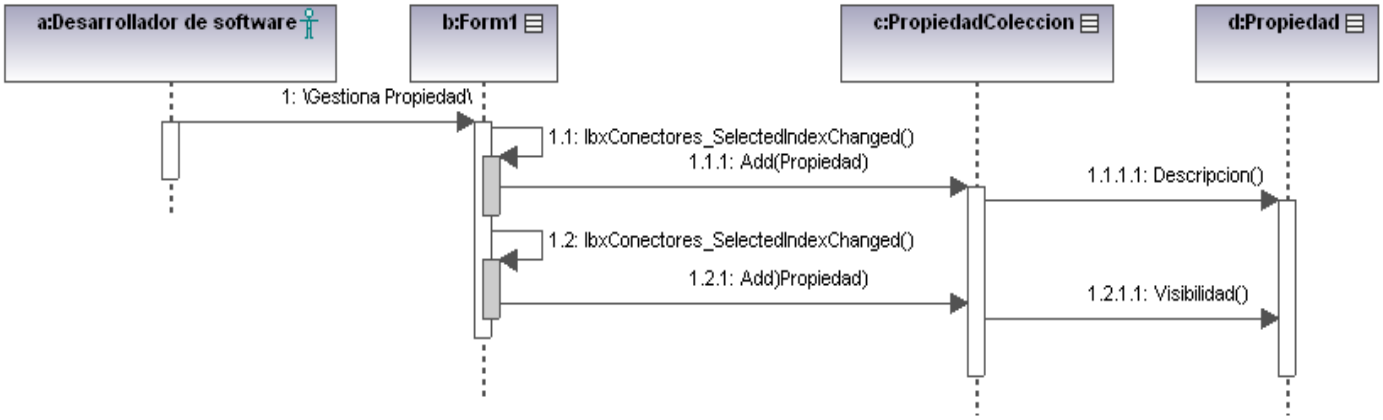

Figura 4.17 Diagrama de secuencia para el caso de uso Gestionar Propiedad.

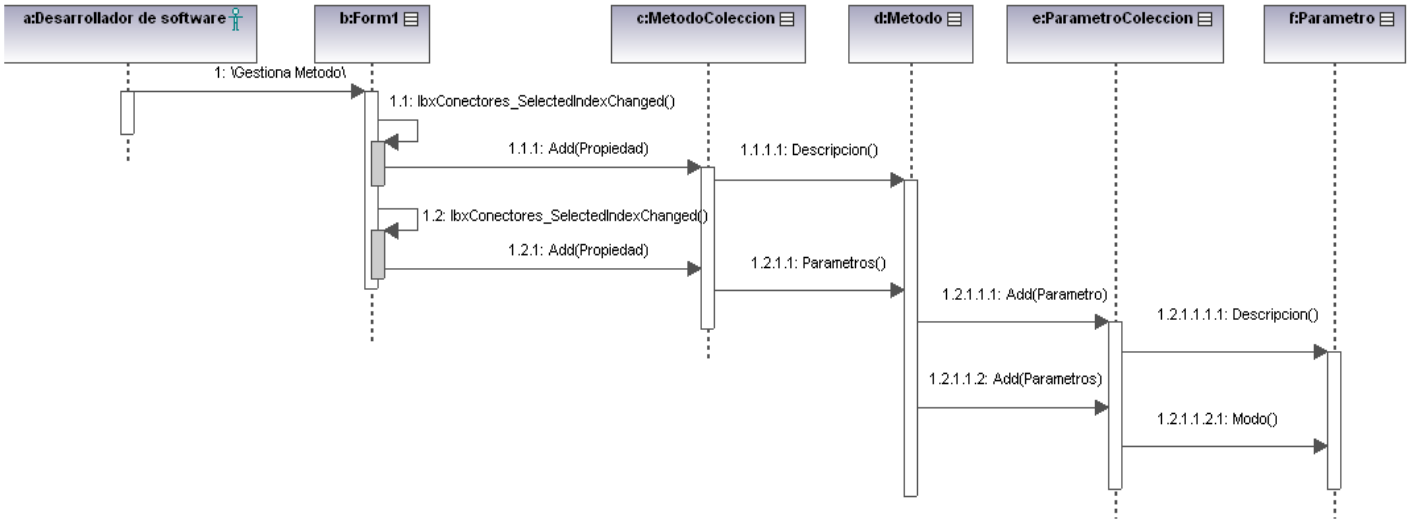

Figura 4.18 Diagrama de secuencia para el caso de uso Gestionar Método.

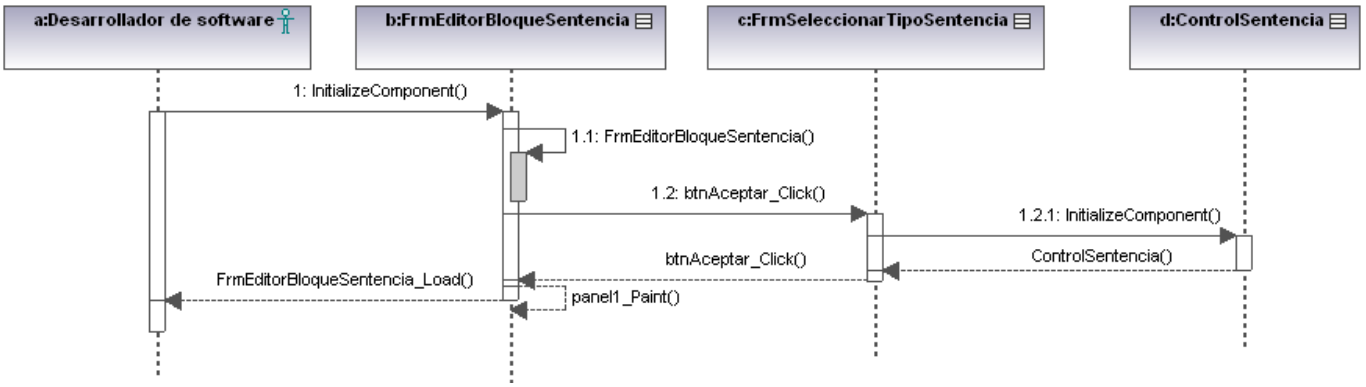

Figura 4.19 Diagrama de secuencia para el caso de uso Configurar Sentencia.

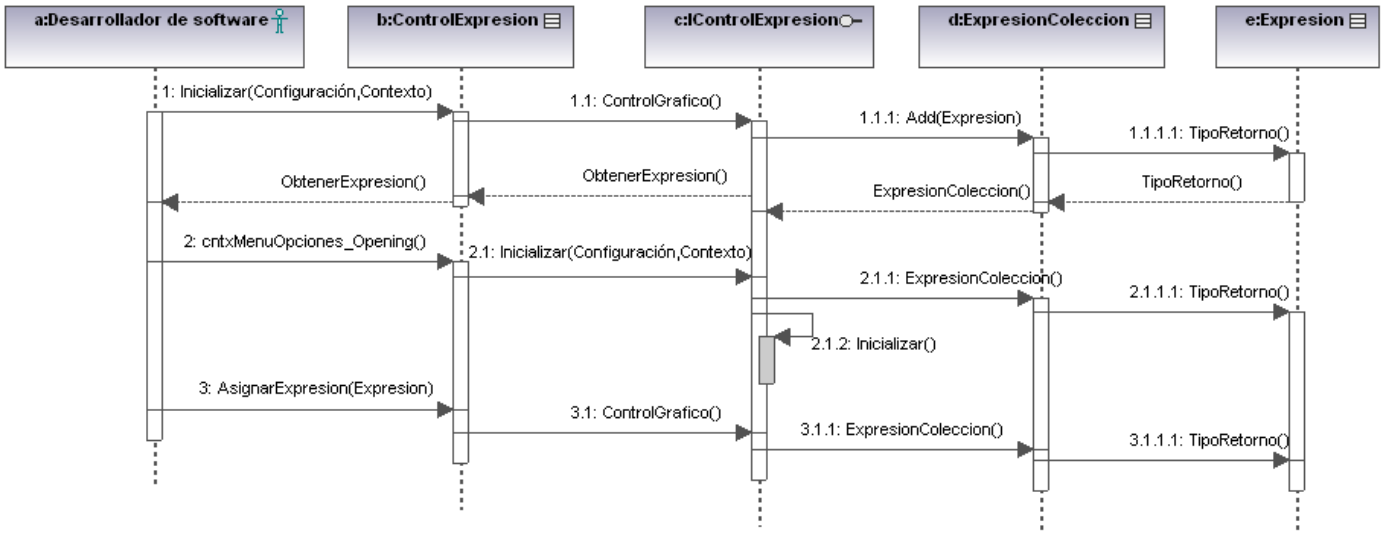

Figura 4.20 Diagrama de secuencia para el caso de uso Configurar Expresión.

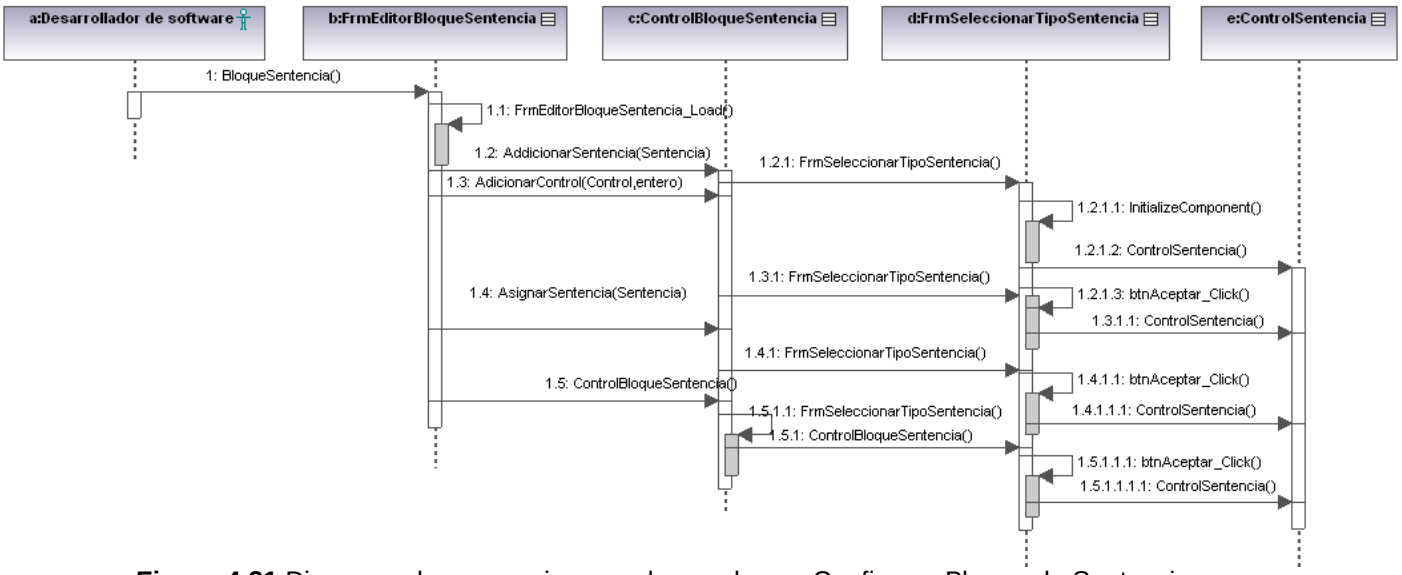

Figura 4.21 Diagrama de secuencia para el caso de uso Configurar Bloque de Sentencia.

4.4 Diseño a aplicar

La interfaz gráfica del usuario constituye el medio de interacción con el sistema. Por tales razones debe ser lo más agradable y clara posible, no sólo para lograr uniformidad entre sus componentes, también para que el cliente se sienta cómodo y logre adaptarse a su ambiente de trabajo.

4.4.1 Aspectos definidos para las interfaces de usuario

- Todos los botones deben tener imágenes sugerentes de manera que el usuario memorice la imagen y no necesite buscar mucho mientras se adapte al entorno.
- El área de trabajo debe estar dividida en 2 partes, conectores y tipos, pero necesariamente para procesarlos el usuario debe ubicarse en el entorno correspondiente.

- Los botones de generación de código no deben estar muy separados ni en contextos diferentes.
- Las colecciones de cada elemento deben ser mostradas en listbox con opciones de eliminar y adicionar en la esquina inferior derecha.
- La herramienta debe contar con varias áreas de trabajo.

La interfaz de usuario de la herramienta propuesta, está compuesta por formularios Windows y la pantalla principal se divide en 3 partes:

1. Menú de conectores y botones de las operaciones principales (véase **figura 4.4.1**).
2. Menú de tipos (véase **figura 4.4.2**).
3. Áreas de trabajo dentro de cada menú (véase **figura 4.4.3**).

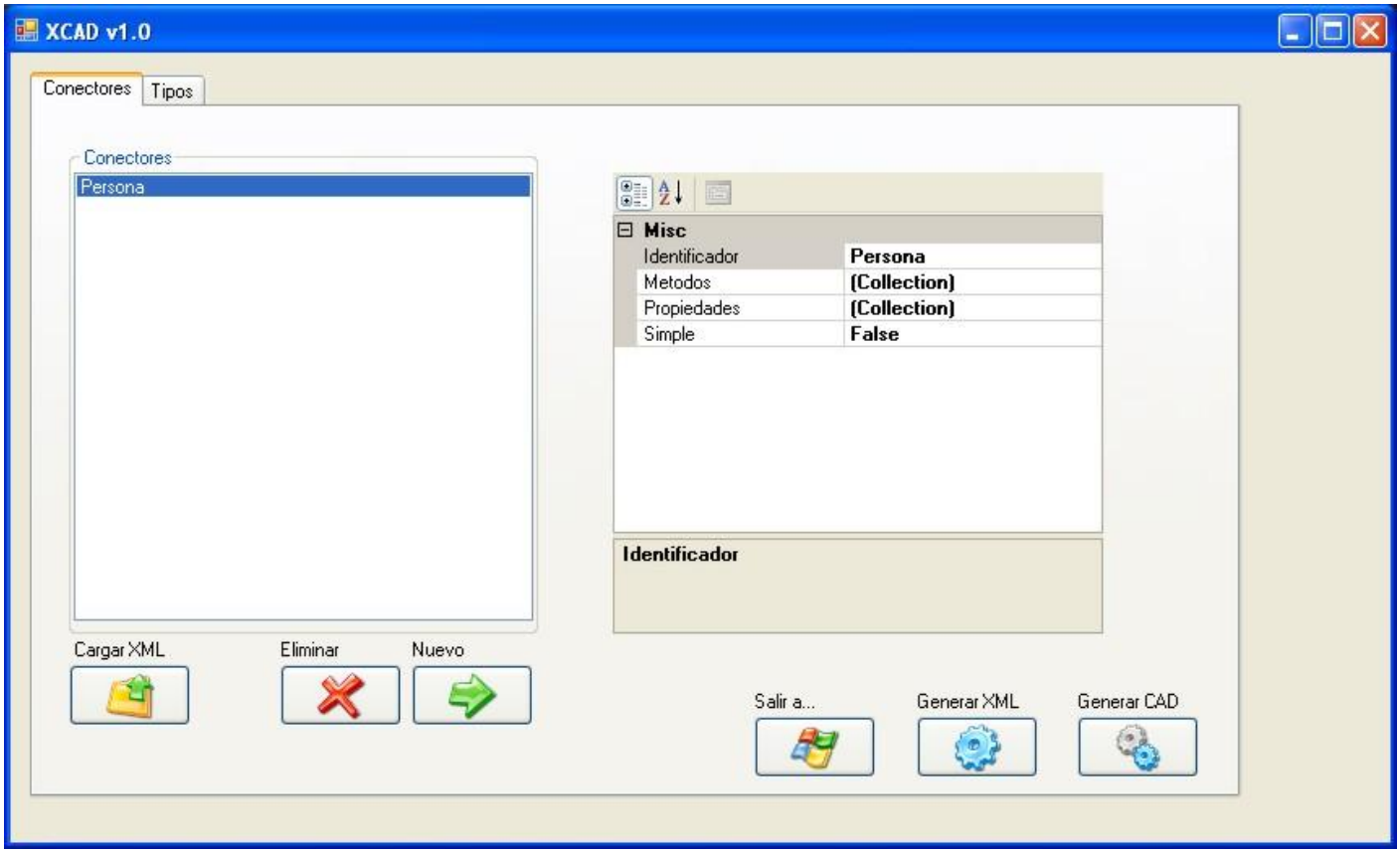

Figura 4.22 Entorno de *Conector* (Persona en este caso), interfaz principal de XCAD.

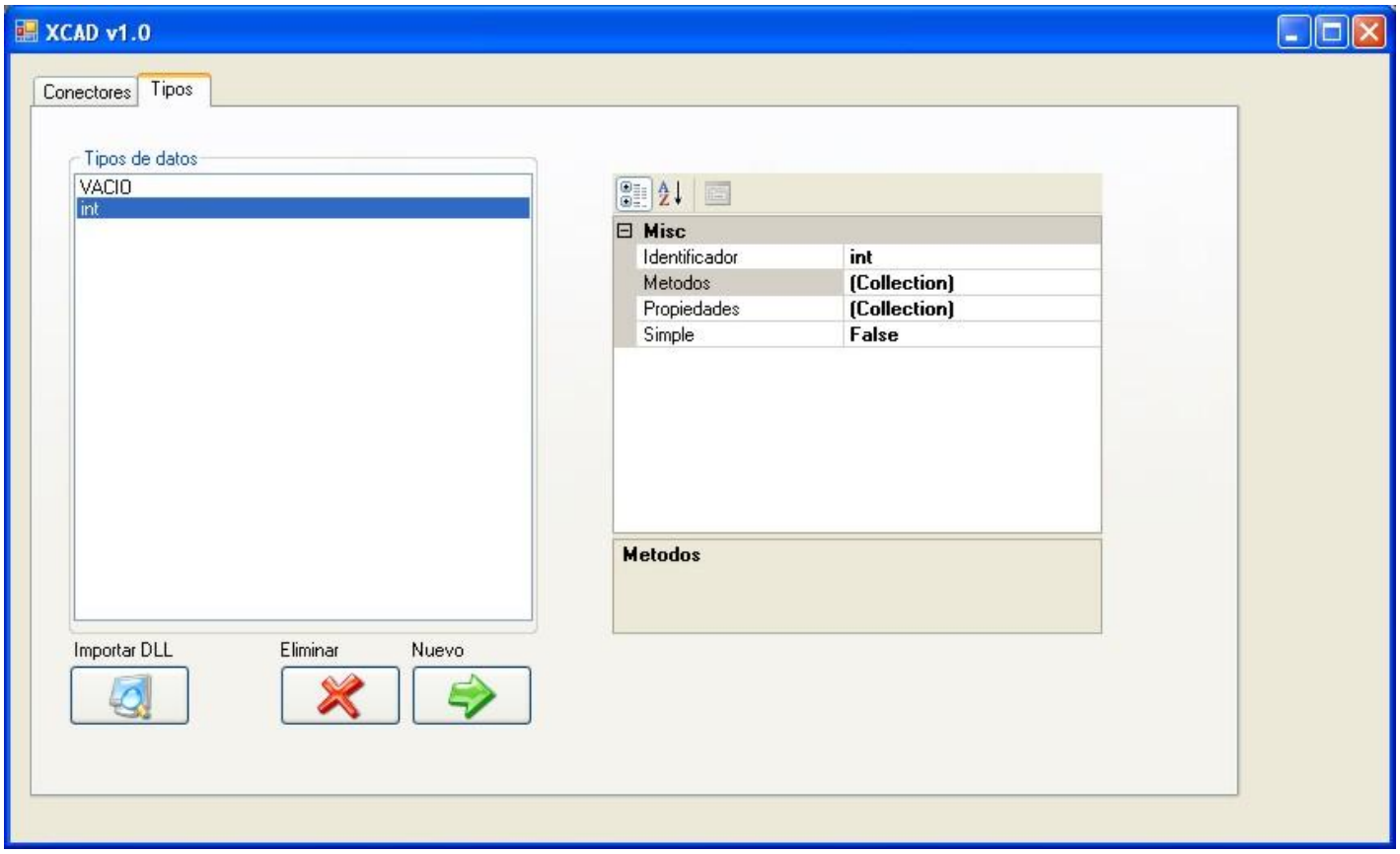

Figura 4.23 Entorno de *Tipos* (se aprecia el Tipo “int” creado).

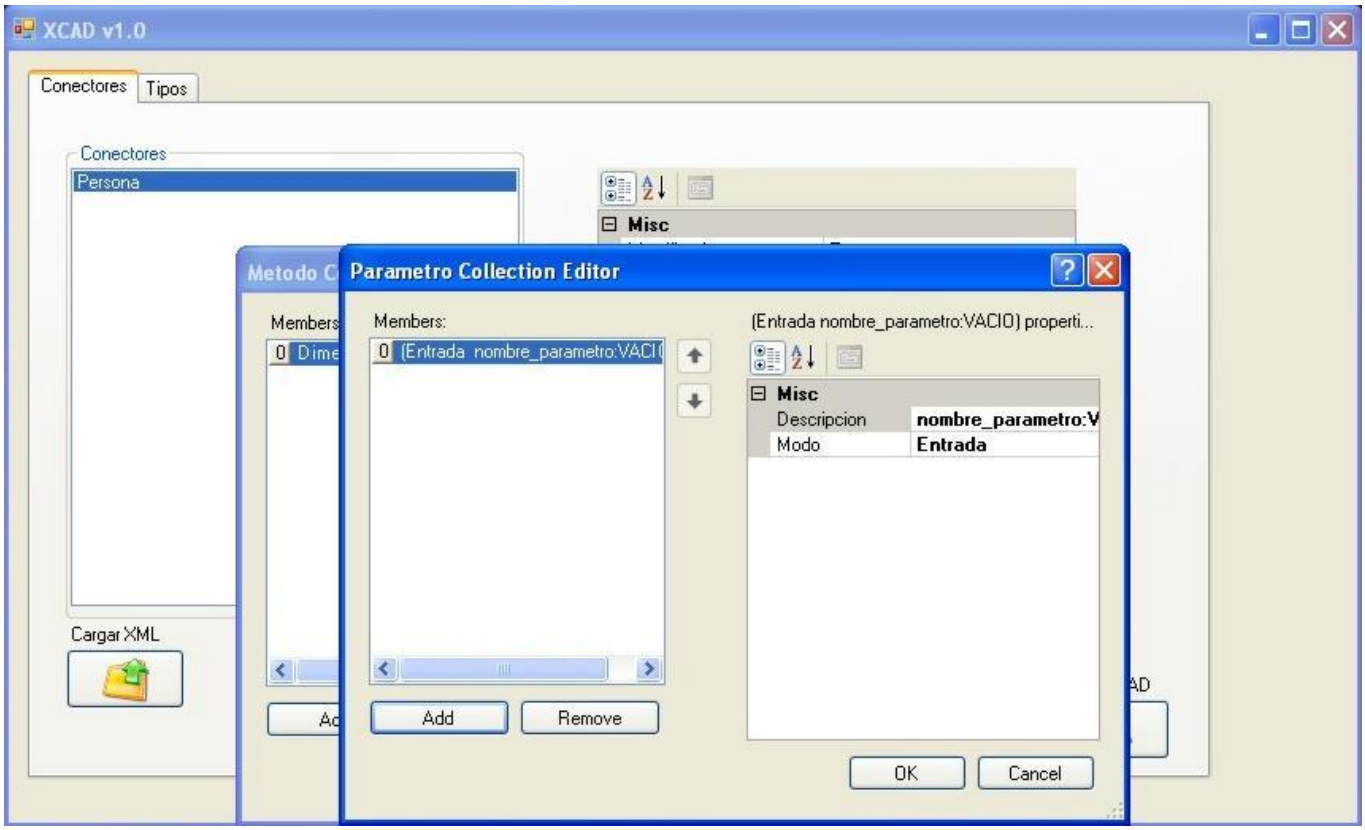

Figura 4.24 Se aprecia que la edición de parámetros, a un método del conector *Persona* (véase, Anexo 4).

4.4.2 Tratamiento de errores

Para que la herramienta tenga buenos resultados en su funcionamiento, es preciso informarle al usuario cuando es incorrecta la operación que realice. La herramienta necesita una buena validación en sus controles a nivel de interfaz y tratamiento de excepciones a nivel del código.

Siempre se debe:

- Detectar las excepciones.
- Mostrar la información sobre la excepción detectada.

Un sistema para el tratamiento de excepciones debe estar bien encapsulado. Con el Framework 2.0 de .NET se realizan las operaciones de detección, encapsulamiento y

propagación de excepciones de manera simple. Con el buen uso de estas funcionalidades se pueden generar mensajes uniformes de manera que el usuario comprenda que hizo mal y lo solucione. Como indica la siguiente figura:

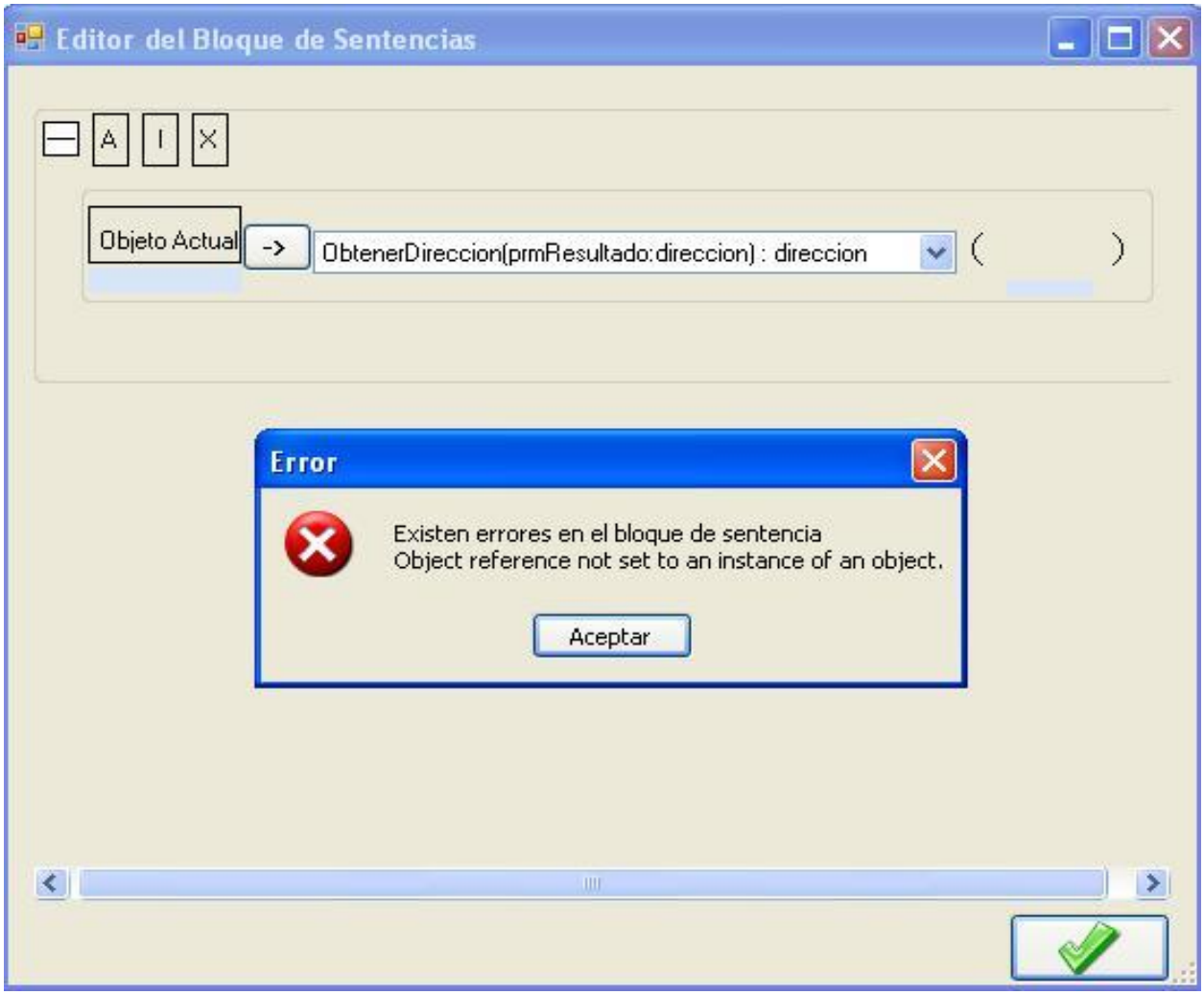

Figura 4.25 Mensaje de error personalizado.

4.5 Conclusiones

En este capítulo se realizó un minucioso razonamiento sobre los principales detalles a contemplar en los diagramas del análisis, del diseño y los de interacción (específicamente los de secuencia). Además se describieron los principios de diseño fundamentales en que se basa la herramienta propuesta para ir definiendo paulatinamente sus características físicas y técnicas.

CAPÍTULO 5: IMPLEMENTACIÓN

5.1 Introducción

En el siguiente capítulo se presenta cómo está diseñada y estructurada la herramienta XCAD y su comportamiento después de su instalación. Además se describe cómo interactúa con sus componentes internos y las ventajas que proporciona la integración con algunos componentes externos.

5.2 Diagrama de Despliegue

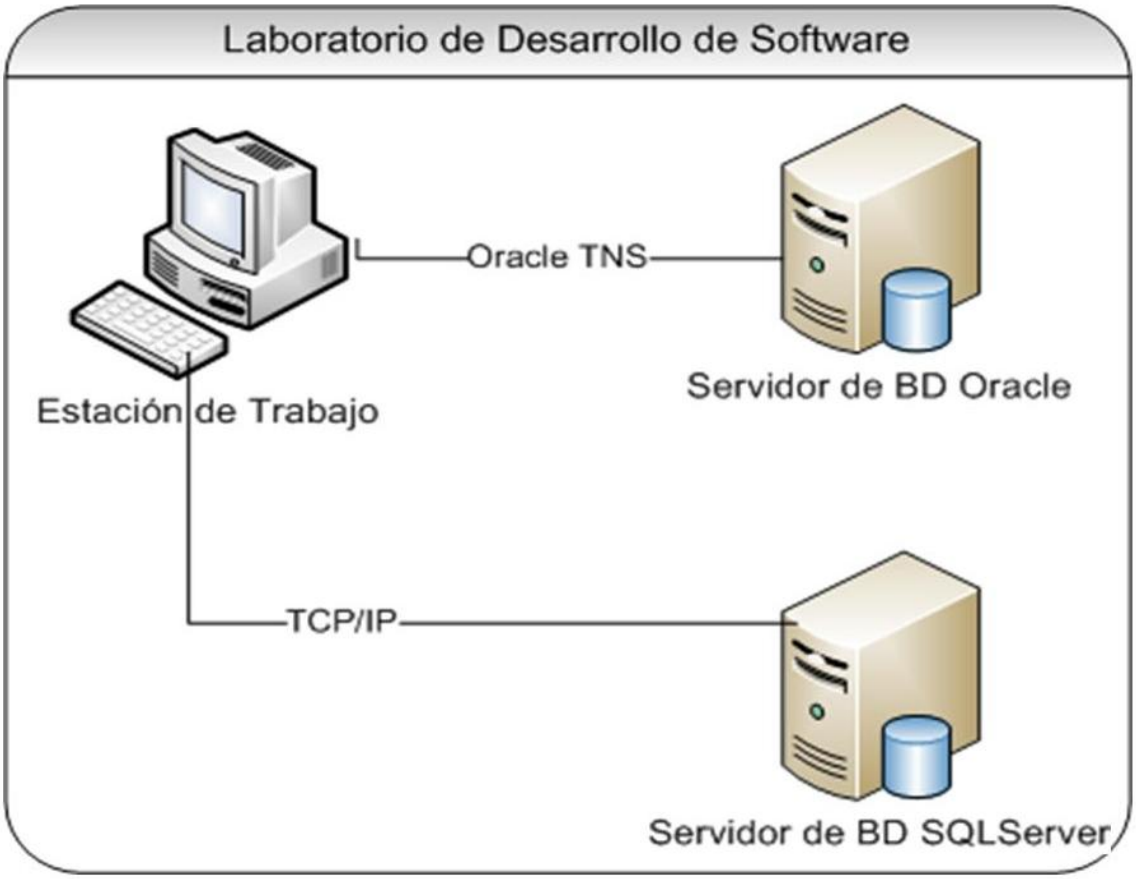

Figura 5.1 Diagrama de despliegue.

5.2.1 Descripción del Diagrama de Despliegue

El modelo de despliegue es un modelo de objetos que describe la distribución física de la herramienta en las estaciones de trabajo. A pesar de sólo modelarse dos entornos (base de datos de Oracle y base de datos en SQLServer) es válido señalar que la herramienta por sus cualidades genéricas es fácilmente adaptable para otros proveedores.

5.3 Diagrama de Componentes

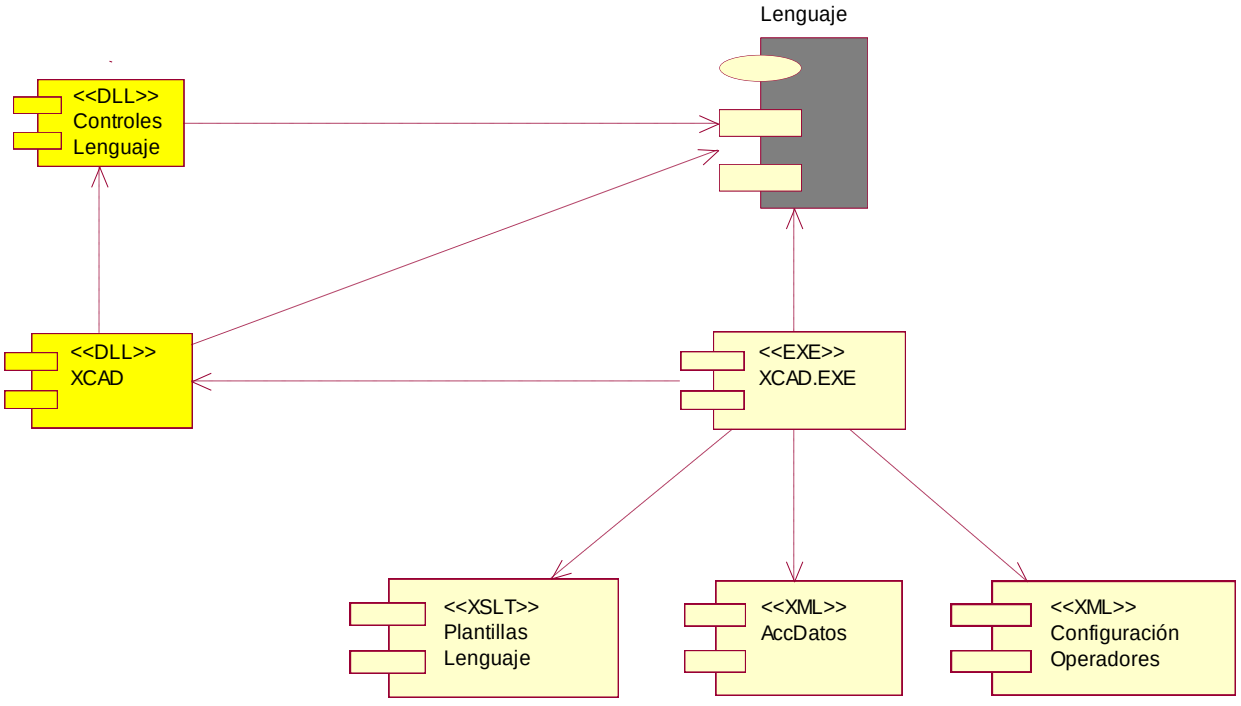

Figura 5.2 Diagrama de componentes.

5.3.1 Descripción del Diagrama de Componentes

El anterior diagrama muestra los principales componentes para la arquitectura y las referencias que hacen a otros. A pesar de ser complicado de apreciar, hay una influencia del patrón de arquitectura Modelo Vista Controlador entre las capas de interfaz de la herramienta

pues los componentes gráficos, los formularios y el negocio como tal, se encuentran en niveles distintos. En síntesis, en el nivel superior de la herramienta, se encuentran ubicados los controles visuales en formularios y en otro nivel se encuentran los controles del lenguaje, que son componentes gráficos que interactúan con el DOM (Document Object Model o Modelo de Objetos de Documento). Además existen otros componentes externos que se manejan desde las interfaces sin estar precisamente al nivel de estas. Se hace referencia a información externa configurable por el usuario y almacenada en XML.

La configuración de operadores almacenada en XML, le permite al usuario tener una herramienta más configurable, pues con un XML donde se maneje la información de los operadores es posible configurar el comportamiento de estos en el lenguaje para el que se van a utilizar. Algo similar ocurre con las plantillas que tienen la información del lenguaje, deben estar incluidas solo en el momento de generar en la aplicación

5.3.2 Descripción y uso de las plantillas del lenguaje

Las plantillas del lenguaje se crean para proporcionarle a la herramienta la independencia del negocio en que se esté desarrollando. Basta hacer un cambio de plantillas para obtener lo que se modele en otro lenguaje. Estas plantillas son XSLT para aplicar sobre el XML, en el momento de generar la CAD, se mostrará una interfaz para ensamblar (**Véase la figura 5.3**) lo que se debe generar: un XML con la información creada, la plantilla que se seleccione proporciona el lenguaje y la ubicación del fichero a generar.

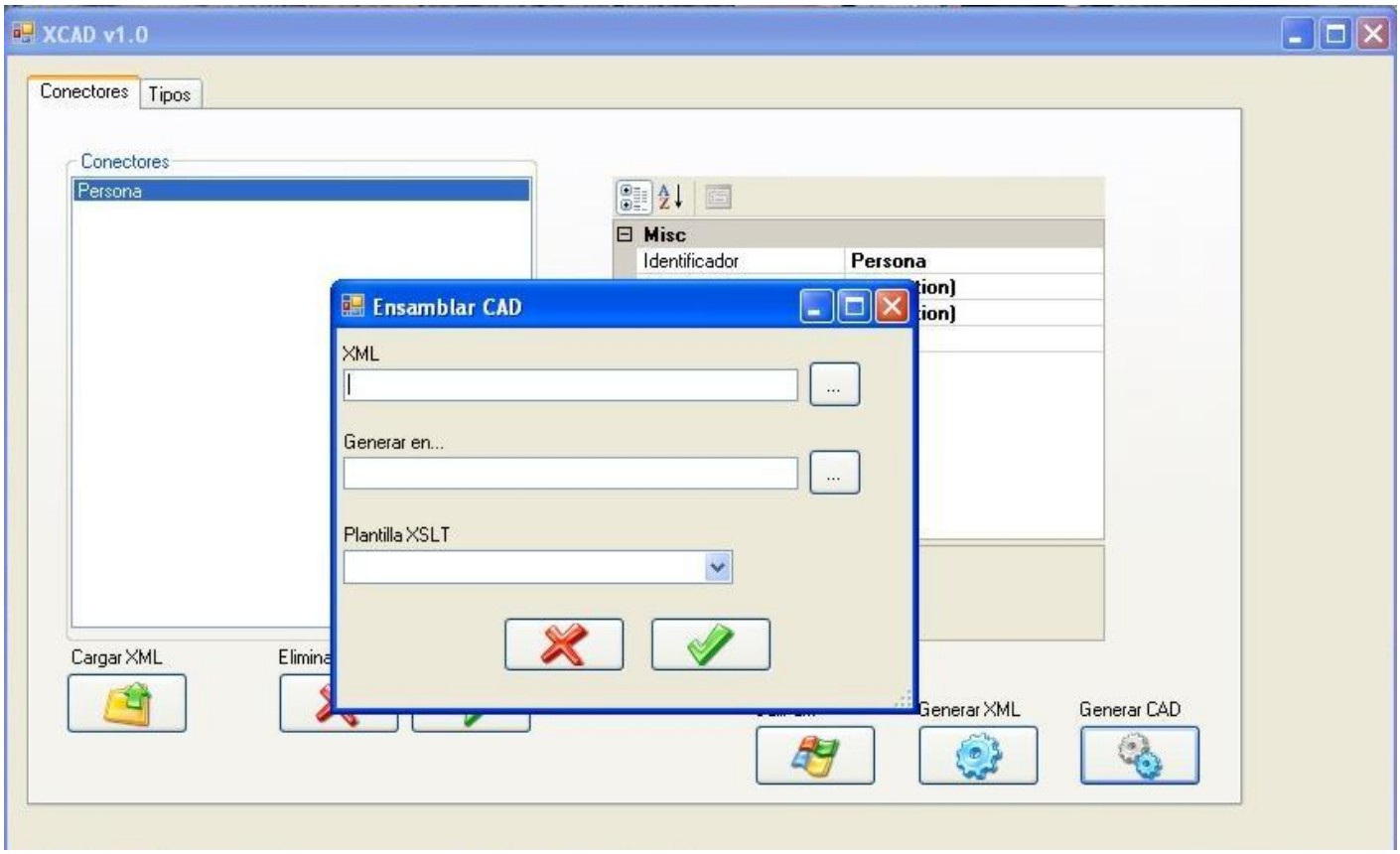

Figura 5.3 Pantalla para ensamblar CAD (sin completar, véase **Anexo 4 figura 5.4**)

5.3.3 Configuración de operadores en XML

La configuración de operadores, se persiste en un fichero XML para almacenar la información de los operadores que se utilizan comúnmente en los lenguajes de programación pero que difieren en su comportamiento. Se conoce que cada lenguaje tiene definidos sus operadores y sobrecargas, luego para lograr un producto escalable a otros lenguajes, se deben establecer externamente (en un fichero de configuración que la herramienta reconozca) las similitudes y diferencias de todos los operadores.

Uno de los objetivos de esta herramienta es precisamente la independencia de los lenguajes para los que se genere el código. Con esta idea se fundamenta por qué es necesario

mantener las informaciones de los lenguajes excluidas de la lógica interna de la herramienta. XCAD debe interactuar con el entorno, no depender de él.

5.3.4 Generación de código de acceso a datos en XML

Se conoce que persistir información en ficheros XML, es un método efectivo de almacenamiento de datos. Manteniendo ese fundamento, es que la herramienta provee al usuario de un medio para almacenar la información de su trabajo. El desarrollador debe modelar su acceso a datos y este último debe ser almacenado, por eso se genera el fichero XML de la solución. Este fichero contiene los resultados de las transformaciones realizadas por el usuario, ordenados jerárquicamente. Véase, en el siguiente ejemplo, un fichero XML generado por XCAD:

```
<Paquete Nombre="XMLPrueba.xml">
        <Tipos>
                <Tipo Identificador="VACIO"/>
                <Tipo Identificador="int"/>
                <Tipo Identificador="string"/>
                <Tipo Identificador="char"/>
                <Tipo Identificador="FechaNac"/>
                <Tipo Identificador="date"/>
                <Tipo Identificador="Persona"/>
        </Tipos>
        <Definiciones>
                <Tipo Identificador="VACIO" Simple="False">
                        <Propiedades/>
                        <Metodos/>
                </Tipo>
                <Tipo Identificador="int" Simple="True">
                        <Propiedades/>
                        <Metodos/>
                </Tipo>
                <Tipo Identificador="string" Simple="True">
                        <Propiedades/>
                        <Metodos/>
                </Tipo>
                <Tipo Identificador="char" Simple="True">
```

```
                <Propiedades/>
                <Metodos/>
            </Tipo>
            <Tipo Identificador="FechaNac" Simple="False">
                <Propiedades/>
                <Metodos/>
            </Tipo>
            <Tipo Identificador="date" Simple="True">
                <Propiedades/>
                <Metodos/>
            </Tipo>
            <Tipo Identificador="Persona" Simple="False">
                <Propiedades>
                    <Propiedad Nombre="CI" Tipo="string" Visibilidad="Privada"/>
                    <Propiedad Nombre="Direccion" Tipo="string" Visibilidad="Publica"/>
                    <Propiedad Nombre="Nombre" Tipo="string" Visibilidad="Publica"/>
                    <Propiedad Nombre="Apellidos" Tipo="string" Visibilidad="Publica"/>
                    <Propiedad Nombre="FechaNac" Tipo="date" Visibilidad="Publica"/>
                </Propiedades>
                <Metodos>
                    <Metodo Nombre="ObtenerEdad" TipoRetorno="int">
                        <Parametros>
                            <Parametro Nombre="CI" Tipo="string"
Modo="Entrada"/>
                            <Parametro Nombre="Edad" Tipo="int" Modo="Salida"/>
                        </Parametros>
                        <BloqueSentencia/>
                    </Metodo>
                </Metodos>
            </Tipo>
        </Definiciones>
</Paquete>
```

5.4 Conclusiones

En este capítulo se explicaron detalladamente los diagramas de clases utilizados en la implementación y se mostraron descripciones de otros elementos fundamentales en la arquitectura de la herramienta, para su comprensión se incluyeron diagramas y figuras de ayuda gráfica.

CONCLUSIONES

Con la implementación de esta herramienta, se han cumplido los objetivos identificados en la decisión de estandarizar el acceso a datos del proyecto Identidad, disminuyendo así los costos de producción, por la adquisición de licencias y contribuir a mejorar la relación horas/hombre en función de la productividad.

- ✓ Posee una interfaz agradable e intuitiva, altamente interactiva con el usuario.
- ✓ Se permite guardar o cargar cambios en las operaciones con XML.
- ✓ Se genera código en XML.
- ✓ Se reconocen las configuraciones en XML realizadas a los operadores para cada lenguaje.
- ✓ Se generan capas de acceso a datos en varios lenguajes de programación.

RECOMENDACIONES

Para las siguientes versiones de la solución propuesta, se recomienda:

- Implementar las funcionalidades que faltan y que ya están documentadas.
- Agregarle nuevas funcionalidades al producto aprovechando sus características de integración con otros tipos de lenguajes.
- Mejorarle la interfaz visual.
- Validar sus controles.

BIBLIOGRAFÍA CITADA

1. Foundation, Hibernate. [Online] http://es.wikipedia.org/wiki/Hibernate .

2. Foundation, NHibernate. [Online] http://es.wikipedia.org/wiki/NHibernate.

3. Code Generation Network, FireStorm/DAO. [Online] http://www.codegeneration.net/generators-search.php?search=DAO .

4. Foundation, W. Lenguaje Unificado de Modelado. [Online] enero 2007. http://es.wikipedia.org/wiki/Lenguaje_Unificado_de_Modelado.

5. Foundation, Proceso Unificado de Rational. [Online] junio 2007. http://es.wikipedia.org/wiki/Proceso_Unificado_de_Rational.

6. Foundation, FrameWork.NET. [Online] from: http://es.wikipedia.org/wiki/.NET .

7. E.d. Características de C#. [Online] http://www.clikear.com/manuales/csharp/c10.asp.

8. Foundation, Oracle. [Online] http://es.wikipedia.org/wiki/Oracle.

9. Foundation, XSL. [Online] http://es.wikipedia.org/wiki/XSL.

10. Foundation, XSLT. [Online] http://es.wikipedia.org/wiki/XSLT.

11. Software, E.d.p.d.I.d. Flujo de trabajo Captura de requisitos. Universidad de Ciencias Informáticas: Departamento Central de Ingeniería de Software : s.n., 2004.

12. —. Flujo de trabajo Captura de requisitos. Requisitos no funcionales. Universidad de Ciencias Informáticas: Departamento Central de Ingeniería de Software : s.n., 2004.

13. —. Modelo de Negocio. Universidad de Ciencias Informáticas: Departamento Central de Ingeniería de Software : s.n., 2004.

BIBLIOGRAFÍA CONSULTADA

1. Foundation, Hibernate. [Online] http://es.wikipedia.org/wiki/Hibernate .
2. Foundation, NHibernate. [Online] http://es.wikipedia.org/wiki/NHibernate.
3. Code Generation Network, FireStorm/DAO. [Online] http://www.codegeneration.net/generators-search.php?search=DAO .
4. Foundation, W. Lenguaje Unificado de Modelado. [Online] enero 2007. http://es.wikipedia.org/wiki/Lenguaje_Unificado_de_Modelado.
5. Foundation, Proceso Unificado de Rational. [Online] junio 2007. http://es.wikipedia.org/wiki/Proceso_Unificado_de_Rational.
6. Foundation, FrameWork.NET. [Online] from: http://es.wikipedia.org/wiki/.NET .
7. E.d. Características de C#. [Online] http://www.clikear.com/manuales/csharp/c10.asp.
8. Foundation, Oracle. [Online] http://es.wikipedia.org/wiki/Oracle.
9. Foundation, XSL. [Online] http://es.wikipedia.org/wiki/XSL.
10. Foundation, XSLT. [Online] http://es.wikipedia.org/wiki/XSLT.
11. Wesley, Addison. *Data Access Patterns.* 2003. p. 512. 0-13-140157-2.
12. O'Reilly. *XSLT.* 2001. p. 478. 0-596-00053-7.
13. Software, E.d.p.d.I.d. Flujo de Implementación. *Departamento Central de Ingeniería de Software.* Departamento Central de Ingeniería de Software: Universidad de Ciencias Informáticas : s.n., 2005.
14. —. Flujo de Análisis. Modelo de Análisis. Universidad de Ciencias Informáticas: Departamento Central de Ingeniería de Software : s.n., 2004.
15. —. Flujo de trabajo Captura de requisitos. Universidad de Ciencias Informáticas: Departamento Central de Ingeniería de Software : s.n., 2004.
16. —. Flujo de trabajo Captura de requisitos. Requisitos no funcionales. Universidad de Ciencias Informáticas: Departamento Central de Ingeniería de Software : s.n., 2004.
17. —. Modelación de casos de uso del sistema. Universidad de Ciencias Informáticas: Departamento Central de Ingeniería de Software : s.n., 2004.

18. —. Modelo de Negocio. Universidad de Ciencias Informáticas: Departamento Central de Ingeniería de Software : s.n., 2004.

GLOSARIO DE TÉRMINOS

ONIDEX: Oficina Nacional de Identificación y Extranjería.

UCI: Universidad de las Ciencias Informáticas.

Tier Developer 4.0: Herramienta utilizada para generar el acceso a datos.

DBMS: Administrador del sistema de base de datos (Database Management System).

DOM: Modelo de objetos del documento (Document Object Model).

SGML: Standard Generalized Markup Language o Lenguaje estándar genérico por etiquetas, estándar universal para la escritura de archivos de hipertexto en la Internet.

Oracle: Uno de los gestores de base de datos más potentes a nivel mundial.

SAIME: Servicio Autónomo de Identificación Migración y Extranjería.

Portal Web: Es un sitio Web que brinda al usuario la posibilidad de acceder de forma sencilla a un conjunto de recursos y servicios, y en la actualidad son altamente configurables y fáciles de personalizar.

Root: En muchos sistemas operativos el usuario mas privilegiado.

OMG: Object Management Group (Grupo de Gestión de Objetos), es un consorcio dedicado al cuidado y el establecimiento de diversos estándares de tecnologías orientadas a objetos, tales como UML**.**

Software Escalable: Es un software al que es posible agregarle nuevas funcionalidades en futuras versiones sin tener cambiar aspectos claves como la arquitectura del mismo.

Sintaxis: Forma correcta en que deben estar dispuestos los símbolos que componen una instrucción ejecutable por el ordenador.

Metalenguaje: Un metalenguaje es un lenguaje usado para hacer referencia a otros lenguajes. En un aspecto más general, puede referirse a cualquier terminología o lenguaje usado para discutir acerca del mismo lenguaje.

Framework: Conjunto de clases modeladas de forma general para resolver problemas relacionados en un contexto específico.

XML: Lenguaje extensible de marcas, desarrollado por el Consorcio WWW (w3c), es una simplificación y adaptación del SGML para definir la gramática de lenguajes sencillos como HTML.

Dirección IP: Es un número que identifica de manera lógica y jerárquica una interfaz de red de una computadora usando el protocolo del mismo nombre (Internet Protocol o Protocolo de Internet).

Formulario: Interfaz o pantalla de la aplicación que permite la interacción del usuario con esta, para la captura de datos y los eventos que este provoca
Modal: Es un tipo de ventana que se muestra por encima de las demás sin permitir que el usuario vuelva a interactuar con las otras antes de cerrarla.

Excepciones: Errores que ocurren en procesos internos de la aplicación que deben ser tratados para no caer en estados inestables y brindar seguridad y confianza al usuario

TNS: Transparent Network Substrate (Sustrato de Red Transparente), es un protocolo que provee una capa transparente al usuario que permite la conformación de una red homogénea a partir de una red heterogénea.

ANEXOS

ANEXO 1: IMÁGENES DE LA HERRAMIENTA

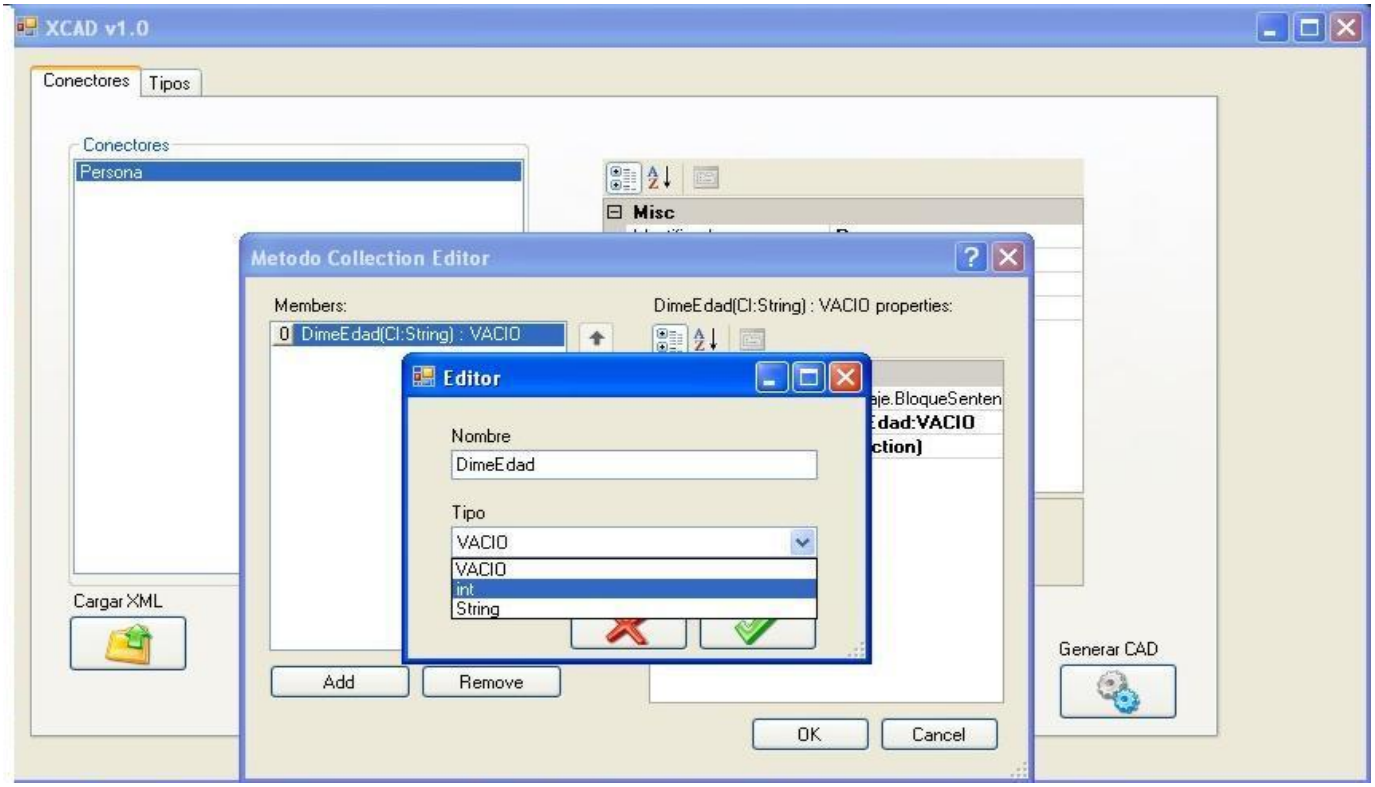

Figura 4.4.1 Se aprecia la edición de un método del conector *Persona.*

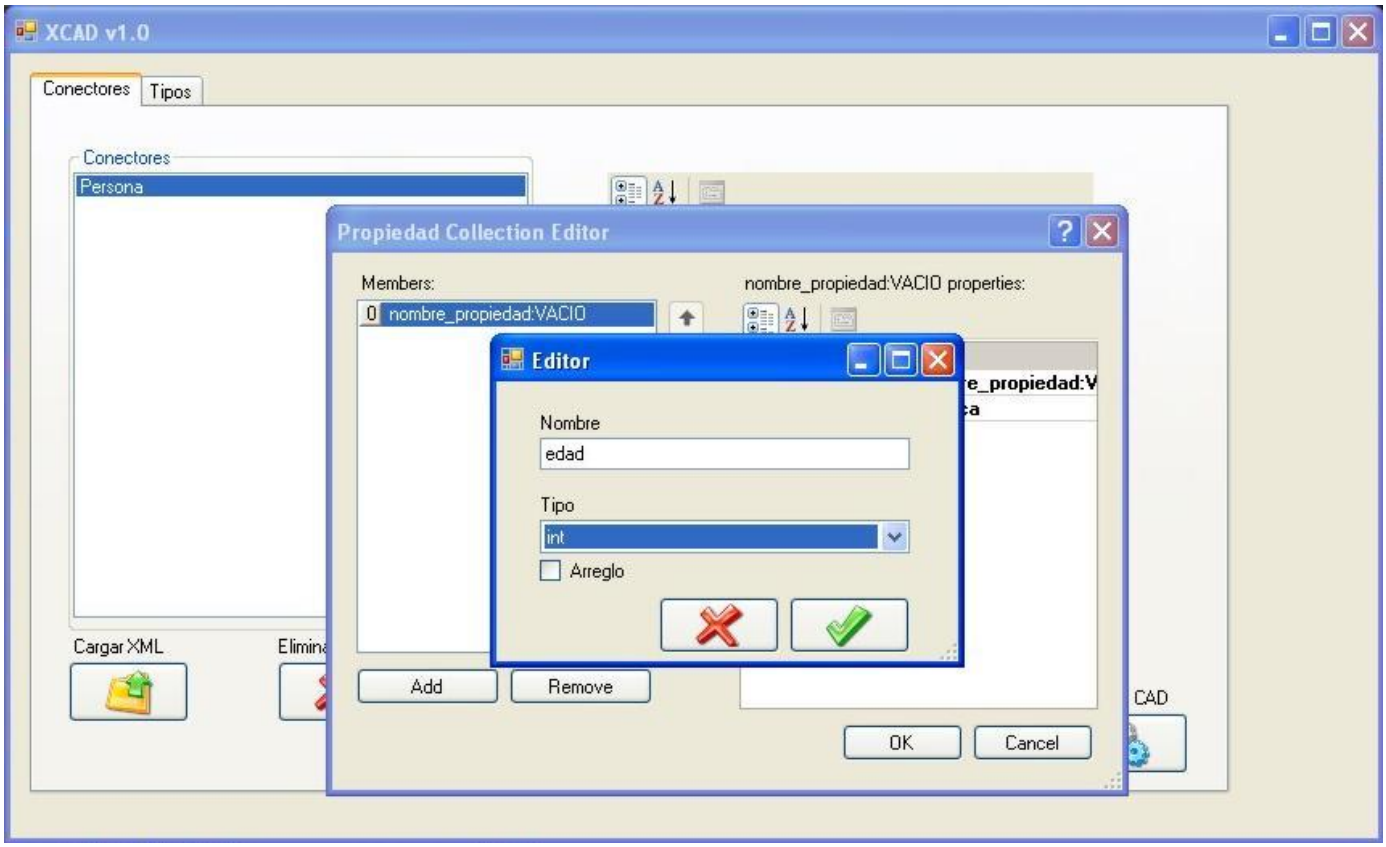

Figura 4.4.2 Se aprecia la edición de una propiedad del conector *Persona.*

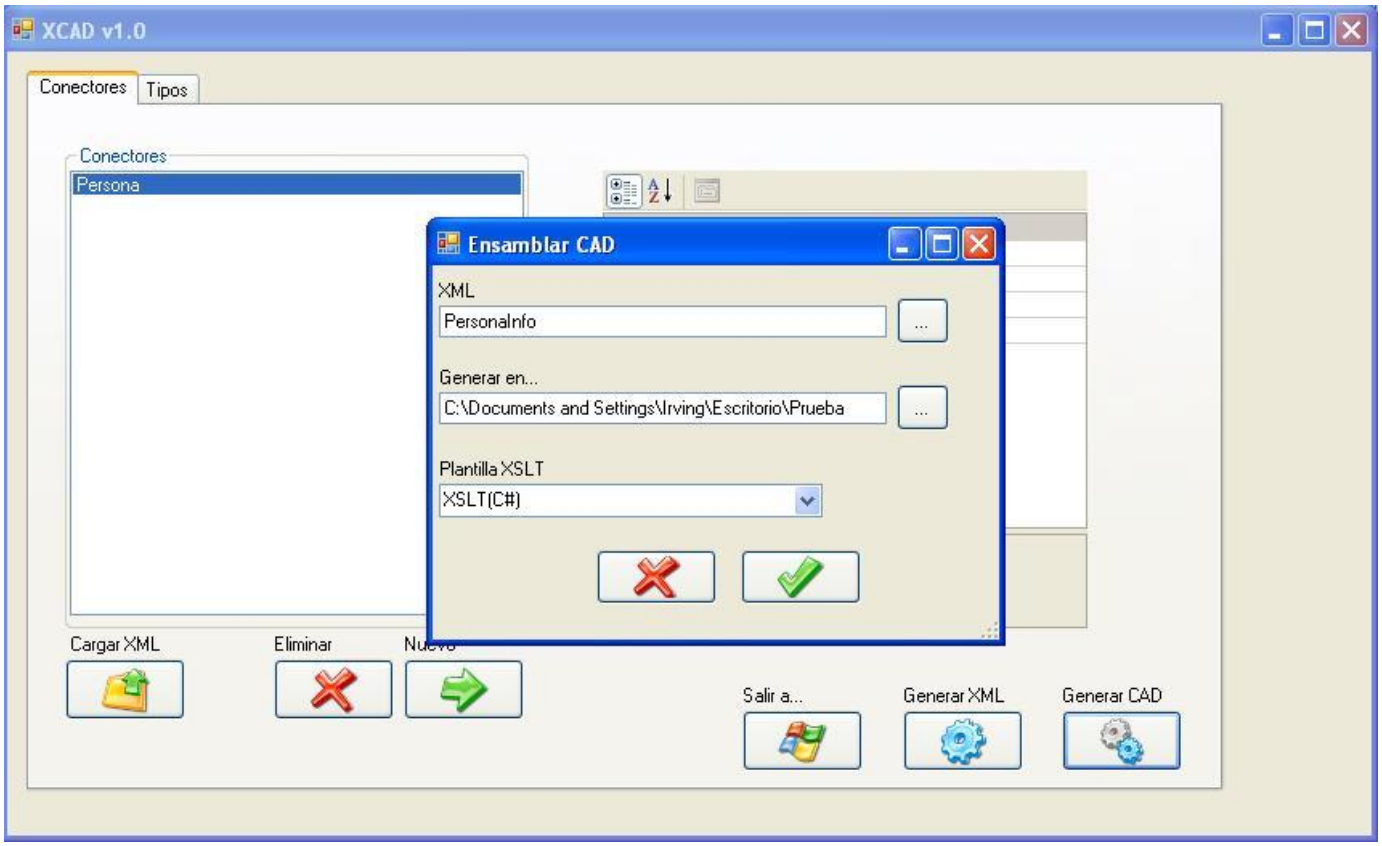

Figura 4.4.3 Pantalla para ensamblar CAD (completada).

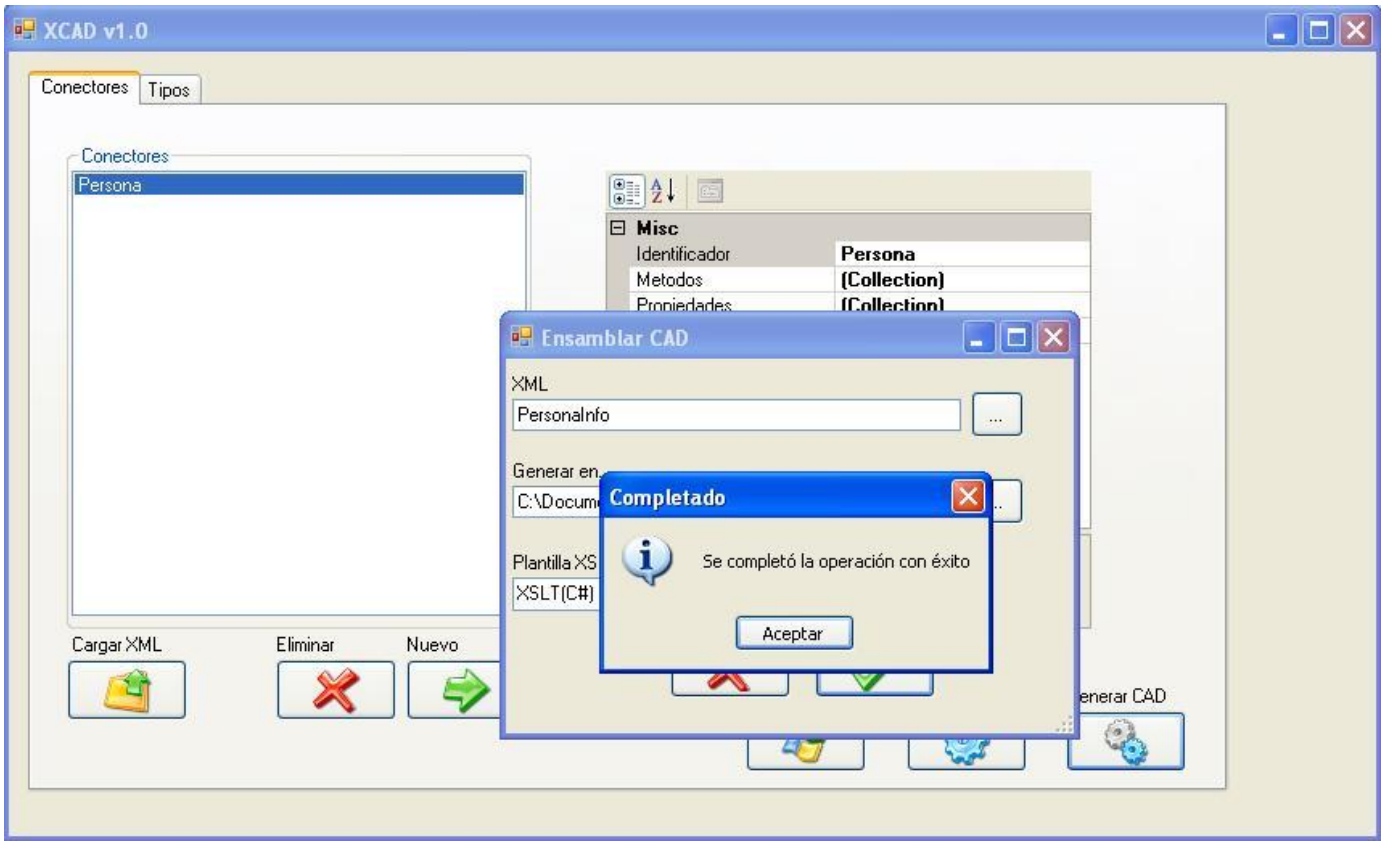

Figura 4.4.4 Notificación al terminar de generar (de manera correcta).