Universidad de las Ciencias Informáticas
FACULTAD 4

Título

ANÁLISIS Y DISEÑO DE UNA HERRAMIENTA PARA MODELACIÓN DE BASES DE DATOS

Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas

Autor

Luis Angel Quintana Santisteban

Tutor

Ing. Alain Eduardo Rodríguez Arias

Ciudad de la Habana, Cuba
Junio, 2008

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos 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 _____.

<Nombre autor>	**<nombre tutor>**
_____	_____
Firma del Autor	Firma del Tutor

DATOS DE CONTACTO

Ing. Alain Eduardo Rodríguez Arias

Email: arod@uci.cu

Graduado en el 2004 del Instituto Politécnico José Antonio Echavarría (CUJAE). Ha ejercido toda su vida laboral en la UCI. Formó parte del proyecto Digitalización de R&N. Actualmente funge como Asesor de Arquitectura y Tecnología de la Facultad 4.

"La esperanza de realizar un sueño, es lo que hace la vida posible."

Paulo Coelho.

DEDICATORIA

A mis abuelos Amador y Dora, por todo lo que siempre han hecho por mí, por la educación que siempre me han dado, por su inmenso cariño y amor hacia mí.

Gracias Abuelita, por los momentos maravillosos que tenemos, por tu sonrisa, por tus consejos que han sido mi guía, por tu amor y cariño hacia mí.

Gracias Abuelo por las infinitas razones que me atan a ti, por esta oportunidad que me brindas y que nos brindas a todos en la familia; sin tu apoyo y sin tu ejemplo no estaría ahora aquí cumpliendo mis sueños que ya son tuyos, TU ERES MI EJEMPLO MAS GRANDE. Tú eres el ejemplo que he seguido y quiero seguir y es a ti a quien quiero premiar con mi esfuerzo, gracias abuelo por sacrificarte y darme todo, pero sobre todo el amor y cariño que siempre me has regalado.

Abuelos les doy un millón de veces las gracias porque no existe otra palabra para agradecer tan maravilloso esfuerzo y dedicación, aun existiendo tal palabra no reflejaría la inmensidad de mi amor hacia ustedes. Sepan siempre que lo que soy hoy es gracias a ti abuelo y abuela y lo que seré el día de mañana también, con mucha honra llevaré el más mínimo de sus recuerdos en mis pensamientos, es lo que me guía y es lo que siempre me guiará. A ustedes abuelos simplemente les debo mi vida.

AGRADECIMIENTOS

Siempre es buen momento de tener un recuerdo agradecido. Son muchos quienes a lo largo del tiempo han dejado su huella en mí. Experiencias sobre las que puedo recordar nombres. Quizás algunos me hayan lanzado a una aventura para sacar algo de mí y despertar mi "yo dormido", conseguir aquello que me parecía imposible, cambiar, crecer y madurar. Agradecer a todas aquellas personas que de una forma u otra participaron en la realización de este trabajo, sale de mi corazón y da un inmenso placer hacerlo. Primeramente quisiera agradecer a mi Comandante en Jefe Fidel, a Raúl, a la generación del Moncada y a todos aquellos heroicos patriotas que desde los mismos inicios de la Revolución dieron sus vidas para que este país fuera lo que es hoy, consagrado, culto, unido y fuerte, simplemente sin esta magnifica revolución nada sería posible para mí y para todos nosotros, Los Cubanos. A mi mamá, por esforzarse tanto conmigo y guiarme siempre por el camino correcto, por todo su cariño y amor. A mi papá por su ejemplo, educación y apoyo en cada etapa de mi vida. A mi hermanito por ser más que eso para mí. A mi abuela Dignora por quererme tanto y estar siempre en mis pensamientos. A Mabelita mi novia por ser como es, por estar siempre unidos en el sentimiento y en el amor, por ser tan linda, dulce y especial. A mi prima Niurka, por sus mágicas palabras y cordial aporte. A mi tia Idian y a Luis por ayudarme y preocuparse tanto por mí en todo este tiempo. A mi tia Silvia por confiar en mi, por sus sentimientos, por ayudarme tanto en toda mi vida. A toda mi familia porque sin ella no todo sería posible. A mi tutor Alain Eduardo por su magnifica guía y apoyo en la realización del presente trabajo, su ayuda fue realmente muy significativa. A mis amigas y amigos del aula y aquellos que transitaron todos estos años junto a mí compartiendo principalmente alegrías. Con el corazón en la mano y con muchos sentimientos agradezco a todas aquellas amistades que puedo recordar, y a las que no también, porque han sido parte importante de mi vida. Sepan siempre que no les podré olvidar pero nunca, nunca. Los quiero.

RESUMEN

Para la realización de nuevos software o la mejora de los ya existentes en el mercado informático, se requiere una estricta organización de todas y cada una de las tareas a ejecutar de forma correcta y eficiente. En el presente trabajo se aborda la realización del análisis y diseño que servirá de base para la implementación de una herramienta para modelado de bases de datos para servidores PostgreSQL que cumpla con las necesidades y expectativas que presenta hoy día la Universidad de las Ciencias Informáticas (UCI), lográndose así que se automaticen los procesos y se faciliten las tareas de coordinación de los diferentes eventos que necesitan ser mejorados en el diseño de las bases de datos. Con vistas a esto se realizó un estudio de oportunidad dentro de la misma universidad para valorar el uso activo de este tipo de herramientas ya sean libres o propietarias y obtener criterios más acertados en cuanto al problema primitivo que se plantea. Se tomaron como base algunas de las herramientas disponibles hoy día en Internet y en la UCI para tales fines (Erwin, Erstudio 2.5, System Architect 4.0, PowerDesigner 6.1, DB Designer 4 y el ERECASE, se analizó cuales serían las funcionalidades a incluir dentro de la propuesta a partir del levantamiento de los requerimientos funcionales y no funcionales. La información obtenida fue debidamente registrada siguiendo la metodología Proceso Unificado de Desarrollo (RUP) y el Lenguaje de Modelado Unificado (UML), además del Visual Paradigm (VP) para realizar el modelado en todo el ciclo de vida del proyecto, específicamente el análisis y diseño, objetivo fundamental del presente trabajo de diploma.

PALABRAS CLAVE

Herramientas Case, Herramienta de modelado de base de datos.

TABLA DE CONTENIDOS

INTRODUCCION

Actualmente la tendencia es hacia un mundo heterogéneo en el cual convivan diversos productos que se complementen, por ello son múltiples las empresas creadoras de software alrededor del mundo que compiten por el dominio del mercado informático, para lograr tal objetivo muchas de estas empresas desarrollan herramientas abiertas con conectividad a diversas plataformas, basadas en tecnología orientada a objetos y que permitan la reutilización del software; de este modo, se han extendido a la adquisición de herramientas CASE (Computer Aided Software Engineering, Ingeniería Asistida por Computadora) con el fin de automatizar los aspectos clave de todo lo que implica el proceso de desarrollo de un sistema e incrementar su posición en el mercado competitivo. Sin embargo, en algunos casos se obtienen elevados costos tanto en la adquisición de herramientas, o la falta de adaptación de tal herramienta a la arquitectura de la información y a metodologías de desarrollo utilizadas por alguna empresa, organización o institución. Por otra parte, algunas herramientas CASE no ofrecen o evalúan soluciones potenciales para los problemas relacionados con sistemas o virtualmente no llevan a cabo ningún análisis de los requerimientos de la aplicación. Sin embargo, CASE proporciona un conjunto de herramientas semiautomatizadas y automatizadas que están desarrollando una cultura de ingeniería nueva para muchas empresas. La industria del software cubano en conjunto con la UCI atendiendo las constantes transformaciones en la producción de software y sus altos costos de adquisición en el mercado internacional trabajan en la creación de sus propios productos informáticos, lográndose así que se cumplan los objetivos trazados, y uno de ellos es el de sustituir herramientas propietarias que son de mucho uso hoy en día en nuestras empresas e instituciones por otras herramientas para los mismos propósitos pero que sean de fácil acceso y mejor aun que su código fuente esté abierto, todo esto implica grandes beneficios para Cuba, país que se encuentra en vía de desarrollo, o para un usuario en particular ya que el gasto en la compra del producto informático sería mínimo o prácticamente gratuito, por otra parte se tendría el código fuente para futuras transformaciones o transformaciones inmediatas que posibles usuarios, instituciones o empresas cubanas requieran.

SITUACION PROBLEMICA

La UCI no cuenta con una herramienta propia de modelación y generación de bases de datos por lo que actualmente se estudia la posibilidad de crear y/o mejorar las herramientas ya existentes para la obtención de nuevos productos más económicos acorde a los requerimientos de los usuarios y con la calidad que se exigen por parte de las entidades nacionales e internacionales.

PROBLEMA CIENTIFICO

¿Cómo diseñar una herramienta para modelación de bases de datos?

OBJETIVO GENERAL

Analizar y Diseñar una herramienta para la modelación y generación de bases de datos que cumpla con las necesidades que presenta la Facultad 4.

OBJETIVOS ESPECIFICOS

- Investigar sobre el uso de herramientas libres y propietarias destinadas a la modelación y generación de bases de datos en la UCI.
- Estudiar herramientas para modelado y generación de bases de datos, como el Erwin, Erstudio 2.5, System Architect 4.0, PowerDesigner 6.1, DB Designer 4 y el ERECASE.
- Identificar las funcionalidades a incluir dentro de la herramienta y a partir de las mismas levantar todos los requerimientos tanto funcionales como no funcionales.
- Realizar el análisis y diseño de la herramienta a desarrollar.
- Documentar adecuadamente el proceso de desarrollo del análisis y diseño de la herramienta siguiendo la metodología RUP.

CAMPO DE ACCION

Todos los proyectos de la facultad 4 que necesiten este tipo de herramientas libres para la modelación y generación de bases de datos.

TAREAS A DESARROLLAR PARA CUMPLIR LOS OBJETIVOS

- Estudiar las últimas tendencias y tecnologías que se utilizan para diseñar una herramienta CASE.
- Seleccionar el proceso de desarrollo de software a Utilizar.
- Realización del levantamiento de requisitos de la herramienta a analizar y diseñar.
- Realizar el análisis y diseño del sistema utilizando el proceso de desarrollo de software seleccionado.

CAPITULO 1: FUNDAMENTACION TEORICA.

1.1 INTRODUCCION

En este capítulo se aborda la metodología de desarrollo utilizada para la presentación mediante el lenguaje de modelado (UML) del análisis y diseño de una herramienta modelación de bases de datos. Se listan los beneficios que proveen las herramientas CASE en el proceso de desarrollo de software, así como la descripción de sus componentes y funcionalidades. Se seleccionan, comparan y se caracterizan algunas de las herramientas utilizadas en la UCI para tal propósito, como el ERWIN, ER/STUDIO 2.5, DATA ARCHICHET, POWER DESIGNER 6.1, DB Designer 4 y el ERECASE ya que estas herramientas presentan funcionalidades muy importantes por lo que fue necesario estudiarlas para obtener una mejor visión sobre la propuesta de la herramienta a diseñar. También se caracteriza la plataforma y lenguaje de programación en que posteriormente será implementada la herramienta atendiendo a las más modernas y adecuadas tecnologías para desarrollar software.

1.2 FUNDAMENTACION DE LA METODOLOGIA A UTILIZAR.

Un proceso define quién está haciendo qué, cuándo y cómo alcanzar un determinado objetivo. En la Ingeniería del software el objetivo es construir un producto software o mejorar uno. (1) Un proceso efectivo proporciona normas para el desarrollo eficiente de software de calidad y presenta las mejores prácticas que el estado actual que la tecnología permite; en consecuencia, reduce el riesgo y hace el proyecto más predecible. La rama de la metodología, dentro de la ingeniería de software, se encarga de elaborar estrategias de desarrollo de software; centradas en las personas o los equipos, orientadas hacia la funcionalidad y la entrega, de comunicación intensiva y que requieren implicación directa del cliente. Su objetivo es elevar la calidad del software (en todas las fases por las que pasa) a través de una mayor transparencia y control sobre el proceso. Todo desarrollo de software es riesgoso y difícil de controlar, pero si utilizamos un proceso o una metodología de desarrollo, entonces obtendremos clientes y desarrolladores más satisfechos con el resultado.

1.2.1 RATIONAL UNIFIED PROCESS (RUP)

Proceso Unificado Rational (Rational Unified Process) es una metodología pesada, de IBM Rational para el desarrollo y construcción de software basado íntegramente en UML como soporte a la metodología. El Proceso Unificado es más que un simple proceso; es un marco de trabajo genérico que puede especializarse para una gran variedad de sistemas software, para diferentes áreas de

aplicación, diferentes tipos de organizaciones, diferentes niveles de aptitud y diferentes tamaños de proyecto. (1) RUP establece las actividades y los criterios para conducir un sistema desde su máximo nivel de abstracción (la idea del cliente), hasta su nivel más concreto (un programa ejecutándose). Además soporta las técnicas orientadas a objetos.

RUP divide en 4 fases el desarrollo del software:

- Fase Inicio, tiene como objetivo determinar la visión del proyecto.
- Fase Elaboración, el objetivo es determinar la arquitectura óptima.
- Fase Construcción, el objetivo es llevar a obtener la capacidad operacional inicial.
- Fase Transición, el objetivo es llegar a obtener el release del proyecto.

RUP implementa las siguientes mejores prácticas asociadas al proceso de Ingeniería de Software:

- Desarrollo Iterativo
- Manejo de los Requerimientos
- Uso de una Arquitectura basada en componentes
- Modelación Visual
- Verificación Continua de la Calidad
- Manejo de los Cambios

Principales características son:

- Forma disciplinada de asignar tareas y responsabilidades.
- Desarrollo basado en componentes.
- Utilización de un único lenguaje de modelación.
- Proceso Integrado Único.
- Divide el proceso de desarrollo en ciclos.
- Es un proceso Iterativo e Incremental: particiona el ciclo de vida en iteraciones que producen versiones increméntales de los ejecutables de la aplicación.
- Dirigido por los Casos de Uso: los requerimientos funcionales son expresados en la forma de Casos de Uso, que guían la realización de una arquitectura ejecutable de la aplicación.

- Centrado en la Arquitectura: el proceso focaliza el esfuerzo del equipo en construir los elementos críticos estructuralmente y del comportamiento (llamados Elementos Arquitecturales) antes de construir elementos menos importantes.

1.2.2 RUP COMO METODOLOGIA A UTILIZAR.

Se utiliza RUP porque este proceso es el resultado de varios años de desarrollo y uso práctico, desde el Método de Ericsson pasando por el Proceso Objectory, y por el Proceso Objectory de Rational hasta el Proceso Unificado del Rational. Él utiliza UML como lenguaje de modelado. RUP como proceso define quién (Trabajadores) está haciendo qué (Artefactos), cuándo (Flujo de actividades) y cómo (Actividades) para alcanzar un determinado objetivo (Producto). En RUP se definen 9 flujos de trabajo principales y 4 fases de desarrollo. Los 6 primeros son conocidos como flujos de ingeniería y los tres últimos como de apoyo. En la figura se representa el proceso en el que se grafican los flujos de trabajo y las fases y muestra la dinámica expresada en iteraciones y puntos de control. (2)

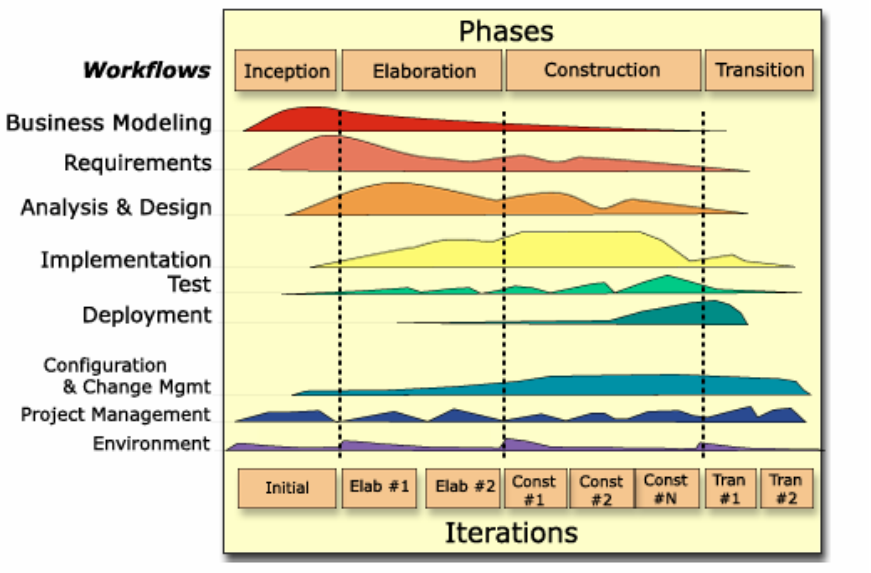


Fig. 1 Flujos de trabajo y fases

Los flujos de trabajo que se tratarán en el presente trabajo de diploma son:

- Modelamiento del negocio: Describe los procesos de negocio, identificando quiénes participan y las actividades que requieren automatización.
- Requerimientos: Define qué es lo que el sistema debe hacer, para lo cual se identifican las funcionalidades requeridas y las restricciones que se imponen.
- Análisis y diseño: Describe cómo el sistema será realizado a partir de la funcionalidad prevista y las restricciones impuestas (requerimientos), por lo que indica con precisión lo que se debe programar.

1.2.3 PROGRAMACION EXTREMA (XP)

Es una de las metodologías de desarrollo de software utilizadas para proyectos de corto plazo y corto equipo. La metodología consiste en una programación rápida o extrema, cuya particularidad es tener como parte del equipo, al usuario final, pues es uno de los requisitos para llegar al éxito del proyecto.
La metodología se basa en:

- Pruebas Unitarias: Son las pruebas realizadas a los principales procesos, de tal manera que, se puedan hacer pruebas de las fallas que pudieran ocurrir.

- Refabricación: Se basa en la reutilización de código, para lo cual se crean patrones o modelos estándares, siendo más flexible al cambio.

- Programación en pares: Una particularidad de esta metodología es que propone la programación en pares, la cual consiste en que dos desarrolladores participen en un proyecto en una misma estación de trabajo. Cada miembro lleva a cabo la acción que el otro no está haciendo en ese momento. (3)

1.2.4 FUNDAMENTACIÓN DE LA METODOLOGIA A UTILIZAR

A partir de las características de las metodologías establecidas anteriormente, se decidió utilizar como metodología RUP, para el control y planificación de este trabajo, por sus características y las facilidades que aporta. Pues la metodología XP no cumple con los objetivos de este trabajo, al ser solo

recomendable emplearla en proyectos a corto plazo. Además maneja con poco rigor el análisis y diseño y genera poca documentación.

1.3 HERRAMIENTAS PARA MODELADO

La misión de cualquier herramienta CASE, que utiliza UML como notación para elaborar los modelos, es comunicar, de la manera más eficiente posible, a los agentes del proyecto, todas aquellas decisiones que se toman con respecto a la arquitectura del sistema en discusión y que son determinantes para cumplir con los objetivos de las distintas fases de un proyecto.

1.3.1 RATIONAL ROSE ENTERPRISE SUITE

El Rational Rose, es una herramienta CASE desarrollada por Rational Corporation, basada en UML, que permite crear los diagramas que se van generando durante el proceso de Ingeniería en el desarrollo del software. Las personas que desarrollaron RUP son miembros de Rational Corporation, por lo que el mismo es completamente compatible con esta metodología, brinda muchas facilidades en la generación de la documentación del software que se está desarrollando, además posee un gran número de estereotipos predefinidos que facilitan el proceso de modelación del software. Dicha herramienta es capaz de generar el código fuente de las clases definidas en el flujo de trabajo de diseño, pero tiene la limitación de que aún hay varios lenguajes de programación que no soporta o que solo lo hace a medias. Por otra parte, una vez que se tiene el diagrama de clases persistentes a partir del cual se genera la base de datos del sistema, no existe la posibilidad de exportar ese modelo hacia algún sistema gestor de bases de datos. (4)

1.3.2 VISUAL PARADIGM, SOFTWARE UTLIZADO PARA MODELAR EL SISTEMA.

Es un software privativo gratuito para modelado en UML. Esta herramienta tiene unas características gráficas muy cómodas, que facilitan la realización de los diagramas de modelado que sigue el estándar de UML, los mismo son: Diagramas de clase, Casos de Uso, Comunicación, Secuencia, Estado, Actividad, Componentes, etc.

Se integra con las siguientes herramientas Java:

- Eclipse/IBM WebSphere
- JBuilder
- NetBeans IDE
- Oracle JDeveloper
- BEA Weblogic

Visual Paradigm ofrece:

- Entorno de creación de diagramas para UML 2.0
- Diseño centrado en casos de uso y enfocado al negocio que genera un software de mayor calidad
- Uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación.
- Capacidades de ingeniería directa e inversa.
- Modelo y código que permanece sincronizado en todo el ciclo de desarrollo.
- Disponibilidad de múltiples versiones, para cada necesidad.
- Disponibilidad en múltiples plataformas.

Características:

- Producto de calidad.
- Soporta aplicaciones web.
- Las imágenes y reportes generados, no son de muy buena calidad.
- Varios idiomas.
- Generación de código para Java y exportación como HTML.
- Fácil de instalar y actualizar.
- Compatibilidad entre ediciones.

1.3.3 SELECCION DE LA HERRAMIENTA

Como herramienta de modelado se seleccionó Visual Paradigm pues permite la integración con el Eclipse lo que facilitará la futura implementación de la aplicación.

1.4 LENGUAJE DE MODELACION

El Lenguaje Unificado de Modelado (UML) es un lenguaje para la especificación, la visualización, la construcción y la documentación de los artefactos de los sistemas de software y también para otros tipos de sistemas. Representa una colección de las mejores prácticas de ingeniería que han sido probadas con éxito en el modelado de sistemas grandes y complejos. (5) Se convirtió en estándar del Object Management Group (OMG) en 1997, después de tres años de trabajo, como consecuencia de la llamada Guerra de las Metodologías. (6)

La principal ventaja de UML es ser un lenguaje de propósito general, aunque esto en ocasiones se puede convertir en una desventaja, porque no se pueden representar cabalmente las situaciones o características propias de dominios específicos. (7) Es un lenguaje gráfico, que puede ser usado en todas las fases de desarrollo de software y que permite representar los sistemas con varios modelos parciales, lo que facilita su entendimiento y la comunicación.
Por todo lo anterior, cada vez más proyectos utilizan UML para representar la arquitectura de sus sistemas. (8)

1.5 ¿QUE SON LAS HERRAMIENTAS CASE?

De acuerdo con Kendall y Kendall la ingeniería de sistemas asistida por ordenador es la aplicación de tecnología informática a las actividades, las técnicas y las metodologías propias de desarrollo, su objetivo es acelerar el proceso para el que han sido diseñadas, en el caso de CASE para automatizar o apoyar una o mas fases del ciclo de vida del desarrollo de sistemas.

Cuando se hace la planificación de la base de datos, la primera etapa del ciclo de vida de las aplicaciones de bases de datos, también se puede escoger una herramienta CASE que permita llevar a cabo el resto de tareas del modo más eficiente y efectivo posible. Una herramienta CASE suele incluir:

- Un diccionario de datos para almacenar información sobre los datos de la aplicación de bases de datos.
- Herramientas de diseño para dar apoyo al análisis de datos.
- Herramientas que permitan desarrollar el modelo de datos corporativo, así como los esquemas conceptual y lógico.

- Herramientas para desarrollar los prototipos de las aplicaciones.

El uso de las herramientas CASE puede mejorar la productividad en el desarrollo de una aplicación de bases de datos. (9)

1.5.1 TECNOLOGÍA CASE

La tecnología CASE supone la automatización del desarrollo del software, contribuyendo a mejorar la calidad y la productividad en el desarrollo de sistemas de información y se plantean los siguientes objetivos:

- Permitir la aplicación práctica de metodologías estructuradas, las cuales al ser realizadas con una herramienta se consigue agilizar el trabajo.
- Facilitar la realización de prototipos y el desarrollo conjunto de aplicaciones.
- Simplificar el mantenimiento de los programas.
- Mejorar y estandarizar la documentación.
- Aumentar la portabilidad de las aplicaciones.
- Facilitar la reutilización de componentes software.
- Permitir un desarrollo y un refinamiento visual de las aplicaciones, mediante la utilización de gráficos.

Automatizar:

- El desarrollo del software
- La documentación
- La generación del código
- El chequeo de errores
- La gestión del proyecto

Permitir:

- La reutilización del software
- La portabilidad del software

- La estandarización de la documentación

1.6 CLASIFICACION DE LAS HERRAMIENTAS CASE.

No existe una única clasificación de herramientas CASE y, en ocasiones, es difícil incluirlas en una clase en común. Podrían clasificarse así:

- Las plataformas que soportan.
- Las fases del ciclo de vida del desarrollo de sistemas que abarca.
- La arquitectura de las aplicaciones que produce.
- Su funcionalidad. Las herramientas CASE, en función de las fases del ciclo de vida que cubre, se pueden agrupar de la forma siguiente:

1. Herramientas integradas, I-CASE (Integrated CASE, CASE integrado): abarcan todas las fases del ciclo de vida del desarrollo de sistemas. Son llamadas también CASE workbench.

2. Herramientas de alto nivel, U-CASE (Upper CASE - CASE superior), orientadas a la automatización y soporte de las actividades desarrolladas durante las primeras fases del desarrollo: análisis y diseño.

3. Herramientas de bajo nivel, L-CASE (Lower CASE - CASE inferior), dirigidas a las últimas fases del desarrollo: construcción e implantación.

4. Juegos de herramientas o Tools-Case, son el tipo más simple de Herramientas CASE. Automatizan una fase dentro del ciclo de vida. Dentro de este grupo se encontrarían las herramientas de reingeniería, orientadas a la fase de mantenimiento. (9)

1.7 BENEFICIOS QUE PROVEEN LAS HERRAMIENTAS CASE EN EL PROCESO DE DESARROLLO DE SOFTWARE.

1.7.1 FACTIBILIDAD PARA LA REVISION DE APLICACIONES.

La experiencia muestra que una vez que las aplicaciones se implementan, se emplean por mucho tiempo. Las herramientas CASE proporcionan un beneficio substancial para las organizaciones al

facilitar la revisión de las aplicaciones. Contar con un depósito central agiliza el proceso de revisión ya que éste proporciona bases para las definiciones y estándares para los datos. Las capacidades de generación interna, si se encuentran presentes, contribuyen a modificar el sistema por medio de las especificaciones más que por los ajustes al código fuente.

1.7.2 SOPORTE PARA EL DESARROLLO DE PROTOTIPOS DEL SISTEMA.

En general, el desarrollo de prototipos de aplicaciones toma varias formas. En ocasiones se desarrollan diseños para pantallas y reportes con la finalidad de mostrar la organización y composición de los datos, encabezados y mensajes. Los ajustes necesarios al diseño se hacen con rapidez para alterar la presentación y las características de la interface. Sin embargo, no se prepara el código fuente, de naturaleza orientada hacia procedimientos, como una parte del prototipo.

Como disyuntiva, el desarrollo de prototipos puede producir un sistema que funcione. Las características de entrada y salida son desarrolladas junto con el código orientado hacia los procedimientos y archivos de datos.

Muchas herramientas CASE soportan las primeras etapas del desarrollo del prototipo. Muy pocas brindan apoyo durante todo el proceso de desarrollo del prototipo. Las que proporcionan la capacidad para generar código soportan de hecho todo proceso, ya que el código puede ser generado al inducir la actividad de generación después de cambiar las especificaciones o requerimientos.

1.7.3 GENERACION DE CODIGO.

Como ya se mencionó, algunas herramientas CASE tienen la capacidad de producir el código fuente. La ventaja más visible de esta característica es la disminución del tiempo necesario para preparar un programa. Sin embargo, la generación del código también asegura una estructura estándar y consistente para el programa (lo que tiene gran influencia en el mantenimiento) y disminuye la ocurrencia de varios tipos de errores, mejorando de esta manera la calidad. Las características de la generación del código permiten volver a utilizar el software y las estructuras estándares para generar dicho código, así como el cambio de una especificación modular, lo que significa volver a generar el código y los enlaces con otros módulos. Ninguna de las herramientas que existen en el presente es capaz de generar un código completo en los dominios.

1.7.4 MEJORA EN LA HABILIDAD PARA SATISFACER LOS REQUERIMIENTOS DE USUARIO.

Es bien conocida la importancia de satisfacer los requerimientos del usuario, ya que esto guarda relación con el éxito del sistema. De manera similar, tener los requerimientos correctos mejora la calidad de las prácticas de desarrollo. Parece ser que las herramientas CASE disminuyen el tiempo de desarrollo, una característica que es importante para los usuarios. Las herramientas afectan la naturaleza y cantidad de interacción entre los encargados del desarrollo y el usuario. Las descripciones gráficas y los diagramas, así como los prototipos de reportes y la composición de las pantallas, contribuyen a un intercambio de ideas más efectivo.

1.7.5 SOPORTE INTERACTIVO PARA EL PROCESO DE DESARROLLO.

La experiencia ha demostrado que el desarrollo de sistemas es un proceso interactivo. Las herramientas CASE soportan pasos interactivos al eliminar el tedio manual de dibujar diagramas, elaborar catálogos y clasificar. Como resultado de esto, se anticipa que los analistas repasarán y revisarán los detalles del sistema con mayor frecuencia y en forma más consistente.

1.8 FORMAS PARA MODELAR LAS BASES DE DATOS.

Las que se basan en el modelo de datos persistentes.

El modelado de la información persistente constituye unos de los factores importantes cuando se está construyendo un sistema de información. En términos prácticos dicha persistencia estará soportada en bases de datos relacionales. Es por esto que las herramientas CASE como es el caso del Rational Rose y el Visual Paradigm le deben permitir al usuario facilidades para definir el esquema relacional, bien sea a través de una interfaz para construccion de diagramas entidad-relacion o modelos de datos, o a través de la extensión de UML, modelando clases persistentes que se pueden implementar en motores relacionales. Los modelos que se construyan deben prever la facilidad de generar scripts SQL, con las instrucciones DDL, para la creación de los objetos de las bases de datos.

Las que se basan en el modelo relacional.

El modelo relacional para la gestión de una base de datos es un modelo de datos basado en la lógica de predicado y en la teoría de conjuntos. Es el modelo más utilizado en la actualidad para modelar problemas reales y administrar datos dinámicamente. Tras ser postuladas sus bases en 1970 por Edgar Frank Codd, de los laboratorios IBM en San José (California), no tardó en consolidarse como un nuevo paradigma en los modelos de base de datos. Su idea fundamental es el uso de relaciones. Estas relaciones podrían considerarse en forma lógica como conjuntos de datos llamados tuplas. Pese a que ésta es la teoría de las bases de datos relacionales creadas por Edgar Frank Codd, la mayoría de las veces se conceptualiza de una manera más fácil de imaginar, esto es, pensando en cada relación como si fuese una tabla que está compuestas por registros (cada fila de la tabla sería un registro), que representarían las tuplas y campos (las columnas de una tabla), las herramientas CASE como el ERWIN 3.0, ER/STUDIO 2.5, SYSTEM ARCHITECT 4.0, POWERDESIGNER 6.1, se apoyan en el modelado relacional.

1.9 COMPARACION DEL ERWIN 3.0, ER/STUDIO 2.5, SYSTEM ARCHITECT 4.0, POWERDESIGNER 6.1

1.9.1 CARACTERISTICAS GENERALES.

ERWIN 3.0

Erwin es una herramienta para modelar, que ayuda a diseñar bases de datos de alto desempeño para cliente/servidor y *web/intrane*t, así como aplicaciones de *data warehousin*g. La herramienta Erwin no solo ayuda a diseñar modelos de datos lógicos, también construye automáticamente estructuras de datos físicos con la información del diagrama. Cuando el modelo de datos esta listo para usarse, simplemente se selecciona el servidor donde se quiere construir la base de datos y se eligen las opciones de generación de esquema que se quieran incorporar. En minutos, Erwin automáticamente construye la base de datos física, incluyendo todas las tablas, índices, procedimientos almacenados, triggers de integridad referencial y otros componentes necesarios para manejar exitosamente los datos usados en la organización.

ER/STUDIO 2.5

Es una herramienta de modelado de datos fácil de usar y multinivel, para el diseño y construcción de bases de datos a nivel físico y lógico. Direcciona las necesidades diarias de los administradores de bases de datos, desarrolladores y arquitectos de datos que construyen y mantienen aplicaciones de bases de datos grandes y complejos. ER/Studio está equipado para crear y manejar diseños de bases de datos funcionales y confiables. Ofrece fuertes capacidades de diseño lógico, sincronización bidireccional de los diseños físicos y lógicos, construcción automática de bases de datos, documentación y fácil creación de reportes.

SYSTEM ARCHITECT 4.0

System Architect es una herramienta poderosa de modelado estructurado de datos, tiene la capacidad de identificar y clasificar personal para autorizar su entrada al sistema. Los usuarios de red trabajan en un diagrama de proyecto y una llave de registro de diccionario de datos. Define propiedades para cualquier entrada de diccionario, incluyendo definiciones, símbolos y diagramas. Construye ligas entre varios objetos del diccionario. Especifica y define requerimientos, planes de prueba, cambio de requerimientos, objetivos de negocios, metas, así como símbolos o grupo de símbolos que son afectados.

POWERDESIGNER 6.1

Es una herramienta para crear bases de datos y aplicaciones cliente/servidor basadas o no en *We*b. Permite a los diseñadores de aplicaciones complejas de cliente/servidor tener una descripción general de los procesos particulares para comprender mejor a la organización. Exporta información del modelo físico y extiende atributos al diccionario de 4GL. Importa atributos extendidos de PowerBuilder. Soporta definición de atributos extendidos para PowerBuilder, Progress, Uniface, PowerHouse, Axiant, y NS-DK.

1.9.2 DIAGRAMAS

ERWIN 3.0

Los diagramas de modelos de datos en Erwin se usan para generar o actualizar bases de datos relacionales, o sea, sistemas de almacenamiento de datos ordenados en tablas. Cuando se crea un

diagrama Erwin, el modelo de la información se representa por entidades (gente, lugares y cosas), atributos (hechos acerca de una entidad, tales como nombre de la persona, dirección, edad, etc.), y relaciones entre entidades. Cada entidad corresponde a una tabla en la base de datos, con instancias de entidades que corresponden a los renglones de la tabla y atributos de entidades correspondientes a encabezados de columnas. Las relaciones, usadas por DBMS *(data base management syste*m) para ligar renglones de datos en tablas diferentes, están representadas como frases verbales en una línea conectando a dos entidades. Cuando se actualiza una base de datos física, Erwin automáticamente genera un *script* de definición de datos SQL, para crear tablas de bases de datos, incluyendo llaves, constraints y códigos *trigger* SQL para reforzar la integridad referencial entre tablas relacionadas.

ERSTUDIO 2.5

La creación de diagramas es clara y rápida. Tiene la posibilidad de realizar diagramas con desempeño rápido. También es posible cambiar el estilo de las líneas, los colores, tipos de letra, niveles de acercamiento, y modelos de despliegue. Es posible crear subvistas para separar y manejar áreas importantes. ER/Studio automáticamente mantiene todas las dependencias entre subvistas y el diagrama completo. El *Explorer Navigatior* facilita el trabajo hasta con los diagramas más grandes. Se usa el Explorer para encontrar y seleccionar entidades. Un solo click inmediatamente enfoca una ventana de diagrama.

SYSTEM ARCHITECT 4.0

El área de dibujo de diagramas puede ser del tamaño de 64" x 60" a 50" x 150". Es posible ver los diagramas en las siguientes opciones: tamaño actual, página completa, área usada, o porcentaje de reducción. Permite la edición de un diagrama en cualquier modo de vista, seleccionar y mover objetos individualmente o usando el ratón para obtener la porción del diagrama que se desee, y cambiar el tamaño objetos individuales proporcionalmente o no proporcionalmente usando el ratón. La herramienta *Leveling Automatically* nivela diagramas y usa un mecanismo simple para cambiar la herencia en cualquier dirección. Automáticamente crea *Decomposition Diagrams* (Diagramas descompuestos) de la herencia del *Data Flow Diagrams* (Diagramas de flujo de datos).

POWERDESIGNER 6.1

PowerDesigner cuenta con herramientas para la creación y control de diagramas como son: *Off-page Connecto*r; que representa los flujos de entradas y salidas en un proceso, *Business Rules* que define las reglas de uso para Procesos, Almacenamiento de datos, Entidades externas, y Flujos de datos; y CRUD Matrix, que define el efecto de un proceso de datos en términos de Crear, Leer, Actualizar, y Borrar operaciones (CRUD).

1.9.3 AYUDA

ERWIN 3.0

La herramienta de Erwin, *Workspace* contiene una ventana de diagrama donde se crea el diagrama del modelo de datos y provee varias herramientas de cliente usadas en el proceso de modelado. Algunas recomendaciones para facilitar a cualquier primerizo su uso:

- Hacer click en la barra de menú de Erwin para conocer información de cómo usar los menús y editores de Erwin.
- Hacer click en la barra de herramientas para obtener rápidamente una idea de cuanto se puede hacer en Erwin sin siquiera abrir un menú.
- Hacer click en la barra de herramientas de color y tipo de letra para ver como se cambia el texto y los colores usados en un diagrama rápidamente.
- Hacer click en la caja de herramientas para aprender más acerca de cada símbolo usado en un diagrama Erwin.
- Hacer click en el *Subject Area List Box* en el lado derecho de la barra de herramientas para aprender como usar las áreas de sujeto Erwin para subdividir un modelo de datos grande.

ERSTUDIO 2.5

Ya sea que se inicie un nuevo diseño o se mantenga uno existente, ER/Studio está equipado con elementos de ayuda para hacer el trabajo de manera efectiva. Las barras de herramientas tienen

algunas sugerencias para el uso de las mismas, además de contar con ayuda en línea sensible al contexto.

SYSTEM ARCHITECT 4.0

La ayuda en línea de System Architect es extensiva, e incluye tutoriales del modelado de datos, BPR, análisis estructurado.

POWERDESIGNER 6.1

La ayuda de Power Designer es sensible y adecuada al contexto.

1.9.4 ESQUEMA DE BASE DE DATOS

ERWIN 3.0

Para diseñar un modelo de datos, Erwin proporciona propiedades específicas de DBMS y del servidor de editores que permiten definir nombres físicos y propiedades para las tablas, columnas y relaciones que genera cuando se crea el esquema de la base de datos. Cuando Erwin crea un esquema de bases de datos, genera un *script* de cliente DDL *(data definition languag*e) usando la sintaxis correcta de SQL para el servidor seleccionado. Se puede ver el código que genera Erwin y, si se desea, se puede modificar antes de que se cree la base de datos. Si el servidor soporta elementos avanzados como procedimientos almacenados y *triggers* de integridad referencial, Erwin proporciona editores de plantillas especiales y macros para guardar la información en tiempos determinados para acelerar la creación de estos objetos en el servidor. En varios servidores, se pueden hasta crear objetos de almacenamiento físico como espacios de tablas y segmentos de enrolamiento de Erwin así como especificar la ubicación y parámetros de almacenamiento para las tablas de bases de datos que Erwin genera en el servidor.

ERSTUDIO 2.5

Las capacidades de diseño que contiene, ayudan a crear un diseño lógico que puede transformarse en cualquier número de diseños físicos. Como resultado, se puede mantener un diseño lógico normalizado mientras se desnormalizan los diseños físicos para su desempeño. ER/Studio mantiene ligas entre todos los niveles de su diseño por lo tanto puede mezclar cambios en cualquier dirección

entre ellos. ER/Studio revisa la normalización y la compilación con la sintaxis de la plataforma de la base de datos. Se pueden desplegar los modelos de datos usando la notación IDEF1X o IE. ER/Studio permite tomar por omisión las opciones para todos los diagramas así como realizar cambios al momento de la ejecución.

SYSTEM ARCHITECT 4.0

System Architect proporciona todos los elementos para diseñar un nuevo sistema o modificar un sistema actual. Es posible crear modelos lógicamente normalizados y modelos de datos físicamente desnormalizados usando el conjunto de herramientas de System Architect. También se puede crear un modelo conceptual de las entidades y especificar su relación con otras. Al avanzar el proyecto, se pueden incluir llaves primarias, atributos, reglas, constraints de integridad referencial, *triggers* personalizados, y cualquier otra información que se elija para mantenerla en el modelo. Si se diseña un nuevo sistema usando un diccionario amplio de datos es posible especificar los requerimientos de los datos antes de comenzar el modelado mientras se está construyendo el modelo, o después de haber completado el diseño lógico.

Si se está modificando un sistema existente es posible usar la ingeniería de reverso de SA para crear un diagrama de modelo de datos físicos para el sistema actual. SA crea automáticamente un DER de un modelo de datos físico. Entonces se puede modificar el DER, creando un modelo lógico normalizado del nuevo sistema. Una vez que se ha completado el diseño lógico, se pueden generar modelos físicos. Si se planea implementar una base de datos desnormalizada, se puede documentar el proceso de desnormalización usando diagramas *Local View* (Vista Local). SA mantiene ligas entre el modelo lógico, las vistas lógicas, y el modelo físico; por lo tanto los cambios al modelo lógico se reflejan automáticamente en el modelo físico. Al final, se tienen dos modelos físicos separados: uno del sistema actual y otro del sistema propuesto.

Para cualquier tipo de proyecto en el que se esté trabajando, SA proporciona flexibilidad para completar el trabajo. Se pueden elegir modos de despliegue en cualquier tiempo durante el proceso de diseño: conceptual, basado en llaves, totalmente atribuido, o despliegue físico. También, una vez que se ha completado el modelo lógico, se pueden ejecutar una serie de reglas revisadas y reportes de normalización para validar la integridad del diseño. SA prueba las Formas Normales: Primera, Segunda, Tercera, y Boyce Codd.

POWERDESIGNER 6.1

Data Architect proporciona capacidades de modelado de datos tradicional, incluyendo diseño de bases de datos, generación, mantenimiento, ingeniería de reversa y documentación para arquitecturas de bases de datos. Permite que los diseñadores de bases de datos creen estructuras de datos flexibles, eficientes y efectivos para usar una ingeniería de aplicación de bases de datos. También proporciona un diseño conceptual de modelo de datos, generación automática de modelo de datos, diseño de normalización física, sistema de manejo de bases de datos múltiples (DBMS) y soporte de herramientas de desarrollo, y elementos de reportes con presentación y calidad. El diseño se realiza en dos niveles:

- Nivel conceptual: entidades, relaciones, dominios, tipos de datos conceptuales, identificadores, y reglas de negocios.
- Nivel físico: tablas, columnas, dominios, llaves primarias, llaves foráneas, llaves alternadas, índices, constranits de integridad referencial declarativa, vistas, parámetros de almacenamiento físico, reglas de negocios, *triggers* y procedimientos almacenados.

1.9.5 CODIGO QUE GENERA EL LENGUAJE

ERWIN 3.0

Erwin combina bases de datos *back-end* y desarrollo de aplicaciones *front-end* en un ambiente unificado. Tiene soporte para multi-clientes, Erwin genera formas de entrada de datos en Visual Basic, DataWindows de Power Builder y PROGESS SmartObjects del mismo modelo de datos, logrando que los desarrolladores incorporen aplicaciones altamente productivas en tres de los ambientes de desarrollo de bases de datos. Erwin extiende el editor estándar Column Property Editor de tal forma que se pueden asignar rápidamente propiedades de columna del lado del cliente, tales como tipo de control por omisión. Despliega formato y reglas de validación de cliente para cada columna y genera formas de entrada de datos en uso y otros componentes de aplicación directamente del mismo modelo Erwin que crea la base de datos *back-en*d.

Para simplificar aún más el desarrollo de aplicaciones en Visual Basic, Logic Works también ofrece DataBOT(tm), un robot de software avanzado que genera dinámicamente todo el código de acceso de

datos SQL requeridos, permitiendo hasta que los programadores novatos creen rápidamente aplicaciones sofisticadas de bases de datos de alto desempeño en los ambientes actuales.

ERSTUDIO 2.5

Genera otros objetos de base de datos: vistas, procedimientos almacenados, defaults, reglas, y tipos de datos de usuario, lo cual ayuda al auto ordenación de tipos de objetos para eliminar errores de dependencia al construir la base de datos. Tiene una opción para generar código fuente o para construir bases de datos. Soporte para crear bases de datos para Servidores SQL; y otra, para incluir código SQL y verificar la creación de objetos. Además de la opción para incluir encabezados de comentarios.

SYSTEM ARCHITECT 4.0

Genera archivos de definición de almacenamiento de trabajo (.WKS) y sección de pantalla (.SCS) para implementaciones de COBOL. SA/PowerBuilder Link permite el intercambio de la información de diseño entre SA y Power Builder, incluyendo DDL y atributos extensos.

POWERDESIGNER 6.1

Mediante el incremento del modelo de la base de datos, AppModeler genera instantáneamente objetos, componentes *data-ware*, y hasta aplicaciones básicas listas para ejecutarse inmediatamente en PowerBuilder, Power++, Visual Basic, Delphi, y Web-based objects. El AppModeler permite a los desarrolladores: diseñar modelos de bases de datos físicas o crearlas instantáneamente a través de la ingeniería de reversa de bases de datos existentes, generar, documentar y mantener bases de datos, generar rápidamente objetos de aplicación y componentes de datos para PowerBuilder 4.0 y 5.0; Visual Basic 3.0, 4.0, y 5.0; Delphi 2.0; Power++; y el Web.

- Generación de objetos PowerBuilder. Soporta todas las ediciones de PowerBuilder 4.0 y 5.0. Genera objetos personalizables de PowerBuilder y componentes basados en modelos de bases de datos físicos y plantillas que se encuentran dentro de las librerías de clases de su elección. Genera objetos ventana y ventana de datos basadas en tablas, vistas y relaciones de llaves primarias-foráneas. Genera y hace ingeniería de reverso a los atributos. Incluye plantillas personalizables para la librería PowerBuilder Foundation Class (PFC).

- Generación de objetos en Visual Basic. Soporta todas las ediciones de Visual Basic 3.0, 4.0, y 5.0. Incluye add-in de Visual Basic para la fácil manipulación de plantillas predeterminadas personalizables. Genera formas basadas en tablas, vistas, y relaciones de llaves primarias-secundarias. Genera proyectos basados en modelos de propiedades. Genera controles tales como menús, listas, etc.

- Generación de objetos Delphi. Soporta todas las ediciones de Delphi 2.0. Incluye add-in de Delphi para una manipulación de plantillas personalizables predefinidas. Genera aplicaciones y objetos (proyectos, formas, y controles) de tablas, columnas y referencias.

1.9.6 INGENIERIA HACIA DELANTE Y DE REVERSA

ERWIN 3.0

Ingeniería hacia adelante

El desarrollo del modelo de Logic Works usa información específica en un modelo de datos Erwin para acelerar la creación y mantenimiento de soporte, migración y documentación de bases de datos relacionales. El proceso de generar una base de datos físicos de un modelo de datos lógico se llama ingeniería hacia delante *(forward-engineerin*g). A la Generación de un modelo lógico a partir de una base de datos física se llama ingeniería de reversa *(reverse-engineerin*g). Es posible llevar a cabo la ingeniería hacia delante en Erwin (que se llama operación de diagrama) mediante la generación automática de un esquema de base de datos física directamente del modelo de datos sin codificar laboriosamente las definiciones de datos necesarias en SQL y los *trigger*s.

Ingeniería de reverso

La capacidad de ingeniería de reversa de Erwin, la cual puede inferir exitosamente las relaciones entre tablas, permite que se capture rápidamente la estructura de una base de datos existente y convertirla en un modelo lógico independiente del DBMS. Se puede usar el modelo de datos para análisis detallado, se define sobre tiempo y lo distribuye como parte de la documentación requerida a través del ciclo de desarrollo. Si se desea migrar la base de datos existente de una plataforma a otra, Erwin puede hacer la ingeniería de reverso de la base de datos existente, crear un modelo de datos,

modificar o agregar nuevos elementos según sea necesario y después construir la base de datos física en cualquier ambiente de servidor de los que soporta.

ERSTUDIO 2.5

Ingeniería hacia adelante

Una vez que se ha diseñado la base de datos, se pude construir o generar código fuente para todo o para parte de los diseños de la base de datos. Propiamente hace la secuencia de la creación de tipos de objetos diferentes para asegurar eficiencia, y construir bases de datos libres de errores.

Actualiza una base de datos del diagrama. ER/Studio permite aplicar cambios de diseño del modelo de datos directamente a la base de datos. Cuando se comparan las diferencias entre los dos, formula una estrategia de alteración inteligente que implementa el diseño de las modificaciones mientras se preserva la tabla con los datos existentes, privilegios de objetos, y dependencias en la base de datos.

Ingeniería de reverso

Cuenta con ingeniería de reverso, cuando necesite iniciar un trabajo de una base de datos existente, ER/Studio puede hacer una ingeniería de reverso al esquema completo para cualquier plataforma de bases de datos. La operación de la ingeniería de reverso extrae eficientemente definiciones de objetos y construye un modelo de datos gráfico.

SYSTEM ARCHITECT 4.0

Ingeniería hacia delante

Común a los anteriores.

Ingeniería de reverso

Genera y hace ingeniería de reverso en los *trigger*s, reglas, defaults, tipos de datos definidos, dispositivos y bases de datos. Elementos de la Ingeniería de reverso de SA.

- Captura datos del sistema de bases de datos más popular: Access, DB2, Informix, Oracle, SQL Server o SYBASE.

- Ingeniería de reverso de un archivo DDL, a través del manejador ODBC, o a través de la interface directa del SQL Server.
- Genera diagramas de entidad relación que incluyen definiciones para llaves, no llaves y relaciones.
- Genera diagramas de modelo de datos físicos que incluyen definiciones para tablas, columnas y constantes.
- Crea llaves primarias, índices alternados, y rutas de acceso de los índices en la estructura de la base de datos.
- Importa *trigger*s, procedimientos almacenados, defaults, y roles de definiciones de bases de datos existentes.

POWERDESIGNER 6.1

Ingeniería hacia delante

Común a los anteriores.

Ingeniería de reverso

Visualiza estructuras de bases de datos existentes directamente vía ODBC o usando archivos de script DDL. Genera el modelo conceptual del modelo físico.

Retro-documentación de bases de datos existentes. Re-orientación de la base de datos existente a un DBMS diferente. Interfaz con herramientas de desarrollo: exporta información del modelo físico y extiende atributos al diccionario de 4GL, importa atributos extendidos de PowerBuilder, soporta definición de atributos extendidos para PowerBuilder, Progress, Uniface, PowerHouse, Axiant, y NS-DK.

1.9.7 SINCRONIZACION DE LA BASE DE DATOS

ERWIN 3.0

El manejo de cambio comprensivo de Erwin y los elementos de generación de script de alteración facilita el almacenamiento del modelo de datos y la base de datos física de manera sincronizada. Durante la sincronización, Erwin desempeña una comparación comprensiva de todos los objetos físicos y lógicos, incluyendo definiciones de tablas y columnas, llaves, índices y parámetros de almacenamiento físico, resaltando cualquier discrepancia y permitiendo que se migren las definiciones de la base de datos modelo al modelo de la base de datos.

Si se exportan cambios de un modelo a una base de datos, Erwin genera *scripts* de alteración de SQL para actualizar el esquema de la base de datos. Erwin automáticamente revisa todas las dependencias del esquema y traduce y preserva los datos existentes cuando se actualiza el esquema, facilitando los cambios de diseño migrados para activar el desarrollo, probar y producir bases de datos. Además de la sincronización del modelo de toda la base, Erwin soporta sincronización de modelo a modelo y modelo a *scrip*t.

ERSTUDIO 2.5

Sincronización entre el diagrama físico y el lógico. Mezcla entre cualquier par de diagramas físicos para la misma plataforma de bases de datos. Comparación lado-a-lado de las diferencias. El usuario puede decidir que diferencias mezclar o ignorar. Objetos reusables. Construir atributos reusables. Aplicarlos a atributos y columnas. Propagación global de actualizaciones. Construir tipos de datos personalizables. Submodelado. Crear cualquier número de subvistas personalizables sobre un diagrama físico o lógico. Cualquier objeto puede existir en cualquier número de subvistas (relaciones de muchos a muchos entre objetos y subvistas). Crear rápidamente subvistas eligiendo un área del diagrama. Control independiente sobre el despliegue de la subvista, incluyendo posición del objeto, colores y letras. Utilidad de búsqueda rápida. Editores en tabla para evitar la necesidad de poner en modo cascada los diálogos. Diferenciación de color de llaves primarias y secundarias inherentes. Sombreado de cajas de entidad.

SYSTEM ARCHITECT 4.0

El Generador de esquemas (Schema Generator) automatiza la creación y mantenimiento de esquemas para DBMS de SQL y 4GL SA Schema Generator traduce modelos de datos (ER o diagramas físicos DM), IDEF1X y diagramas de clases orientadas a objetos en las definiciones de los esquemas para manejo de sistemas con múltiples bases de datos. La transferencia de esquemas en la DBMS se puede llevar a cabo en vivo, vía conexión ODBC, o a través del uso de un archivo. DDL intermedio. SA Schema Generator simplifica el desarrollo y asegura la integridad de los diseños permitiendo crear y mantener esquemas de bases de datos para una amplia variedad de administradores de sistemas de bases de datos SQL y no SQL. Cuando se genera un DBMS múltiple, los tipos de datos definidos en el modelo de datos se mapean automáticamente a los tipos correctos del DBMS seleccionado. Se usa el SA Schema Generator para generar:

- DDL para la creación de tablas y mantenimiento desde el repositorio de SA.Definir Create Table o Alter Table, llaves primarias, y llaves secundarias.
- Crear definiciones para reglas, procedimientos almacenados, defaults, y mensajes.
- Triggers de integridad referencial y constraints.
- Usar tipos de datos definidos por el usuario.
- Definiciones de Disk Init, Disk Mirror, Add Segment, y Create Database.
- Índices de rutas de acceso y llaves.
- Definiciones de tipos para C y libros de COBOL.
- El generador de esquema (Schema Generator) prueba las palabras reservadas de COBOL y automáticamente modifica con extensiones -XX.

POWERDESIGNER 6.1

Soporta más de 30 DBMS, incluyendo Sybase SQL Server, Oracle, Informix, Ingress, Sybase SQL Anywhere, Microsoft SQL Server, SQLBase, Progress, Access, Paradox, FoxPro, etc. Creación directa de bases de datos vía ODBC o usando archivos de scripts DDL. Elige la generación del modelo entero, ciertos submodelos, u objetos individuales. Reglas de negocios definidas de usuario pueden integrarse en triggers y procedimientos almacenados. Para el mantenimiento de la base de datos, incluye:

- Modelos de archivo.

- Comandos alterados generados para preservar datos contenidos.
- Mantenimiento vía ODBC o a través de archivos de scripts DLL.
- Elegir para generar el modelo entero, ciertos submodelos, u objetos individuales.

1.9.8 DISENO

ERWIN 3.0

Complete-Compare es una tecnología de Logic Works que cambia la forma de modelar la interacción de modelos o bases de datos. Provee de una comparación comprensiva de todas las diferencias. Resaltando cualquier discrepancia, los cambios pueden migrarse de un modelo-a- base de datos o de una base de datos-a-un modelo. Erwin también integra la base de datos en el desarrollo del proceso de la aplicación. Cuando se ligan las herramientas de desarrollo, Erwin sincroniza el *back-end* de la base de datos con las formas del *front-en*d.

ERSTUDIO 2.5

ER/Studio ayuda a prolongar la inversión que se ha hecho. Soporta el proceso de diseño iteractivo inherente en el ciclo de vida de la aplicación.

SYSTEM ARCHITECT 4.0

System Architect cuenta con la herramienta Screen Painter, con la que se pueden diseñar pantallas y menús para aplicaciones de Windows, y pantallas de caracteres para aplicaciones de COBOL. Los archivos de Windows .DLG y .MNU se pueden generar automáticamente por SA, incluyendo posición, leyenda, *hot ke*y, orden de tablas, y número de identificación para cada control o elemento de menú incluido en la aplicación original de Windows. Por otra parte, pueden crearse pantallas usando controles estándar de Windows, incluyendo botones de presión, cajas de revisión, botones de opciones, cajas de combo, cajas de listas, cajas de texto. Se pueden generar archivos de diálogos (.DLG), encabezado (.H), y forma de Visual Basic (.FRM) para aplicaciones de Windows. Usa una rutina simple de captura para crear un nuevo menú de cualquier menú de aplicación de Windows. Dibujar menús usando los elementos del menú, submenú y separadores. Agregar accesos directos

para elementos del menú y submenú. Los elementos del submenú y del menú se activan usando el cursor. Los diálogos seleccionados se abren de elementos seleccionados del menú.

POWERDESIGNER 6.1

MetaWorks es un sistema diseñado para proveer los módulos gráficos de PowerDesign con la habilidad de compartir y almacenar modelos de datos en un solo punto de control, el Diccionario MetaWorks se ejecuta en una PC y almacena los modelos de datos en un servidor de bases de datos, que puede ser Sybase, SQL Anywhere o cualquier otro como Oracle, Informix, DB2, MS SQL Server y CA OpenIngres. MetaWorks provee de tres funciones principales: *Data Model y Submodel Extraction/Consolidation, Project (or Dictionary) Managemen*t, y *Environment Administration.* El MetaBrowser presenta vista de árbol en una línea jerárquica de la aplicación bajo el estudio (base de datos, proyecto, modelo, objeto, y submodelo), expande o colapsa vista de objetos, crea, modifica, borra o imprime objetos seleccionados, habilita comparación entre modelos del mismo tipo, en el nivel de objeto, trabaja con listas de objetos a través de cualquier proyecto, modelo o submodelo.

1.10 ERECASE, HERRAMIENTA CREADA EN CUBA. CARACTERÍSTICAS

ERECASE v.2.0, está concebida para brindar al diseñador un amplio conjunto de construcciones del modelo ERE que le permitan construir, desde cero, el diagrama de una base de datos relacional. Esta herramienta no solo dibuja el esquema conceptual, además del editor gráfico contiene otras funciones que lo hacen útil en el diseño de la base de datos, estas funciones se mencionan a continuación:

- Validación estructural del modelo ERE.
- Traducción del esquema conceptual al modelo lógico relacional, generando los esquemas adecuados.
- Generación de un *script* SQL para la creación de la base de datos, en cualquier gestor de bases de datos que soporte el lenguaje SQL.

1.11 CARACTERÍSTICAS DEL DB DESINGER 4.

DBDesigner 4 es un sistema totalmente visual de diseño de bases de datos, que combina características y funciones profesionales con un diseño simple, muy claro y fácil de usar, a fin de ofrecerte un método efectivo para gestionar tus bases de datos. Te permite administrar la base de datos, diseñar tablas, hacer peticiones SQL manuales y mucho más, como ingeniería inversa en

MySQL, Oracle, MSSQL y otras bases de datos ODBC, modelos XML y soporte para la función drag-and-drop. El programa dispone además de una interfaz profesional y de detallados manuales de uso.

El listado de características a destacar del DBDesigner 4.

- Versiones para GNU/Linux y Microsoft Windows
- Modo diseño / Modo consulta
- Ingeniería inversa para MySQL, Oracle, MSSQL y cualquier base de datos ODBC
- Sincronización *modelo visual / base de datos*
- Colocación automática de clave ajena
- Soporte de entidad débil
- Opciones avanzadas a la hora de imprimir el modelo
- Posibilidad de exportar el modelo como imagen
- Soporte integro con MySQL (tipos de datos)
- Tipos de datos de usuario definidos
- Posibilidad de guardar el modelo independientemente de su base de datos correspondiente.
- Acceso multiusuario y en red
- Control de versiones
- Manejador SQL
- Manejador de plugins

1.12 HERRAMIENTAS QUE SE USARÁN PARA EL DESARROLLO.

Para desarrollar una herramienta se debe realizar un estudio de la plataforma y lenguaje en que será implementará así como de las particularidades que nos ofrece el servidor de bases de datos PostgreSQL que es el objetivo a realizar por esta herramienta, que genere y modele bases de datos a dicho gestor.

- Plataforma de desarrollo Java.
- Lenguaje Java.
- XML como lenguaje en que se exporta en forma de archivo la base de datos.
- PostgreSQL como sistema gestor de bases donde se harán las primeras pruebas.

1.12.1 PLATAFORMA DE DESARROLLO JAVA.

Java es una plataforma virtual que provee:

- El lenguaje Java: un lenguaje de programación.
- La *Java Virtual Machine*: Una máquina virtual con su propio set de instrucciones.
- La *Java Platform API*: Una interfaz para la programación de aplicaciones.

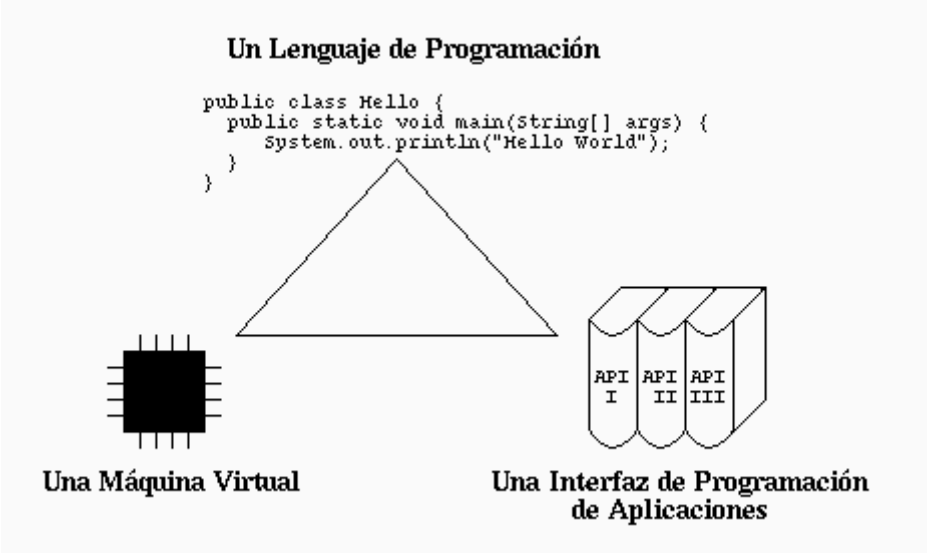


Fig. 2 Plataforma Virtual de Java.

El lenguaje java orientado a objetos:

- Clases
- Objetos
- Métodos
- Subclases
- Herencia simple
- Enlace dinámico
- Encapsulamiento

Diseño y Programación Orientada Objetos:

- Diseño bibliotecas de clases: Se necesita experiencia y habilidad.
- Usar bibliotecas de clases: Mucho más fácil que usar bibliotecas de procedimientos.
- Las clases son un mecanismo de abstracción eficaz.

La API de la Plataforma Java Consiste en una biblioteca de clases que provee de:

- Manejo de archivos
- Interfaces gráficas
- Acceso a la red internet
- Acceso a bases de datos
- Y mucho más.

Multiplataforma: La plataforma Java se incrusta en plataformas reales como:

- Windows'95 y 'NT (pero no 3.1)
- Power/Mac
- Unix (Solaris, Linux, ...)

Implementación de la Plataforma Java: La implementación de la plataforma Java provee:

- Un compilador de Java para traducir código fuente al set de instrucciones de la máquina virtual de Java.
- Un intérprete o un compilador JIT para ejecutar las instrucciones de la máquina virtual.
- Una implementación de la API de Java.

Java es robusto:

- Java siempre chequea los índices.
- Java siempre chequea los *casts*.
- Java hace recolección automática de basuras.
- Java no tiene aritmética de punteros.

Liberación explícita de la memoria:

- 40% del tiempo de desarrollo se dedica a colocar los delete.

- Proporción aumenta a medida que la complejidad aumenta: para mil líneas es fácil, para diez mil es difícil, para cien mil líneas es imposible.
- La mantención de los programas es aún más difícil.

Conclusión:

En Java los programadores son más productivos porque:

- Se necesita programar menos pues hay abundantes bibliotecas de clases.
- No se necesita reescribir los programas cuando se cambia la plataforma.
- No se pierde tiempo liberando explícitamente la memoria.
- Se pierde menos tiempo depurando los programas.
- Java es compilado.

Sin embargo:

- Los programas en Java se ejecutan más lento que sus equivalentes en C o C++.
- La plataforma estándar ocupa como mínimo 16 MB en memoria principal y unos 30 MB en disco.
- Corre bien en Pentium 166MHz con 32 MB.
- La programación de applets es frágil.

1.12.2 EL LENGUAJE JAVA.

Es un lenguaje orientado a objeto, de una plataforma independiente, fue desarrollado por la compañía Sun Microsystems, con la idea original de usarlo para la creación de páginas WEB. Esta programación Java tiene muchas similitudes con el lenguaje C y C++, así que si se tiene conocimiento de este lenguaje, el aprendizaje de la programación Java será de fácil comprensión por un programador que haya realizado programas en estos lenguajes. Con la programación en Java, se pueden realizar distintos aplicativos, como son applets, que son aplicaciones especiales, que se ejecutan dentro de un navegador al ser cargada una pagina HTML en un servidor WEB, Por lo general los applets son programas pequeños y de propósitos específicos. Otra de las utilidades de la programación en Java es el desarrollo de aplicaciones, que son programas que se ejecutan en forma independiente, es decir con la programación Java, se pueden realizar aplicaciones como un procesador de palabras, una hoja que sirva para cálculos, una aplicación grafica, etc. en resumen cualquier tipo de aplicación se puede

realizar con ella. Java permite la modularidad por lo que se pueden hacer rutinas individuales que sean usadas por más de una aplicación, por ejemplo tenemos una rutina de impresión que puede servir para el procesador de palabras, como para la hoja de calculo. La programación en Java, permite el desarrollo de aplicaciones bajo el esquema de Cliente Servidor, como de aplicaciones distribuidas, lo que lo hace capaz de conectar dos o más computadoras u ordenadores, ejecutando tareas simultáneamente, y de esta forma logra distribuir el trabajo a realizar.

1.12.2.1 CARACTERÍSTICAS DEL LENGUAJE JAVA.

Java es un lenguaje orientado a objetos, eso implica que su concepción es muy próxima a la forma de pensar humana. También posee otras características muy importantes:

- Es un lenguaje que es compilado, generando ficheros de clases compilados, pero estas clases compiladas, son en realidad interpretadas por la máquina virtual de java. Siendo la máquina virtual de java la que mantiene el control sobre las clases que se estén ejecutando.
- Es un lenguaje multiplataforma: El mismo código java que funciona en un sistema operativo, funcionará en cualquier otro sistema operativo que tenga instalada la máquina virtual java.
- Es un lenguaje seguro: La máquina virtual, al ejecutar el código java, realiza comprobaciones de seguridad, además el propio lenguaje carece de características inseguras, como por ejemplo los punteros.
- Gracias al API de java podemos ampliar el lenguaje para que sea capaz de, por ejemplo, comunicarse con equipos mediante red, acceder a bases de datos, crear páginas HTML dinámicas, crear aplicaciones visuales al estilo Windows.

Para poder trabajar con java es necesario emplear un software que permita desarrollar en java. Existen varias alternativas comerciales en el mercado: JBuilder, Visual Age, Visual Café, NetBeans y un conjunto de herramientas shareware, e incluso freeware, que permiten trabajar con java. Pero todas estas herramientas en realidad se basan en el uso de una herramienta proporcionada por Sun, el creador de java, que es el Java Development Kit (JDK).

1.12.3 ¿QUE ES XML?

Lenguaje de marcado extensible (XML) puede agregar nuevas palabras para que se adecuen a propósitos específicos y marcado porque incluye símbolos especiales en un documento para cumplir alguna función específica, por lo que proporciona un método para describir datos estructurados. XML es un subconjunto de SGML que es el estándar internacional para la definición de la estructura y el contenido de diferentes tipos de documentos, optimizado para la entrega a través de la web. El Consorcio World Wide Web define los estándares de XML para que los datos estructurados sean uniformes e independientes de las aplicaciones. (10)

XML fue pensado en un principio para utilizarse en aplicaciones Web, sin embargo su utilidad se ha extendido y ahora se emplea también en aplicaciones estándares de la plataforma .NET y para el intercambio de datos entre sistemas desiguales, ya que permite que esto ocurra sin tener en cuenta la plataforma ni el lenguaje de desarrollo. Es una tecnología sencilla. En la actualidad es muy importante la compatibilidad entre sistemas para compartir la información de una manera segura fiable y fácil.

1.12.3.1 ¿COMO FUNCIONA? (VIABILIDAD EN ALGUNOS PROYECTOS)

Cualquier sistema que necesite un control de información y su transformación a otros lenguajes podría gestionarla por medio de XML. Estamos hablando de un metalenguaje que propone una forma estándar de organizar nuestra información, flexible (permite crear nuestras propias etiquetas), independiente de plataforma, sistema operativo y lenguaje de programación, y además pensado para que los navegadores puedan trabajar con él. Es decir, estamos hablando de una gran base de datos virtual a la que cualquier aplicación, independientemente del entorno, sistema operativo y lenguaje de programación puede acceder siguiendo unas API's estándar. En sistemas de documentación. Permite la estructuración y organización inteligente de la información. Permite la manipulación inteligente de documentos o parte de estos. Permite desde una única fuente de datos diferentes formatos de presentación y salida. En el comercio electrónico. Permite el intercambio de información entre sistemas Heterogéneos, lo que facilita el intercambio de información entre empresas.

1.13 CONCLUSIONES

La Fundamentación y estudio de todos los aspectos teóricos que se citan en este capitulo dan muestra de la importancia de utilizar la metodología RUP y el lenguaje de Modelado Unificado (UML) así como las herramientas y tecnologías a utilizar para analizar, diseñar y posteriormente construir una herramienta para modelado de datos. De las herramientas comparadas y caracterizadas se concluye que el ERWIN es la que ofrece muchas ventajas de acuerdo a las características que presenta, ERWIN es la herramienta más sencilla, rápida y ligera para correr, pues se basa en un componente que sólo cubre la base de datos. De acuerdo a las características y muy buenas funcionalidades que presenta el ERWIN prácticamente se realiza el análisis y diseño de la herramienta que se pretende desarrollar.

CAPÍTULO 2: CARACTERISTICAS DEL SISTEMA

2.1 INTRODUCCION

En este capítulo se comienza a tener la concepción práctica sobre lo que se pretende que haga la herramienta, sobre la base de las dificultades, necesidades y características que para la UCI la herramienta a diseñar debe cumplir. Para una mejor comprensión del sistema se creó lo que se conoce cómo modelo de dominio, también se listan los requisitos tanto funcionales cómo no funcionales obtenidos en la fase de inicio así como la descripción de los posibles casos de uso del sistema.

2.2 ¿QUE ES UN MODELO DE DOMINIO?

El Modelo de Dominio (o Modelo Conceptual) es una representación visual de los conceptos u objetos del mundo real significativos para un problema o área de interés. Representa clases conceptuales del dominio del problema. Representa conceptos del mundo real, no de los componentes de software. Una clase conceptual puede ser una idea o un objeto físico (símbolo, definición y extensión). El modelo de dominio se representa en UML con un Diagrama de Clases en los que se muestra:

- conceptos u objetos del dominio del problema: clases conceptuales
- asociaciones entre las clases conceptuales
- atributos de la clase conceptuales

En el Modelo de Dominio no se muestra comportamiento. Las clases conceptuales pueden tener atributos pero no métodos. Cualquiera sea la solución de casos de uso que se haya elegido, los conceptos e ideas propias del dominio del problema son las mismas; un mismo modelo de dominio contempla cualquiera de las soluciones analizadas. El modelo de dominio es global, es decir se realiza para todos los casos de uso y no para uno en particular.

Características:

- No hay cronología
- No se diferencia entre dentro y fuera del sistema
- Es global, no por caso de uso
- No es completo: esquemático, las asociaciones están resumidas.

Guía para la construcción del modelo de dominio

- Listar conceptos
- Representar los conceptos en un diagrama
- Agregar las asociaciones para registrar las relaciones entre conceptos
- Agregar los atributos necesarios para cumplir los requerimientos de información.

2.3 ¿POR QUÉ MODELO DE DOMINIO?

Teniendo en cuenta que la definición de procesos y roles del negocio se hace difícil encontrarlos, por lo que se ve a simple vista la necesidad de describir el funcionamiento de la aplicación mediante una serie de conceptos, entidades y sus relaciones, agrupándose en un modelo de dominio con el fin del fácil entendimiento de la aplicación.

2.4 CONCEPTOS ASOCIADOS AL DOMINIO

Un modelo de dominio captura los tipos más importantes de objetos en el contexto del sistema. Los objetos de dominio representan los principales conceptos. Muchos de los objetos y clases pueden obtenerse de una especificación de requisitos. La modelación del dominio tiene como objetivo fundamental la comprensión y descripción de las clases más importantes en el sistema.

Concepto	Descripción
Usuario	Representa a personas que interactúan con la herramienta.
Entidad	Cualquier tipo de objeto o concepto sobre el que se recoge información: cosa, persona, concepto abstracto o suceso. Las entidades se representan gráficamente mediante rectángulos y su nombre aparece en el interior. Un nombre de entidad sólo puede aparecer una vez en el esquema conceptual. Hay dos tipos de entidades: fuertes y débiles. Una *entidad débil* es una entidad cuya existencia depende de la existencia de otra entidad. Una *entidad fuerte* es una entidad que no es débil.
Relación	Es una correspondencia o asociación entre dos o más entidades. Cada relación tiene un nombre que describe su función.
Atributo	Es una característica de interés o un hecho sobre una entidad o sobre una relación. Los atributos representan las propiedades básicas de las entidades y de las relaciones. Toda la información extensiva es portada por los atributos.

	Gráficamente, se representan mediante bolitas que cuelgan de las entidades o relaciones a las que pertenecen.
Error de validación	Error encontrado en el diseño del modelo.
Reporte de validación	Valida el modelo reportando o no posibles errores en el diseño del modelo de datos.
Modelo	Representación gráfica de la realidad que son clarificados a través de texto explicativo.
Salva de modelo	Almacenará el modelo de datos en un fichero.
Script de base de datos	Contendrá en modelo físico de la base de datos.

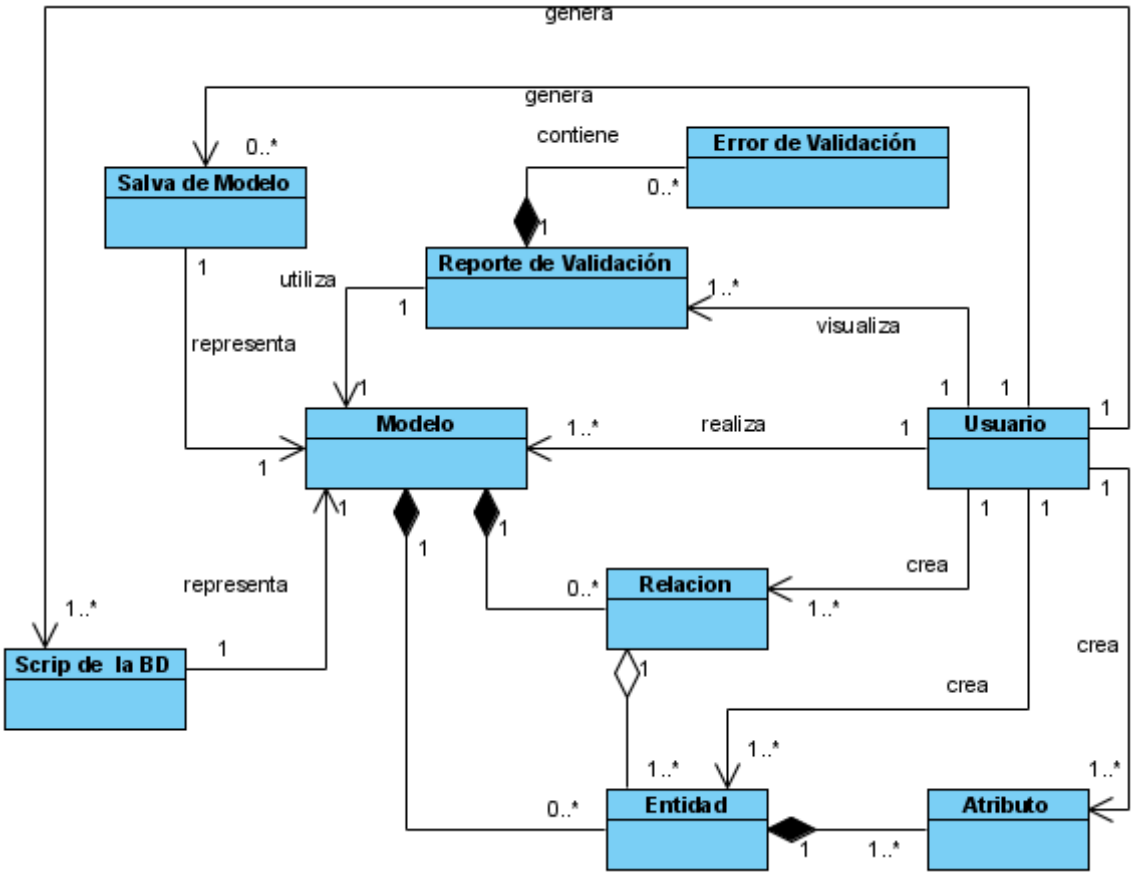


Fig. 3 Diagrama del Modelo de Dominio

2.5 DESCRIPCIÓN DEL MODELO DE DOMINIO

Cada modelo realizado por el usuario contendrá las entidades, relaciones y atributos. Cada reporte usa el modelo asociado para comprobar posibles errores en la validación. El usuario puede o no general salvas, donde estas salvas están representadas por el modelo. Por su parte cada modelo es representado por un script de base de datos donde también puede ser generado una o varias veces por cada usuario.

2.6 REQUERIMIENTOS.

El propósito general de la captura de requisitos es obtener una descripción correcta de lo que debe de hacer el sistema y delimitar su alcance, es decir, que debe y que no debe hacer. Los requisitos juegan un papel importante durante el ciclo de vida de un software. Durante la fase de inicio se identifican la mayoría de los casos de uso para delimitar el sistema y detallar los más importantes. En la fase de elaboración se obtiene el resto de los casos de uso y se planean las tareas que se tienen que llevar a cabo para la elaboración de la herramienta. Si se detectaran algunos otros requisitos, se capturan e implementan en la fase de construcción. Finalmente en la fase de transición, solo hay captura de requisitos si se tuviera que modificar algún caso de uso.
El artefacto obtenido en este flujo de trabajo es el modelo de casos de uso, que incluyen los casos de uso y los actores.

2.6.1 REQUISITOS FUNCIONALES.

R1-Crear Nuevo Modelo
R2-Abrir Modelo
R3-Salvar Modelo
R4-Exportar Modelo a XML
R5- Cerrar Modelo
R6-CRUD Entidad
    R6.1-Crear Entidad
    R6.2-Modificar Entidad
    R6.3-Eliminar Entidad
R7-CRUD Atributo
    R7.1-Crear Atributo

R7.2-Modificar Atributo

R7.3-Eliminar Atributo

R8-CRUD Relación

R8.1- Crear Relación

R8.2- Modificar Relación

R8.3-Eliminar Relación

R9-Validar Modelo

R10-Convertir a Modelo Físico

2.6.2 REQUISITOS NO FUNCIONALES.

- **Interfaz Externa:** Diseño sencillo y amigable que permita a los usuarios no familiarizados un uso extensivo de los componentes de la herramienta.
- **Usabilidad:** Estará dirigido a usuarios con conocimientos básicos en modelado de bases de datos, deberá ser usada fácilmente por los mismos.
- **Rendimiento:** Debe tener cierto grado de velocidad de procesamiento, tiempo de recuperación y de respuesta.
- **Portabilidad:** La herramienta estará diseñada para las distintas versiones de cualquier sistema operativo.
- **Software:** Como sistema operativo las distintas versiones de Linux y Windows y como gestor de bases de datos PostgreSQL.

2.7 MODELO DE CASOS DE USO DEL SISTEMA.

El modelo de casos de uso es un modelo del sistema que contiene actores, casos de uso y sus relaciones. Los actores representan los usuarios del sistema y otras aplicaciones que interactúan con él, es decir, representan terceros fuera del sistema que se relacionan con él. Estos suelen corresponderse con trabajadores o actores del negocio. El único actor definido en el sistema sería:

COMO ACTOR DEL SISTEMA TENDRIAMOS A:

Actores del sistema	Justificación
Usuarios	Diseña el modelo de datos en el ambiente que brinda la herramienta.

CASOS DE USO DEL SISTEMA:

- Crear Nuevo Modelo
- Abrir Modelo
- Salvar Modelo
- Exportar Modelo a XML
- Cerrar Modelo
- CRUD Entidad
- CRUD Atributo
- CRUD Relación
- Validar modelo
- Generar Modelo Físico

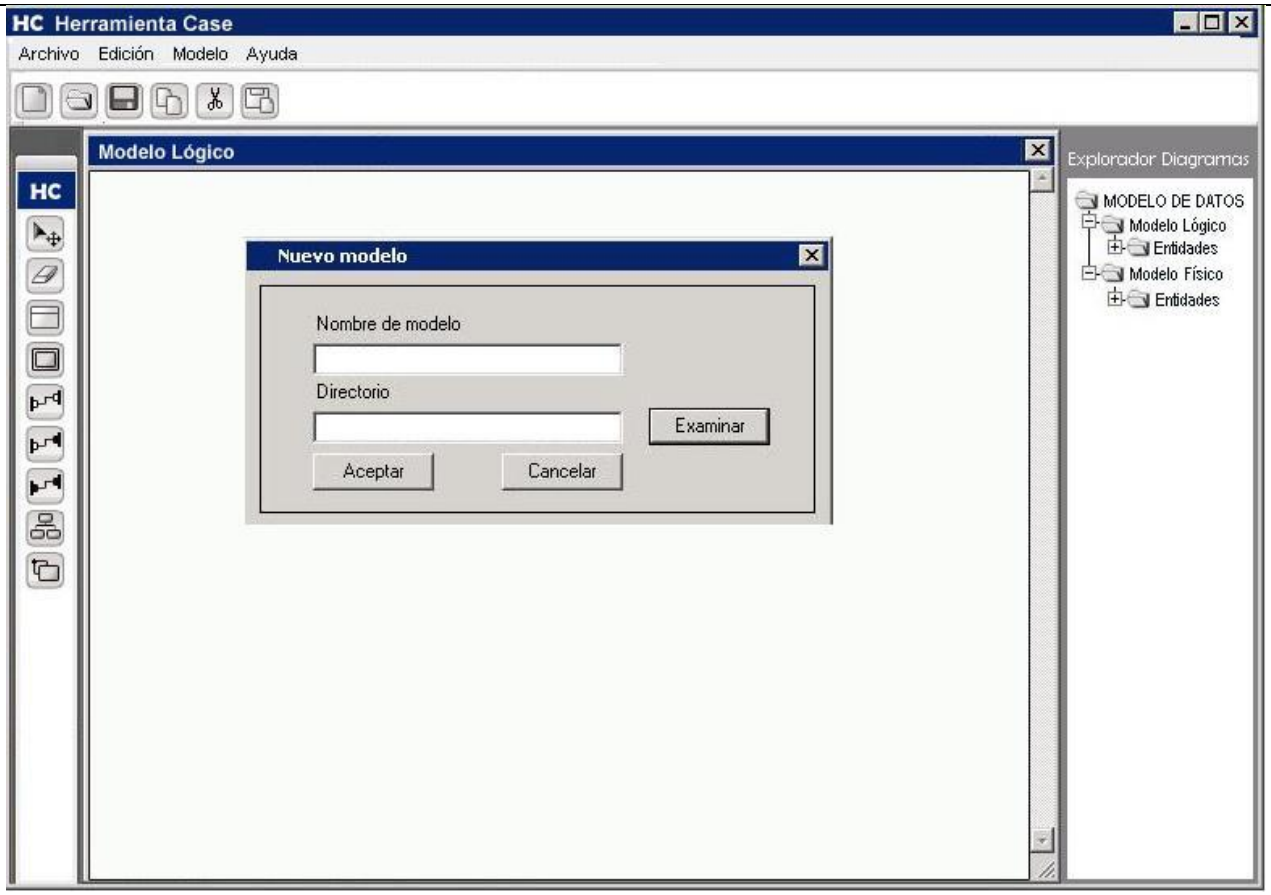
Fig. 4 Diagrama de casos de uso del sistema.

2.8 ESPECIFICACION DE LOS CASOS DE USO EN FORMATO EXPANDIDO.

CASO DE USO CREAR NUEVO MODELO

Caso de Uso:	**CREAR NUEVO MODELO**
Actores:	Usuario
Resumen:	El caso de uso se inicia cuando el usuario desea editar un nuevo modelo de datos y termina cuando el sistema le muestra el área de trabajo del nuevo modelo de datos creado.
Precondiciones:	No existe un modelo de datos.
Referencias	**R1**
Prioridad	Primario

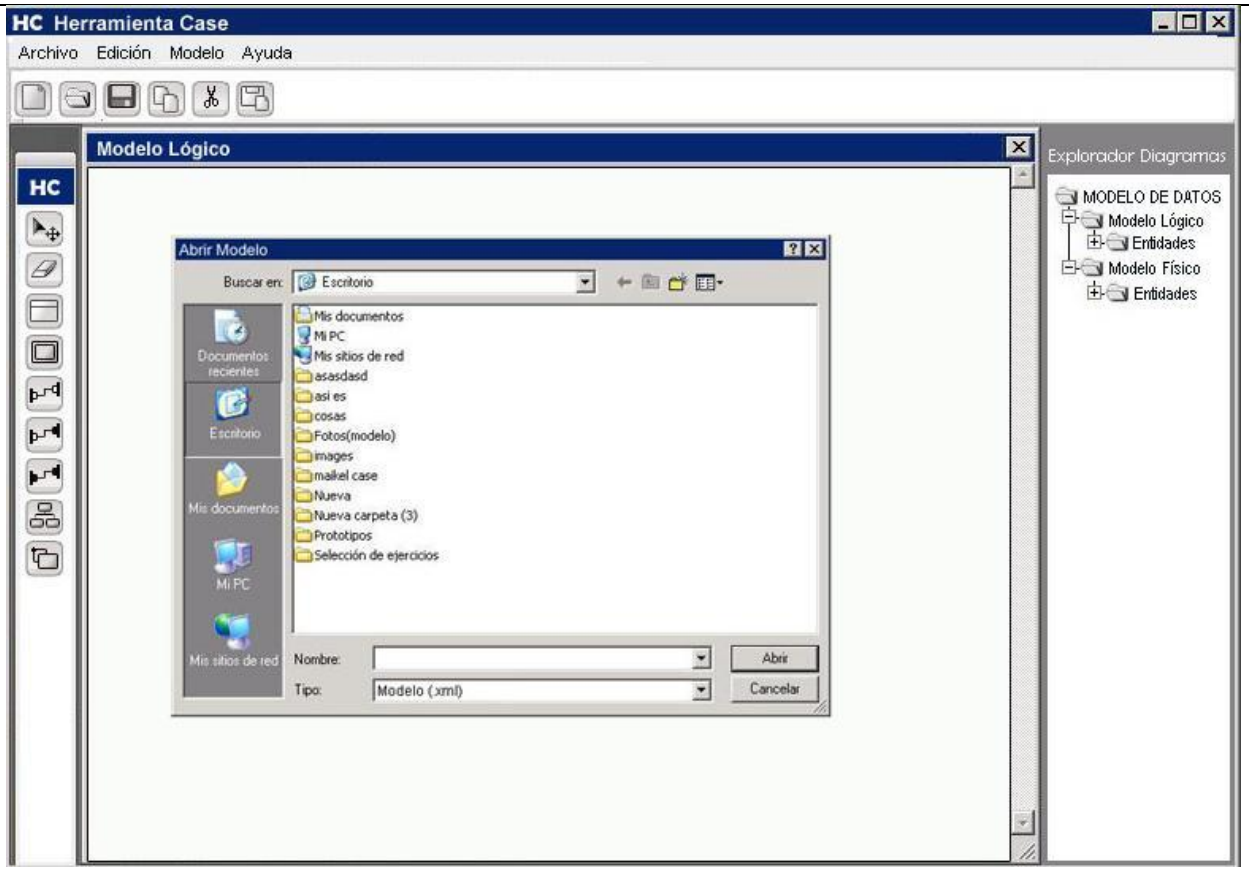
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1- Selecciona la opción “Crear Nuevo Modelo”.	2- Se muestra un formulario para entrar el nombre del Modelo y el directorio de ubicación.
3- Introduce los datos.	4- Si los datos son correctos se crea el modelo y se abre un área de edición para comenzar a editarlo.



Flujos Alterno 1	
Acción del Actor	**Respuesta del Sistema**
3-Cancela la creación del nuevo modelo.	4- Cierra el formulario.
Flujos Alterno 2	
Acción del Actor	**Respuesta del Sistema**
	4-Si los datos no son correctos se muestra un error especificando el tipo de error y retorna al paso 2.
Poscondiciones	El Modelo queda creado.

CASO DE USO ABRIR MODELO

Caso de Uso:	**ABRIR MODELO**
Actores:	Usuario
Resumen:	El caso de uso se inicia cuando el usuario desea abrir un modelo que esté creado en cualquier directorio del ordenador y termina cuando el sistema muestra el modelo seleccionado.
Precondiciones:	Debe de existir al menos un modelo creado.
Referencias	**R2**
Prioridad	Primario
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1- Selecciona la opción “Abrir Modelo”.	2- Muestra un formulario para buscar en el directorio de ubicación el modelo que se desea abrir.
3- Selecciona el archivo que desea abrir.	4- Muestra el modelo seleccionado.



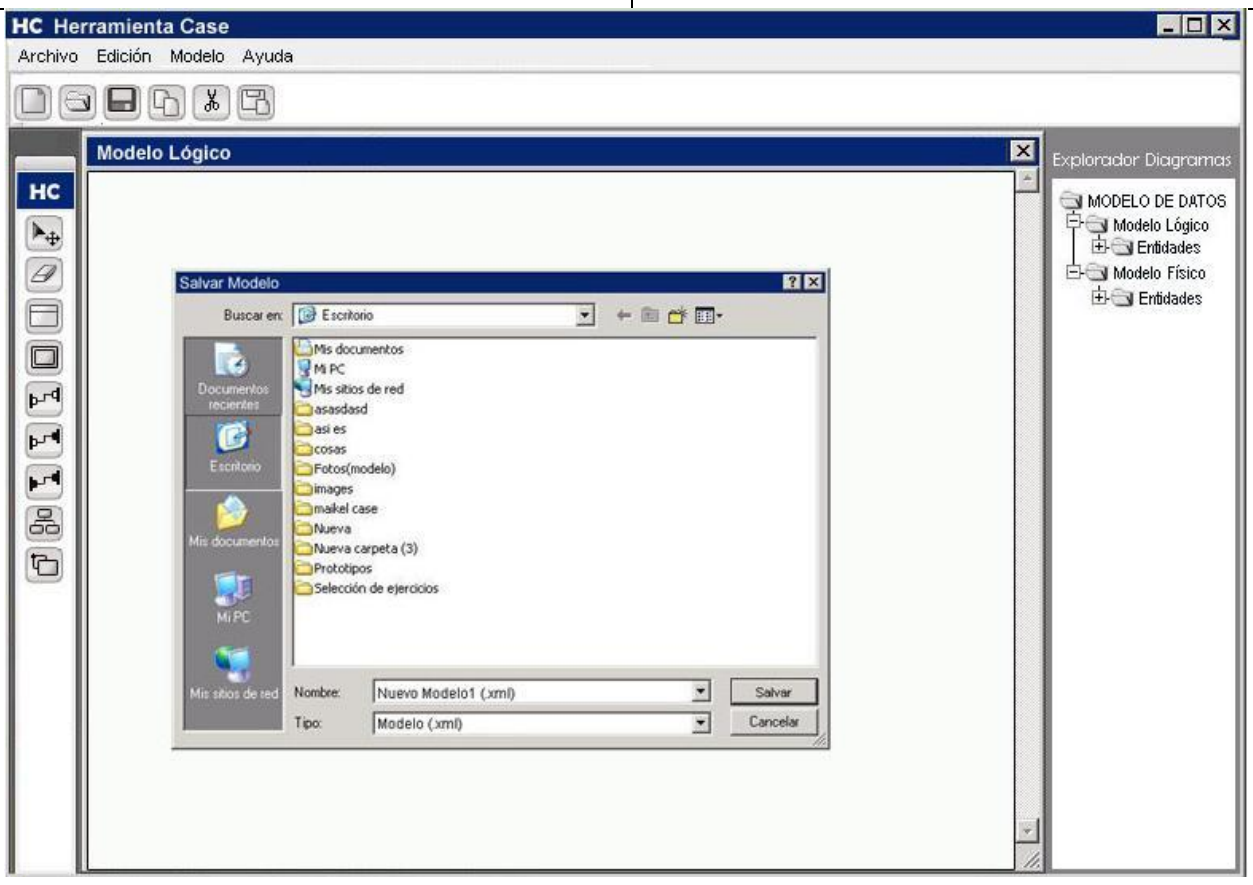
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
	4-Si el modelo seleccionado no es de la extensión de la aplicación el sistema muestra un mensaje y retorna al paso 2.
Poscondiciones	El Modelo que mostrado en la aplicación.

CASO DE USO SALVAR MODELO

Caso de Uso:	**SALVAR MODELO**
Actores:	Usuario
Resumen:	El caso de uso se inicia cuando el usuario desea salvar un modelo que se esté editando y termina cuando el sistema salva el modelo en el directorio

	de ubicación seleccionado.
Precondiciones:	Tiene que estar al menos un modelo editándose.
Referencias	**R3**
Prioridad	Primario

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1-Selecciona la opción “Salvar Modelo”.	2- Muestra un formulario para salvar el modelo en el directorio seleccionado.
3- Introduce el nombre del modelo y acepta salvar.	4- Salva el modelo y cierra la ventana de dialogo.



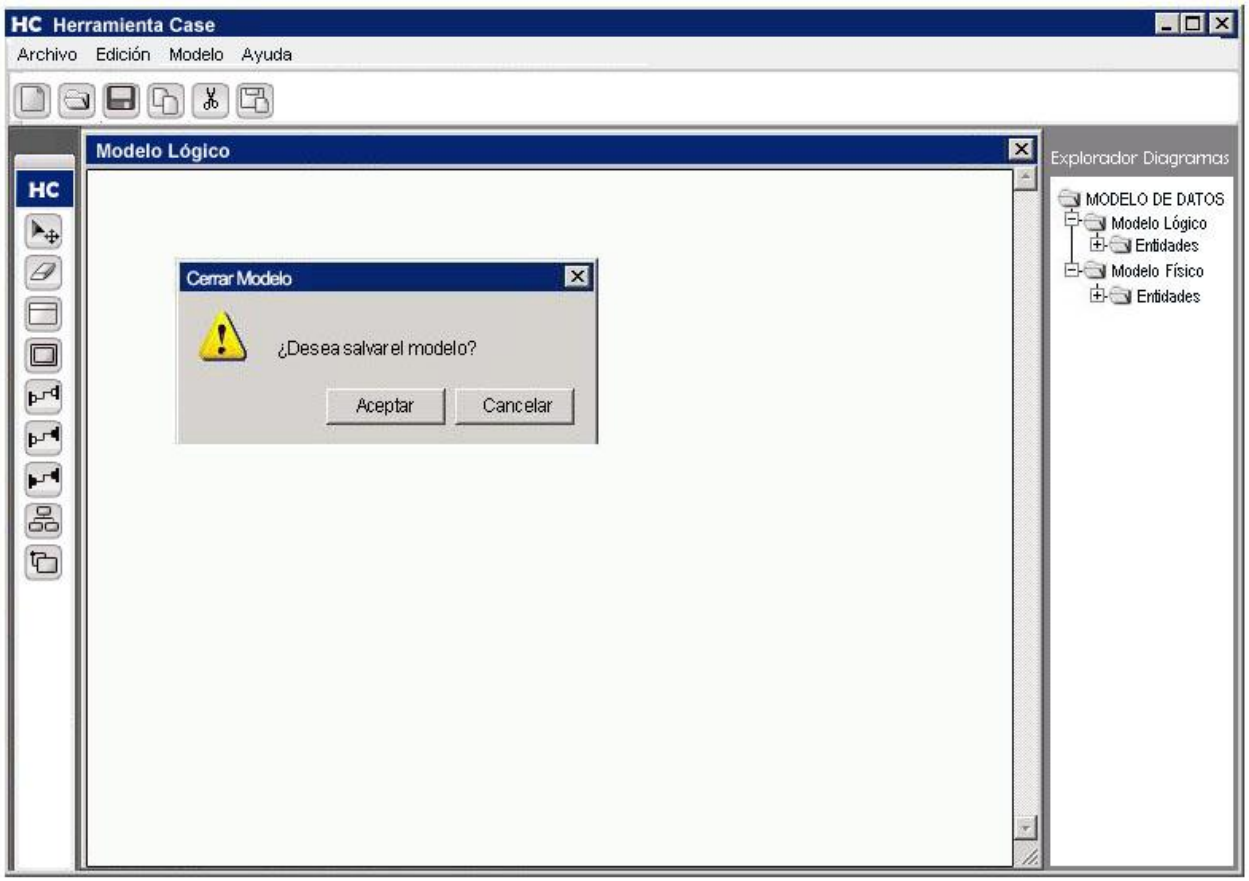
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
4- Cancela salvar modelo.	5-Cierra la ventana de dialogo mostrada y no salva el modelo.

Poscondiciones	Se crea un fichero que almacena todos los datos necesarios para representar un modelo.

CASO DE USO CERRAR MODELO

Caso de Uso:	**CERRAR MODELO**
Actores:	Usuario
Resumen:	El caso de uso se inicia cuando el usuario desea cerrar un modelo que se este editando y termina cuando el modelo queda cerrado.
Precondiciones:	Tiene que haber al menos un modelo que se este editando.
Referencias	**R5**
Prioridad	Primario

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1- Selecciona la opción “Cerrar modelo”.	2- Muestra un formulario con las opciones de aceptar o cancelar Salvar Modelo.
3- Acepta Salvar Modelo	4-Si no existe una ubicación de salva del modelo muestra un cuadro de dialogo para elegir la ubicación de salva del modelo.
5-Introduce los datos	6-Cierra el cuadro de dialogo mostrado y salva el modelo, luego lo cierra.

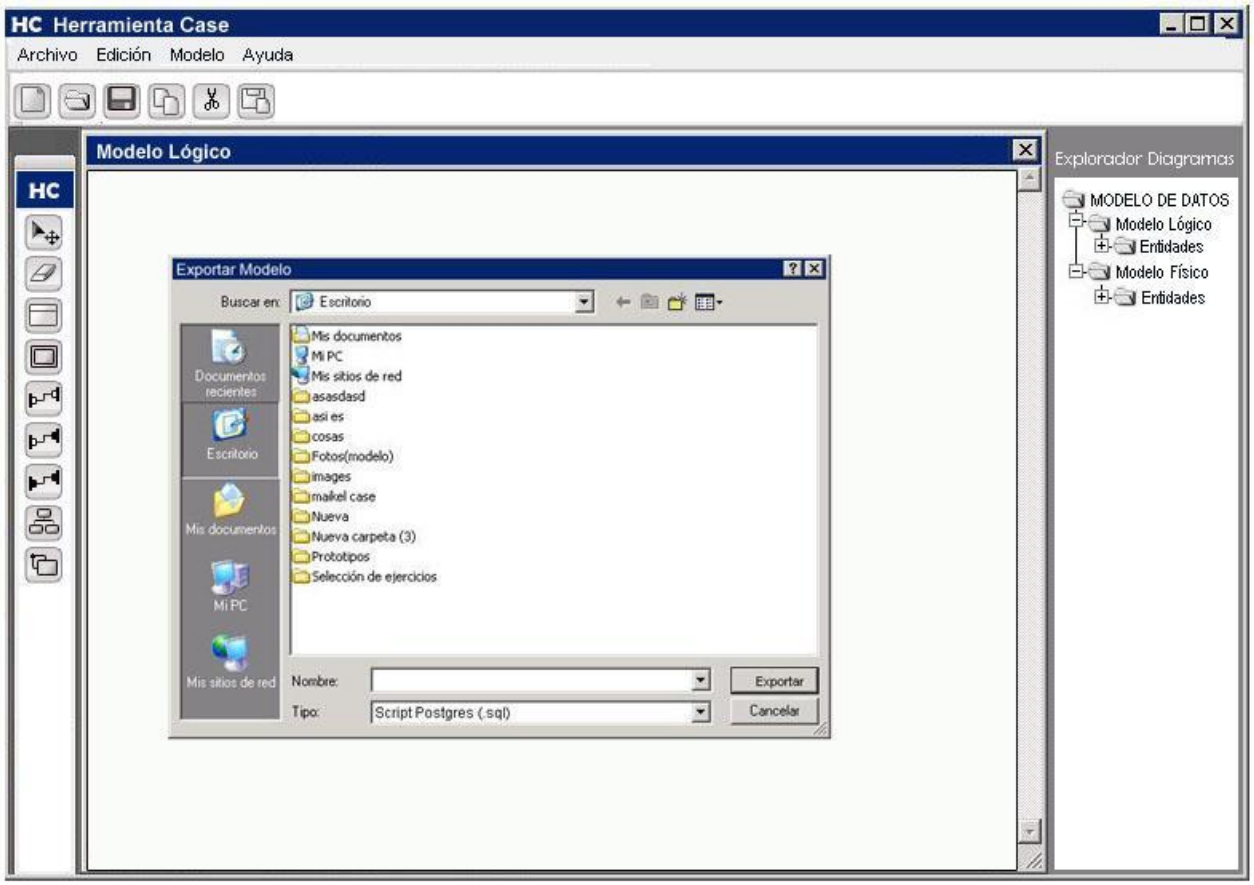


Flujo Alterno 1	
Acción del Actor	**Respuesta del Sistema**
3-Cancela Salvar modelo	4-Cierra el cuadro de dialogo mostrado y no salva el modelo, luego cierra el modelo.
Flujo Alterno 2	
Acción del Actor	**Respuesta del Sistema**
3-Cancela Salvar modelo	5- Si existe una ubicación de salva del modelo lo salva y luego lo cierra.
Poscondiciones	El Modelo queda cerrado.

CASO DE USO EXPORTAR MODELO A XML

Caso de Uso:	**EXPORTAR MODELO A XML**
Actores:	Usuario
Resumen:	El caso de uso se inicia cuando el usuario desea exportar el modelo de datos como archivo y con extensión .xml y termina cuando el archivo está exportado en el directorio donde fue ubicado.
Precondiciones:	No existe un modelo de datos.
Referencias	**R4**
Prioridad	Primario

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1- Selecciona la opción “Exportar Modelo a XML”.	2- Muestra un formulario para entrar el nombre del modelo con extensión XML y el directorio de ubicación.
3- Introduce los datos.	4- Cierra el formulario mostrado y exporta el modelo



Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
3-Cancela exportar modelo.	4-Cierra el formulario mostrado y no exporta el modelo
Poscondiciones	El modelo queda exportado.

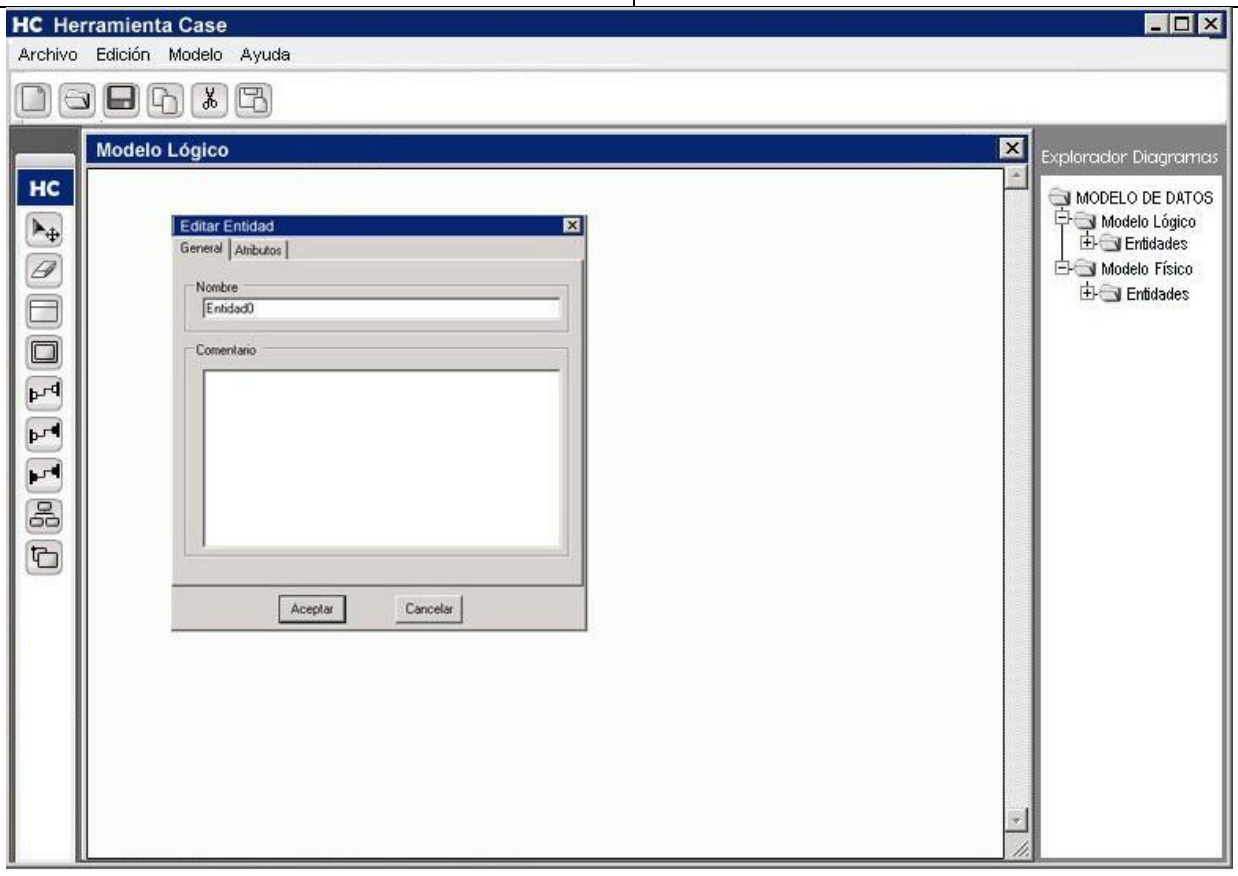
CASO DE USO CRUD ENTIDAD

Caso de Uso:	**CRUD ENTIDAD**
Actores:	Usuario
Resumen:	El caso de uso se inicia cuando el usuario se dispone a crear, modificar o eliminar entidades de un modelo de datos y termina cuando se obtiene

	un modelo de datos actualizado con más o menos entidades.
Precondiciones:	El modelo tiene que estar editándose.
Referencias	**R7.1, R7.2, R7.3**
Prioridad	Primario

Flujo Normal de Eventos

Acción del Actor	Respuesta del Sistema
1- Solicita Editar un modelo abierto.	2 –Muestra el modelo y además las opciones de: a) “crear entidad”. b) “modificar entidad seleccionada“. c) “eliminar entidad seleccionada “.
3-Selecciona una opción de gestión de entidades	4- Si decidió a: ir a la sección “crear”. b: ir a la sección “modificar “. c: ir a la sección “eliminar “.



Flujos Alternos

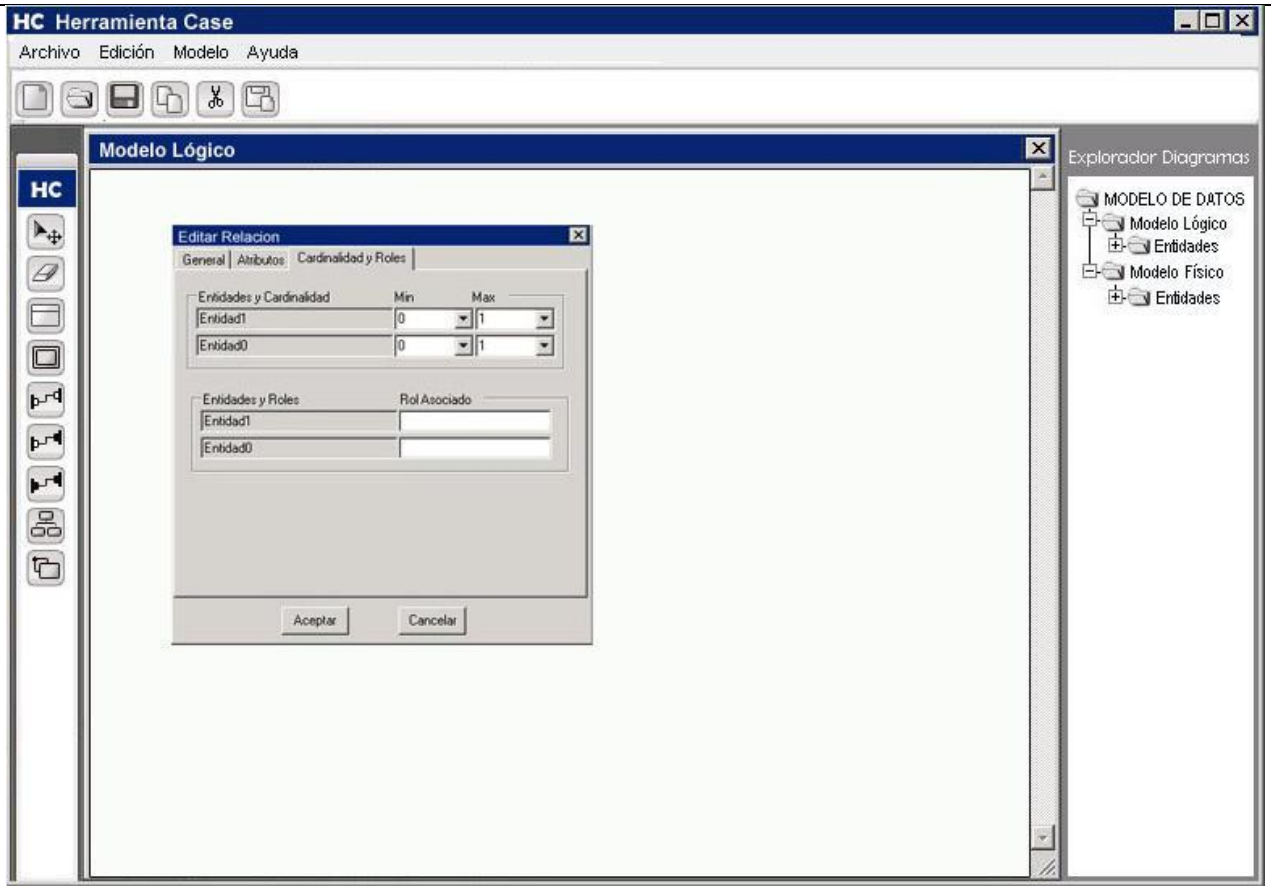
Acción del Actor	Respuesta del Sistema

3- No se selecciona ninguna de las opciones.	4- El sistema sigue mostrando el modelo.
Sección Crear	
Acción del Actor	**Respuesta del Sistema**
	5- Muestra un elemento grafico que representa la entidad y lo inserta en el modelo.
Sección Modificar	
Acción del Actor	**Respuesta del Sistema**
6- Edita el nombre, tipo y atributos de la entidad.	5- Muestra un formulario con los datos de la entidad en el cual permite editarlos.
7-Acepta los datos editados.	8-Verifica si los datos están correctos, si están correctos sigue con el paso 9. 9-El sistema actualiza los datos de la entidad.
Flujo Alterno 1	
Acción del Actor	**Respuesta del Sistema**
7- Solicita cancelar la edición de los datos de la entidad.	8-Cierra el formulario mostrado.
	9-Vuelve al paso 2.
Flujo Alterno 2	
Acción del Actor	**Respuesta del Sistema**
9- Acepta el mensaje de error.	8-Si los datos están incorrectos muestra mensaje de error.
	10-Vuelve al paso 2.
Sección Eliminar	
Acción del Actor	**Respuesta del Sistema**
6-Acepta eliminar entidad.	5- Muestra un formulario de confirmación de eliminar la entidad.
	7-Elimina la entidad que se le solicitó eliminar.
	8-Vuelve al paso 2.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
6- Cancela eliminar entidad.	7-Cierra el formulario mostrado y no elimina la

	entidad.
	8-Vuelve al paso 2.
Poscondiciones	Se crea una entidad nueva, se actualizan los datos de una entidad existente o se elimina una entidad existente.

CASO DE USO CRUD RELACION

Caso de Uso:	**CRUD RELACION**
Actores:	Usuario
Resumen:	El caso de uso se inicia cuando el usuario se dispone a crear, modificar o eliminar relaciones de un modelo de datos y termina cuando se obtiene un modelo de datos con sus respectivas relaciones.
Precondiciones:	El modelo de datos carece de relaciones o no posee relación entre sus entidades.
Referencias	**R9.1, R9.2, R9.3**
Prioridad	primario
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1- Solicita mostrar un modelo de datos que se este editando.	2-Muestra el modelo de datos solicitado con sus opciones de: a) crear relación b) modificar relación seleccionada, c) eliminar relacion seleccionada.
3- Selecciona la opción de crear, modificar o eliminar una relación.	4- Si decidió: a): ir a la sección “crear”. b) ir a la sección “modificar “. c) ir a la sección “eliminar “.



Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
3- No se selecciona ninguna de las opciones.	4- En sistema sigue mostrando el modelo de datos solicitado.
Sección Crear	
Acción del Actor	**Respuesta del Sistema**
	1- Muestra un elemento gráfico que representa la relación y lo inserta entre la entidad o las entidades asociadas.
Sección Modificar	
Acción del Actor	**Respuesta del Sistema**
6- Edita el nombre y el tipo de relación.	5- Muestra un formulario con los datos de la relación en la cual permite editarlos.

7-Acepta los datos editados.	8-Verifica si los datos están correctos, si están correctos sigue con el paso 9. 9-El sistema actualiza los datos de la relación.
Flujo Alterno 1	
Acción del Actor	**Respuesta del Sistema**
7- Solicita cancelar la edición de los datos de la relación.	8-Cierra el formulario mostrado.
Flujo Alterno 2	
Acción del Actor	**Respuesta del Sistema**
9- Acepta el mensaje de error y vuelve al paso 2.	8-Si los datos están incorrectos muestra mensaje de error.
Sección Eliminar	
Acción del Actor	**Respuesta del Sistema**
6-Acepta eliminar relación.	5- Muestra un formulario de confirmación de eliminar la relación.
	7-Elimina la relación que se le solicitó eliminar.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
6- Cancela eliminar relación.	7-Cierra el formulario mostrado y no elimina la relación
Poscondiciones	Se crea una relación.

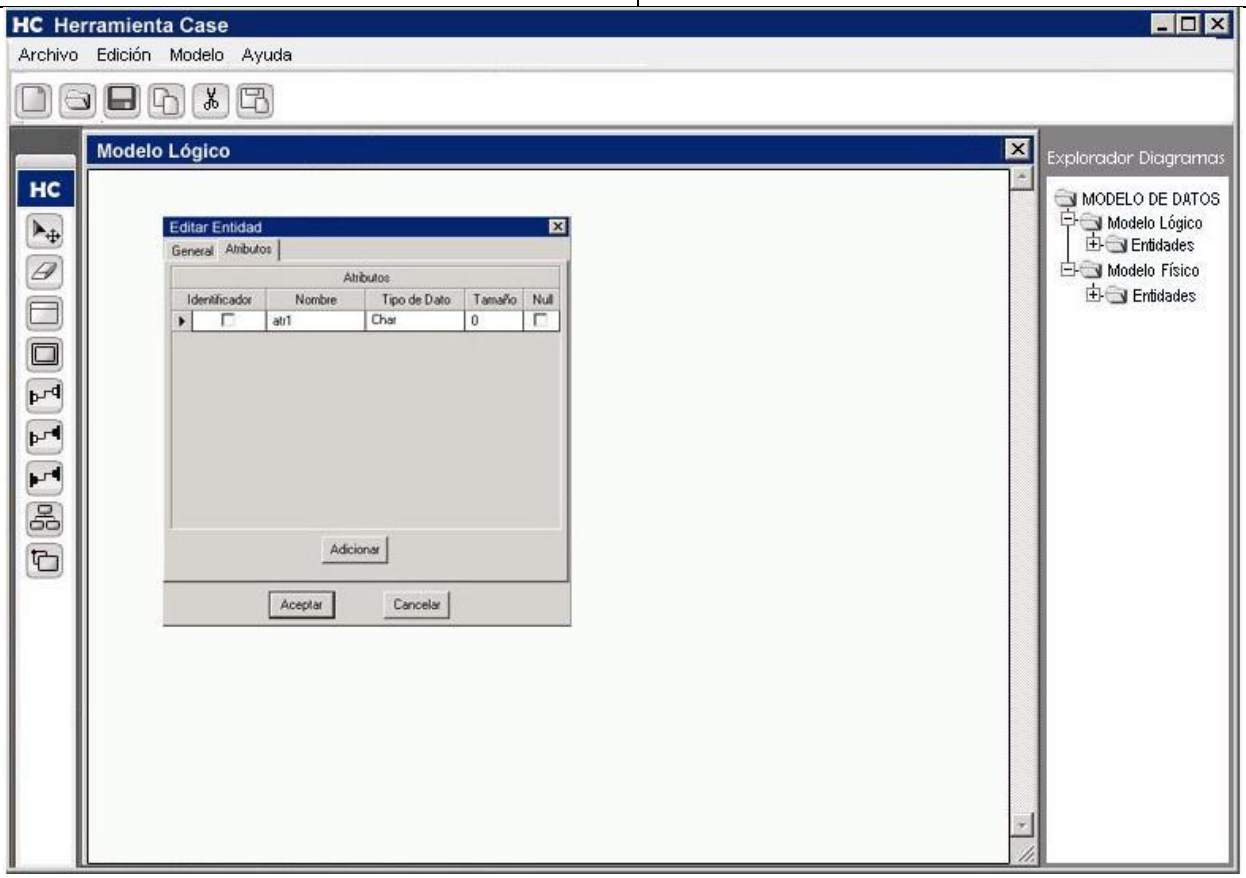
CASO DE USO CRUD ATRIBUTO

Caso de Uso:	**CRUD ATRIBUTO**
Actores:	Usuario
Resumen:	El caso de uso se inicia cuando el usuario se dispone a crear, modificar o eliminar atributos de una entidad ya creada y termina cuando esté creado, modificado y eliminado.

Precondiciones:	Debe existir al menos una entidad.
Referencias	R8.1, R8.2, R8,3, R7.1, R7.2, R7.3
Prioridad	primario

Flujo Normal de Eventos

Acción del Actor	Respuesta del Sistema
1- Solicita seleccionar una entidad para crear gestionar atributos.	2-Muestra la entidad solicitada con sus opciones de: a) crear atributo b) modificar atributo seleccionado c) eliminar atributo seleccionado.
3- Selecciona la opción de crear, modificar o eliminar un atributo.	4- Si decidió: a): ir a la sección “crear”. b) ir a la sección “modificar “. c) ir a la sección “eliminar “.



Flujos Alternos

Acción del Actor	Respuesta del Sistema

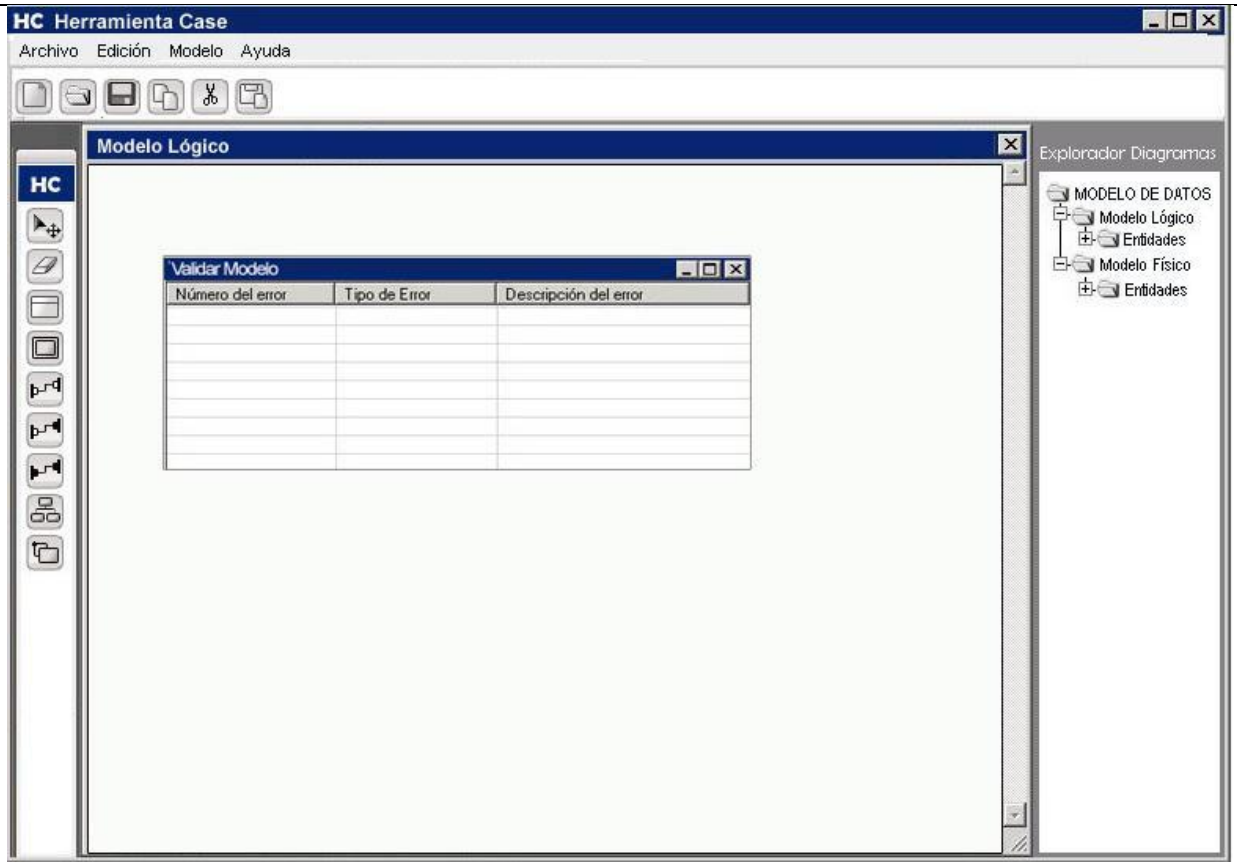
3- No se selecciona ninguna de las opciones.	4- En sistema sigue mostrando la entidad solicitada.
Sección Crear	
Acción del Actor	**Respuesta del Sistema**
	1- Muestra los campos requeridos para la inserción de un atributo.
Sección Modificar	
Acción del Actor	**Respuesta del Sistema**
6- Edita los campos del formulario modificar atributo.	5- Muestra un formulario con los datos de la entidad en el cual permite editarlos.
7-Acepta los datos editados.	8-Verifica si los datos están correctos, si están correctos sigue con el paso 9. 9-El sistema actualiza los datos de la entidad.
Flujo Alterno 1	
Acción del Actor	**Respuesta del Sistema**
7- Solicita cancelar la edición de los datos de la entidad.	8-Cierra el formulario mostrado.
Flujo Alterno 2	
Acción del Actor	**Respuesta del Sistema**
9- Acepta el mensaje de error y vuelve al paso 2.	8-Si los datos están incorrectos muestra mensaje de error.
Sección Eliminar	
Acción del Actor	**Respuesta del Sistema**
6-Acepta eliminar atributo.	5- Muestra un formulario de confirmación de eliminar los atributos.
	7-Elimina el atributo que se le solicitó eliminar.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
6- Cancela eliminar atributo.	7-Cierra el formulario mostrado y no elimina el atributo.
Poscondiciones	Se modifica la entidad y se actualizan los datos de una entidad existente

	en dependencia de si se creó se modificó o eliminó un atributo.

CASO DE USO VALIDAR MODELO

Caso de Uso:	**VALIDAR MODELO**
Actores:	Usuario
Resumen:	El caso de uso se inicia cuando el usuario se dispone a validar un modelo, termina cuando el usuario ve el informe de validación que consiste en notificación de errores en el modelo.
Precondiciones:	Debe existir al menos un modelo que se este editando.
Referencias	**R9**
Prioridad	Primario

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1- Solicita ver el informe de validación del modelo.	2-Valida el modelo y muestra al usuario el informe de validación.
2-Acepta cerrar el informe de Validación	3-Cierra el Informe de Validación.

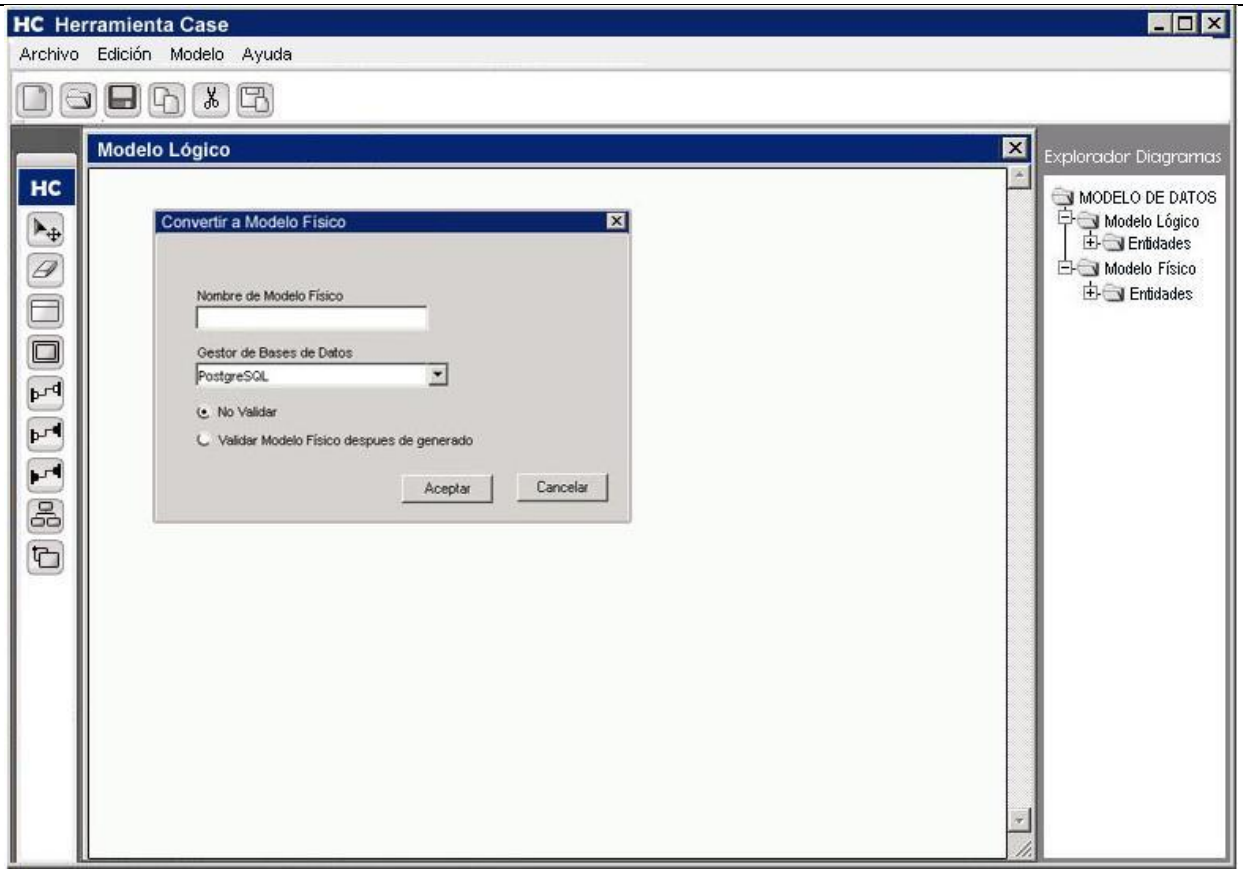


Poscondiciones	El usuario se informa de los errores que existen en el modelo.

CASO DE USO CONVERTIR A MODELO FISICO

Caso de Uso:	**CONVERTIR A MODELO FÍSICO**
Actores:	Usuario
Resumen:	El caso de uso se inicia cuando el usuario se dispone a generar un modelo físico a partir de un modelo lógico, termina cuando se genera el modelo físico.
Precondiciones:	Debe existir al menos un modelo lógico que se este editando.
Referencias	**R10**
Prioridad	Primario

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1- Selecciona la opción “Convertir a Modelo Físico”.	2- Muestra un formulario para entrar el nombre a dar al modelo físico y el tipo de gestor de bases de datos al que será generado, en este caso PostgreSQL, además de la opción de validar modelo Fisico después de generarlo.
3-Introduce los datos, no marcando la opción de generar validar modelo fisico.	4-El sistema genera el modelo Fisico y lo muestra.



Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
4- Marca la opción de validar modelo fisico después de generar.	5- Genera el modelo fisico y lo muestra. Valida el modelo Fisico. (Caso de uso Validar Modelo A partir del punto 2.)
Poscondiciones	El modelo físico queda generado y validado.

CONCLUSIONES

En este capítulo se comenzó a desarrollar la propuesta de solución, obteniéndose a partir de análisis de los procesos de negocio un listado con las propiedades que debe cumplir el sistema y las funcionalidades que este debe realizar, representándolas mediante un Diagrama de Casos de Uso del

Sistema y finalmente describiendo paso a paso todas las acciones del actor con los casos de uso que interactúa. Ahora se puede comenzar a construir el sistema, tratando de que se cumplan todos los requerimientos y las funciones que se han considerados necesarias en este capítulo.

CAPÍTULO 3: ANALISIS Y DISEÑO DEL SISTEMA

3.1 INTRODUCCIÓN

Para el desarrollo de un software es necesario contar con descripciones detalladas y saber como se satisfacen los requerimientos y las restricciones. En este capítulo se describen los elementos más importantes correspondientes a la etapa de análisis y diseño del sistema, utilizando para su modelado el Lenguaje Unificado de Modelación (UML), que permite representar la expansión de los casos de uso y el modelo conceptual, también se presentarán los diagramas de interacción (secuencia) correspondientes a cada caso de uso y los diagramas de clases de análisis y diseño del sistema.

3.2 DIAGRAMAS DE CLASES DEL ANALISIS

En el diagrama de clases del análisis se representa una abstracción de una o varias clases y/o subsistemas en el diseño del sistema. Estas clases se clasifican en:

- **De Interfaz:** son usadas para modelar la interacción entre el sistema y sus actores.
- **De Entidad:** son usadas para modelar información que persiste en el tiempo o tiene una larga vida.
- **De Control:** estas clases realizan la coordinación, secuenciado de transacciones y, en definitiva, el control sobre otros objetos del sistema. (1)

Los Diagramas de Clases del Análisis se podrán consultar en el **(ANEXO I)**.

3.3 PATRONES DE DISEÑO

Los patrones son parejas de problema/solución con un nombre, que codifican buenos principios y sugerencias relacionados frecuentemente con la asignación de responsabilidades. Asignar correctamente las responsabilidades es muy importante en el diseño orientado a objetos. (11)

En el diseño de la herramienta se tuvieron en cuenta los patrones GRASP (General Responsibility Asignment Software Patterns) principalmente los patrones Experto y Controlador. El primero plantea que siempre se debe asignar una responsabilidad al experto en información, o sea, la clase con toda la información necesaria para llevarla a cabo. El segundo expresa asignar la responsabilidad del manejo de un mensaje de los eventos de un sistema a una clase controladora.

3.3.1 PATRON MODELO VISTA CONTROLADOR

La arquitectura Modelo Vista Controlador (MVC) divide una aplicación interactiva en 3 áreas: procesamiento, salida y entrada. Para esto, utiliza las siguientes abstracciones:

- **Modelo (Model)**: Encapsula los datos y las funcionalidades. El modelo es independiente de cualquier representación de salida y/o comportamiento de entrada.

- **Vista (View)**: Muestra la información al usuario. Pueden existir múltiples vistas del modelo. Cada vista tiene asociado un componente controlador.

- **Controlador (Controller)**: Reciben las entradas, usualmente como eventos que codifican los movimientos o pulsación de botones del ratón, pulsaciones de teclas, etc. Los eventos son traducidos a solicitudes de servicio ("**service requests**") para el modelo o la vista.

Se han desarrollado a lo largo de los años, desde la presentación de este patrón a la comunidad científica 3 variantes fundamentales, que se presentan brevemente a continuación.

- **Variante 1:** Variante en la cual no existe ninguna comunicación entre el Modelo y la Vista y esta última recibe los datos a mostrar a través del Controlador.

- **Variante 2:** Variante en la cual se desarrolla una comunicación entre el Modelo y la Vista, donde esta última al mostrar los datos lo busca directamente en el Modelo, dada una indicación del Controlador, disminuyendo el conjunto de responsabilidades de este último.

- **Variante 3:** Variante en la cual se diversifica las funcionalidades del Modelo teniendo en cuenta las características de las aplicaciones multimedia, donde tienen un gran peso las medias utilizadas en estas.

Para la realización de la arquitectura del software se empleó la Variante 1 del patrón MVC.

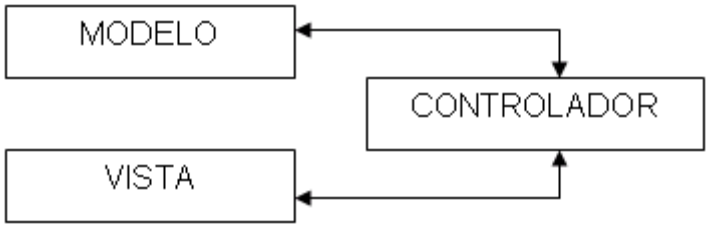


Fig. 5 Variante 1 de patrón MVC

3.3.2 PATRON SINGLETON

El Singleton es quizás el más sencillo de los patrones que se presentan en el catálogo del Gof. Es también uno de los patrones más conocidos y utilizados. El patrón Singleton garantiza que una clase sólo tenga una instancia y proporciona un punto de acceso global a ésta instancia.

Intención

Garantiza que una clase sólo tenga una instancia y proporciona un punto de acceso global a ella.

Aplicabilidad

- Debe haber exactamente una instancia de una clase y ésta deba ser accesible a los clientes desde un punto de acceso conocido.
- La única instancia debería ser extensible mediante herencia y los clientes deberían ser capaces de utilizar una instancia extendida sin modificar su código.

Consecuencias

- Acceso controlado a la única instancia. Puede tener un control estricto sobre cómo y cuando acceden los clientes a la instancia.
- Espacio de nombres reducido. El patrón Singleton es una mejora sobre las variables globales.
- Permite el refinamiento de operaciones y la representación. Se puede crear una subclase de Singleton.
- Permite un número variable de instancias. El patrón hace que sea fácil cambiar de opinión y permitir más de una instancia de la clase Singleton.

3.4 MODELO DEL DISEÑO

El diseño es un refinamiento del análisis que tiene en cuenta los requisitos no funcionales. Este modelo está muy cercano al modelo de implementación, lo que es natural para guardar y mantener el modelo de diseño a través del ciclo de vida completo del software. Entre los propósitos del diseño se encuentran adquirir una comprensión de los aspectos relacionados con los requisitos no funcionales, y restricciones relacionadas con los lenguajes de programación, componentes reutilizables, sistemas operativos, tecnologías de distribución y concurrencia y tecnologías de interfaz de usuario. Además de descomponer los trabajos de implementación en partes más manejables, que puedan ser llevadas a cabo por diferentes equipos de desarrollo.

3.4.1 DIAGRAMAS DE CLASES DEL DISEÑO

Las clases de diseño se especifican utilizando la sintaxis del lenguaje de programación elegido y tienen correspondencia directa con los métodos en la implementación. Un diagrama de clases de diseño es una representación más concreta que el diagrama de clases del análisis, representa la parte estática del sistema y las clases y sus relaciones.
El Diagrama de Clase del Diseño se podrá consultar en el **(ANEXO II)**.

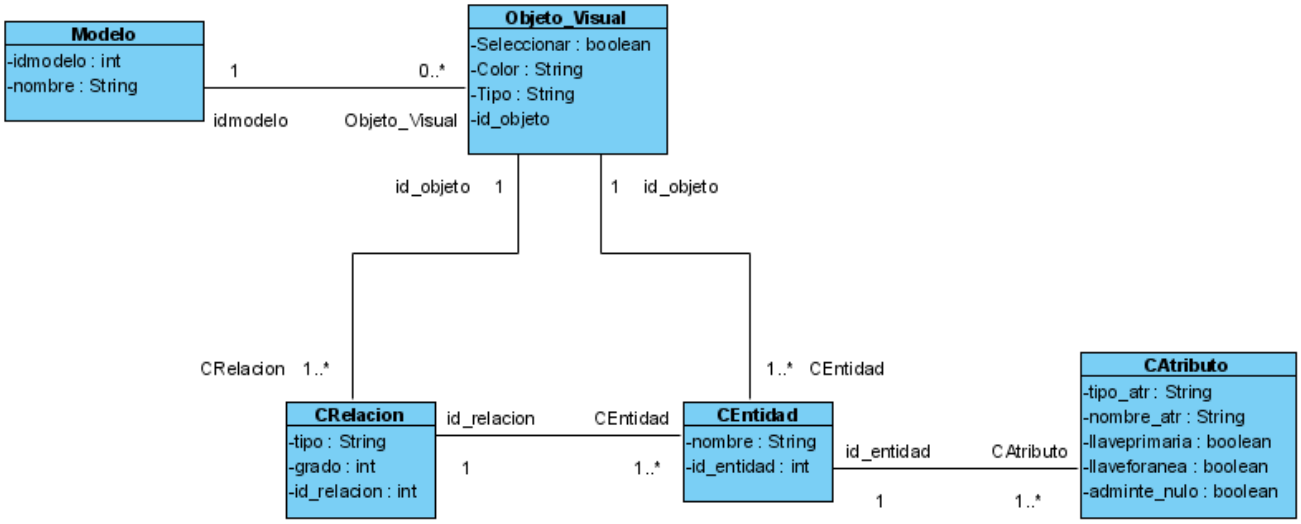
3.5 DIAGRAMAS DE INTERACIÓN

Los diagramas de interacción no son más que una descripción del modo en el que cada operación detectada en los diagramas de secuencia lleva a cabo sus responsabilidades y modifica el estado del sistema. En UML los diagramas de interacción pueden representarse a través de los Diagramas de Colaboración y/o de los Diagramas de Secuencia.

El tipo de diagrama seleccionado para construir los diagramas de interacción fue el de Secuencia, debido a que muestra cómo los objetos se comunican unos con otros en una secuencia de tiempo, qué sucede en cada momento, y para ello contienen objetos con sus ciclos de vida y los mensajes que se envían entre ellos ordenados secuencialmente. (12)

Los Diagramas de Secuencia del Diseño se podrán consultar en el **(ANEXO III).**

3.6 MODELO DE DATOS



3.7 CONCLUSIONES

En este capítulo se realizó la descripción de las clases necesarias para la futura implementación del sistema. Se obtuvieron los diagramas de clases de análisis, los diagramas de clases de diseño y los de interacción (secuencia) atendiendo a los patrones de diseño mencionados.

CONCLUSIONES

- Se concluye el análisis y diseño de una herramienta para modelado de bases de datos, lográndose así el cumplimiento del objetivo principal del presente trabajo de diploma.
- El estudio y análisis de varias de las herramientas CASE para modelado y generación de bases de datos disponibles hoy en el mercado internacional y en la UCI, sirvieron de apoyo para entender cuales fueron las funcionalidades que estuvieron acorde para incluir en la herramienta a diseñar y dar respuesta a la problemática existente.
- Luego del análisis de de las tecnologías mas usadas en la actualidad para el desarrollo de sistemas informáticos similares, se propone utilizar Java como lenguaje de programación, y PostgreSQL como gestor de bases de datos por todos los beneficios y facilidades que provee.
- Se llegó a la conclusión además que la metodología idónea para llevar a cabo es el Proceso Unificado de Rational (RUP).
- No se pudieron identificar bien los procesos que ocurren en el negocio por lo que se modeló dicho negocio a través de un modelo de dominio, en el cual se mostraron los principales conceptos al usuario para lograr un mejor entendimiento.
- Se definieron los requerimientos del sistema, tanto funcionales como no funcionales, y a continuación se elaboró el modelo de casos de uso del sistema, describiéndose cada caso de uso para un mejor conocimiento de las funcionalidades que brindan.
- Se diseñó el sistema a través de diagramas de clases de análisis, diagramas de clases de diseño y diagramas de interacción (secuencia).
- Se puede decir que la motivación para desarrollar la herramienta CASE analizada y diseñada en el presente trabajo de diploma surgió para dar respuesta al problema del llamado licenciamiento de programas, pues la adquisición de una herramienta CASE para este propósito en el mercado mundial de software oscila entre los 1500 y 2100 dólares, no es nada favorable para un país en vía de desarrollo como Cuba depender de estas herramientas para el desarrollo de sus sistemas informáticos teniendo en su propia casa el capital humano preparado y la tecnología adecuada para desarrollar herramientas similares y así lograr que nuestros sistemas informáticos se mantengan más seguros y estables.

RECOMENDACIONES

Los objetivos generales de este trabajo fueron alcanzados, pero durante el desarrollo del análisis y diseño de la herramienta, han surgido ideas que sería recomendable tener en cuenta para su futuro perfeccionamiento:

- Implementar la herramienta lo antes posible para ponerla en práctica y luego según las pruebas pertinentes que se hagan saber si realmente cumplió con el objetivo principal planteado.
- Adicionarle posteriormente nuevas funcionalidades como por ejemplo que la base de datos que se modele pueda ser generada directamente para el gestor PostgreSQL, sin tener primeramente que hacerse una salva en el directorio de ubicación donde se esté trabajando y luego llevarla al gestor antes mencionado.
- No limitar la herramienta a que solamente pueda generar las bases de datos modelada para el gestor PostgreSQL, sino para otros gestores como SQL, Oracle etc.

.

REFERENCIAS BIBLIOGRÁFICAS

1. **Ivar Jacobson, Grady Booch y James Rumbaugh.** *El proceso unificado de software.* 1. s.l. : Pearson Educación, S.A, 2000.
2. **Ivar Jacobson, Grady Booch, James Rumbaugh.** *El proceso Unificado de Desarrollo de Software.* 2005. págs. 23-78. Vol. 1.
3. **Sanchez.** *Metodologías de Desarrollo de Software.* s.l. : Grupo Informatizate. Vol. DOI.
4. **Toranzo, Yoandro Hechevarría.** *Sistema Automatizado para la Planificación Material y finaciera del MINFAR.* Facultad de Ingeniería Industrial Centro de Estudios de Ingeniería de Sistemas (CEIS).
5. **Ivar Jacobson, Grady Booch y James Rumbaugh.** *El proceso unificado de desarrollo de software.* 1. s.l. : Félix Varela, 2004.
6. **Booch, Grady.** *UML in Action Communications of the ACM.* 1999. Vol. 42.
7. **Gregor Engels, Reiko Heckel and Stefan Sauer.** *UML-A Universal Languaje?* Heidelberg, Berlin : s.n., 2000.
8. **Nenad Medvidovic, David S Rosenblum, David F Redmiles and Jason E Robbins.** *Modeling Software Architectures in the Unified Modeling Language. ACM Transactions on Software Engineering and Methodology.* 2002. Vol. 11.
9. **Kendall.** Análisis y Diseño de Sistemas. [En línea] 10 de 12 de 2007.
10. **Fernando Berzal Galiano, Francisco Cortijo Bon.** *XML.* 2005.
11. **Larman.** *UML y Patrones.* Vol. 1.
12. **Sofía Álvarez, Anaisa Hernández.** . *Metodología para el desarrollo de aplicaciones con tecnología Orientada a Objetos utilizando notación UML.* La Habana : s.n., 2000.

BIBLIOGRAFÍA

1. *Tecnología CASE.* http://www.iscmolina.com/Herramientas%20CASE.htmt (5/12/2007)

2. *Beneficios de las herramientas CASE.* http://www.monografias.com/trabajos14/herramicase/herramicase.shtml#herr (22/11/2007).

3. *Comparación de herramientas CASE.* http://html.rincondelvago.com/herramientas-case_4.html (5/11/2007)

4. *Plataforma de Desarrollo Java.* http://www.dcc.uchile.cl/~lmateu/Java/Transparencias/.java/all.htm (5/11/2007).

5. *DB Designer 4.* http://www.fabforce.net (10/12/2007)

6. *El lenguaje Java.* http://www.lenguajes-de-programacion.com/programacion-java.shtml (10/1/2008)

7. *Características del lenguaje Java.* http://tikal.cifn.unam.mx/~jsegura/LCGII/java3.htm (12/1/2008)

8. *XML, como funciona.* http://manuales.dgsca.unam.mx/xml/c%F3mo_funciona.html (16/1/2008)

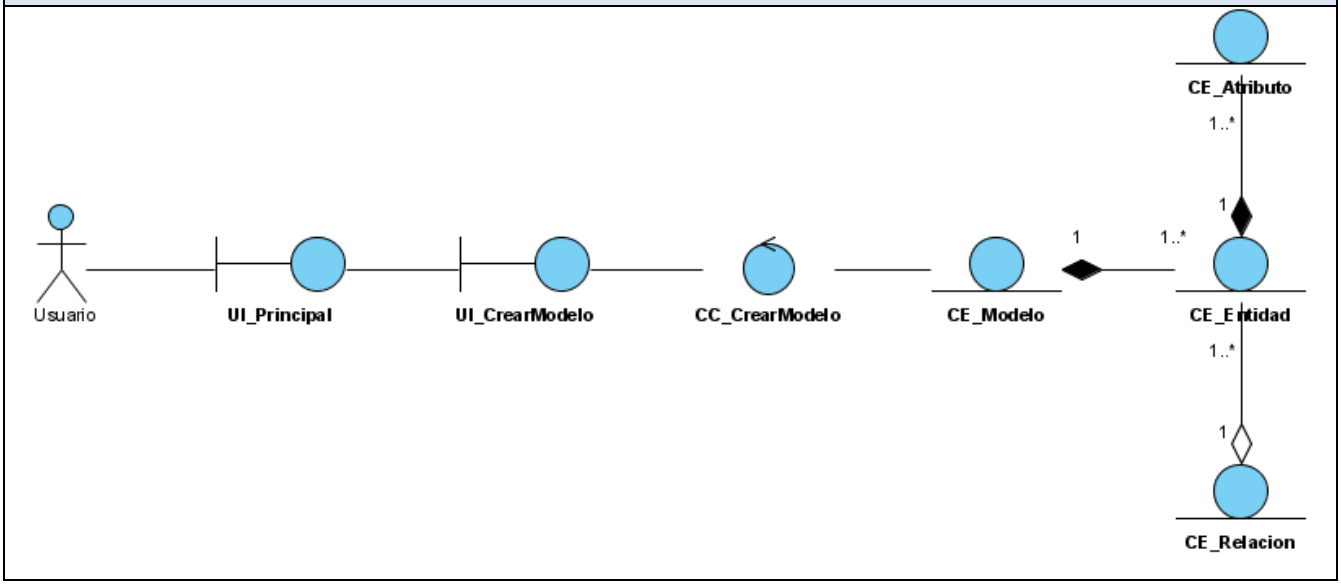
9. *Patrón Modelo Vista Controlador (MVC).* http://www.monografias.com/trabajos43/patron-modelo-vista. (16/1/2008)

10. *Patrones Gof. Singleton.* *http://www.manual-java.com/codigos-java/utilizando-patron-singleton.html* (10/5/2008)

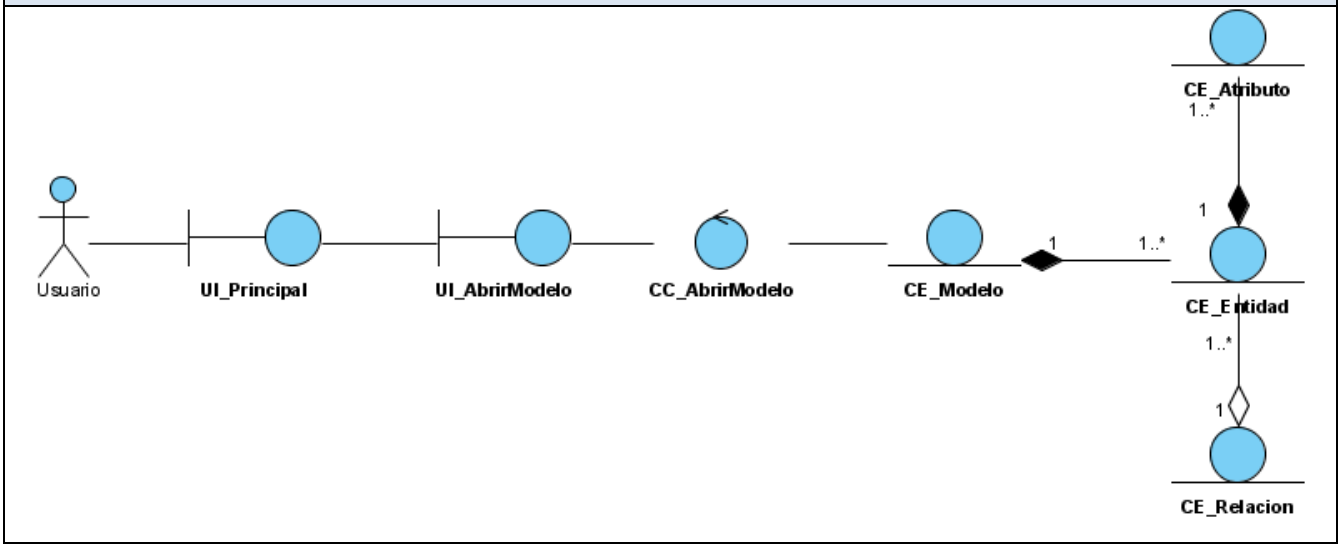
ANEXOS

ANEXO I. DIAGRAMAS DE CLASES DEL ANALISIS

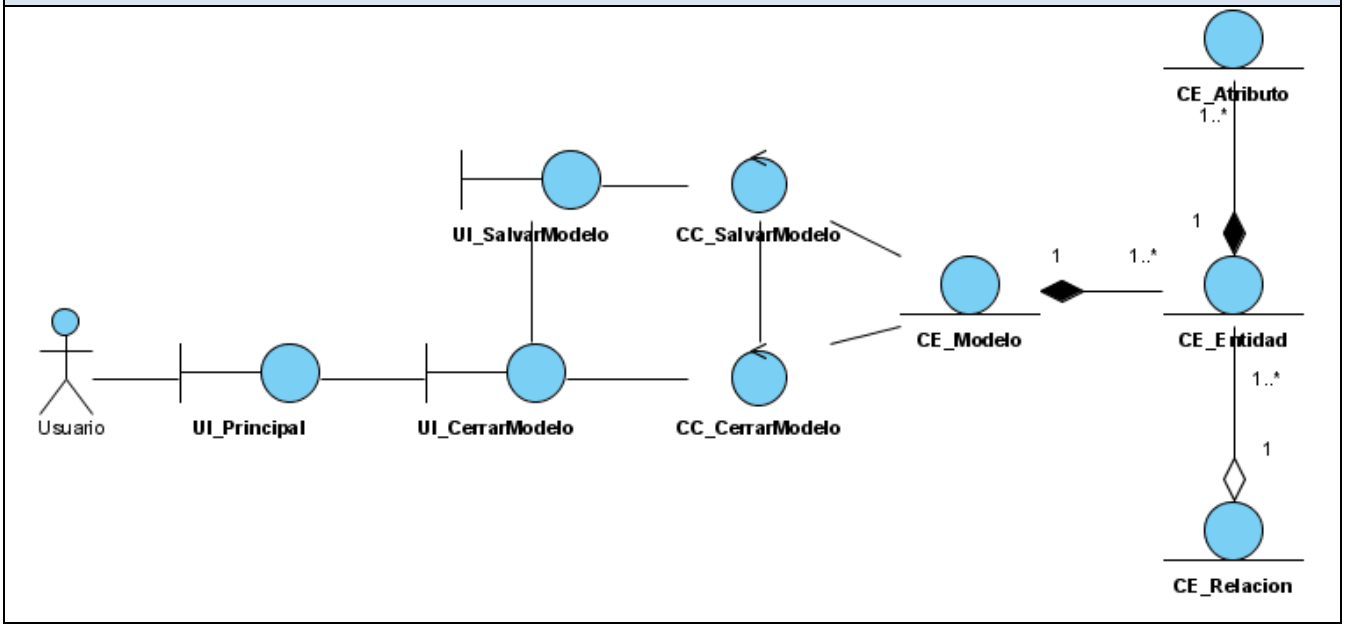
DCA CREAR NUEVO MODELO



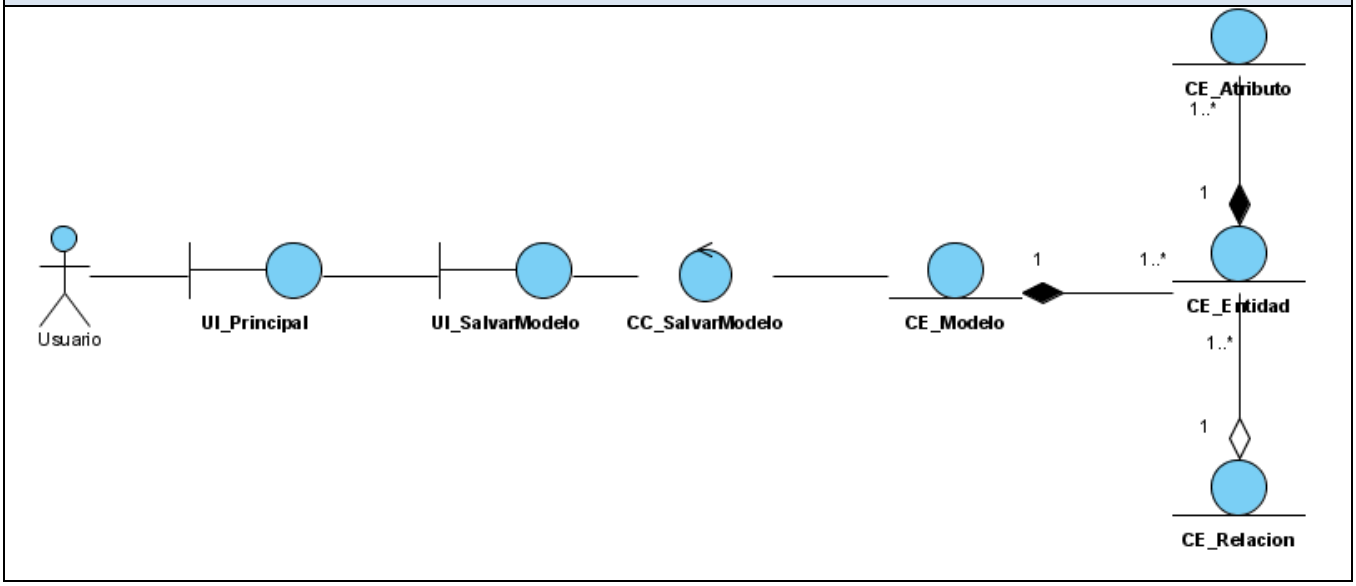
DCA ABRIR MODELO



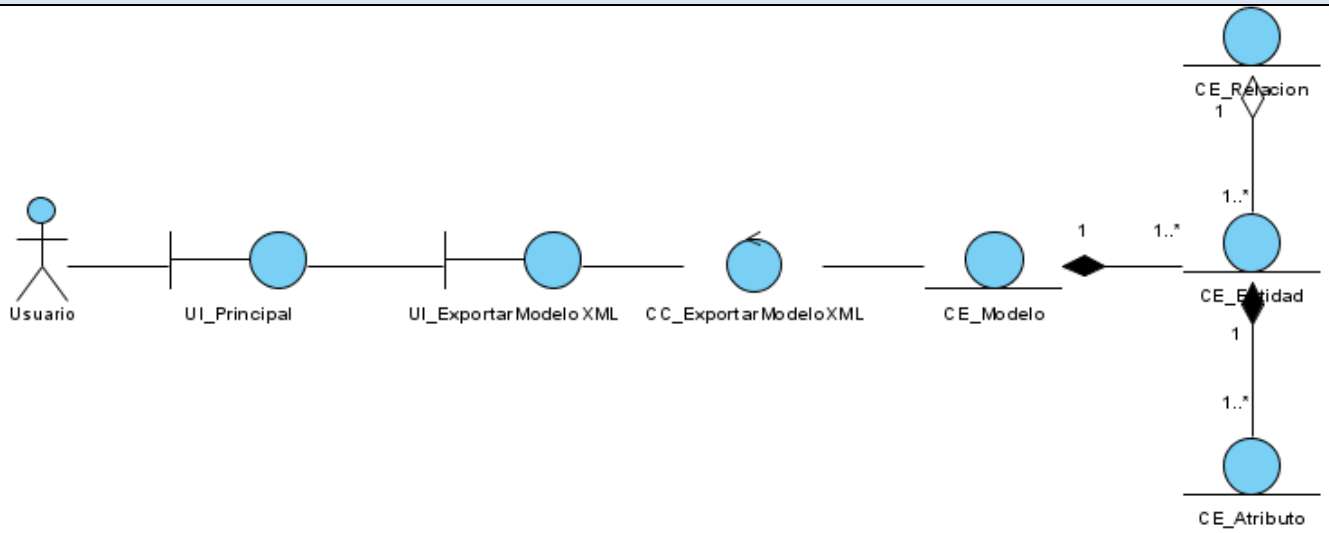
DCA CERRAR MODELO



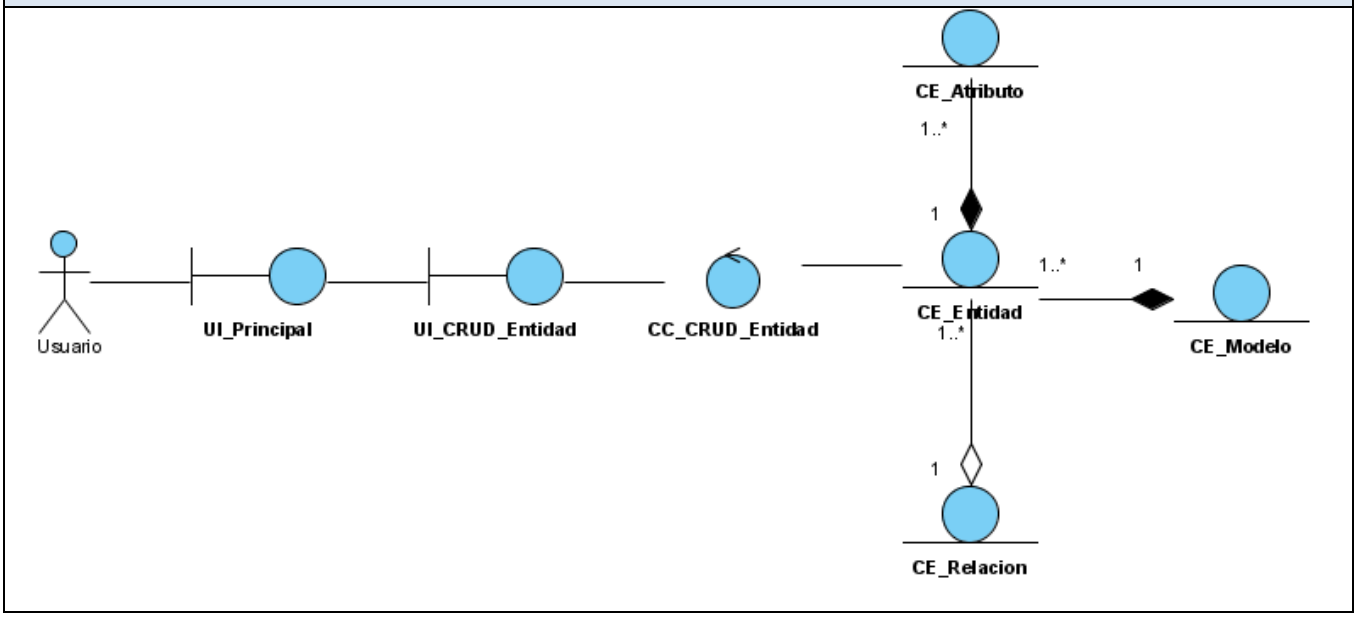
DCA SALVAR MODELO



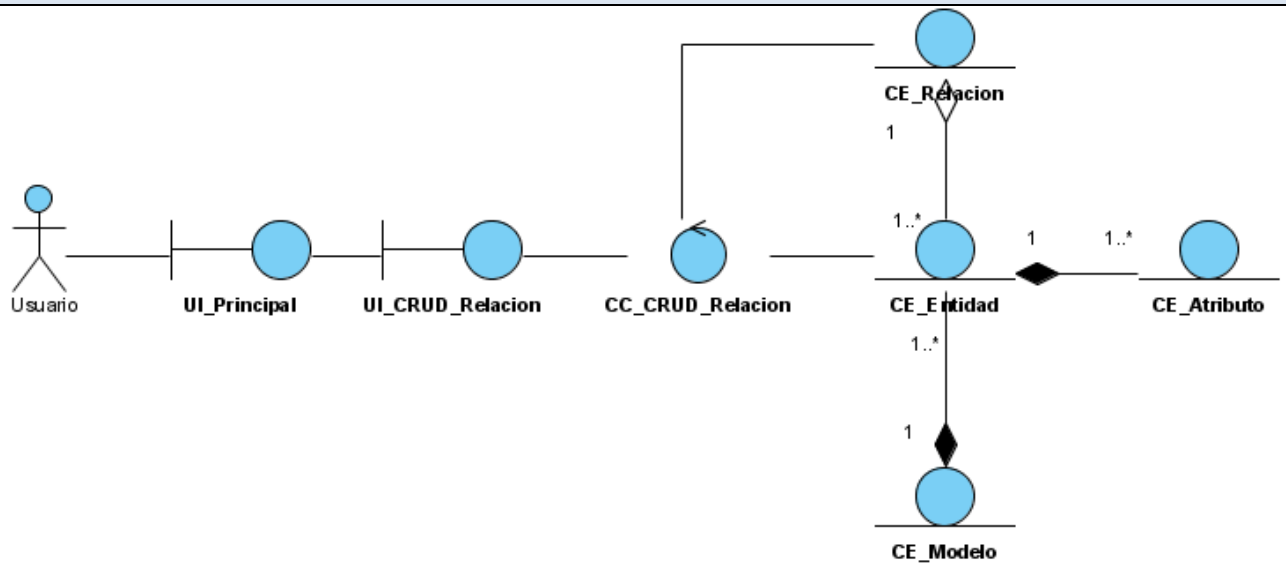
DCA EXPORTAR MODELO A XML



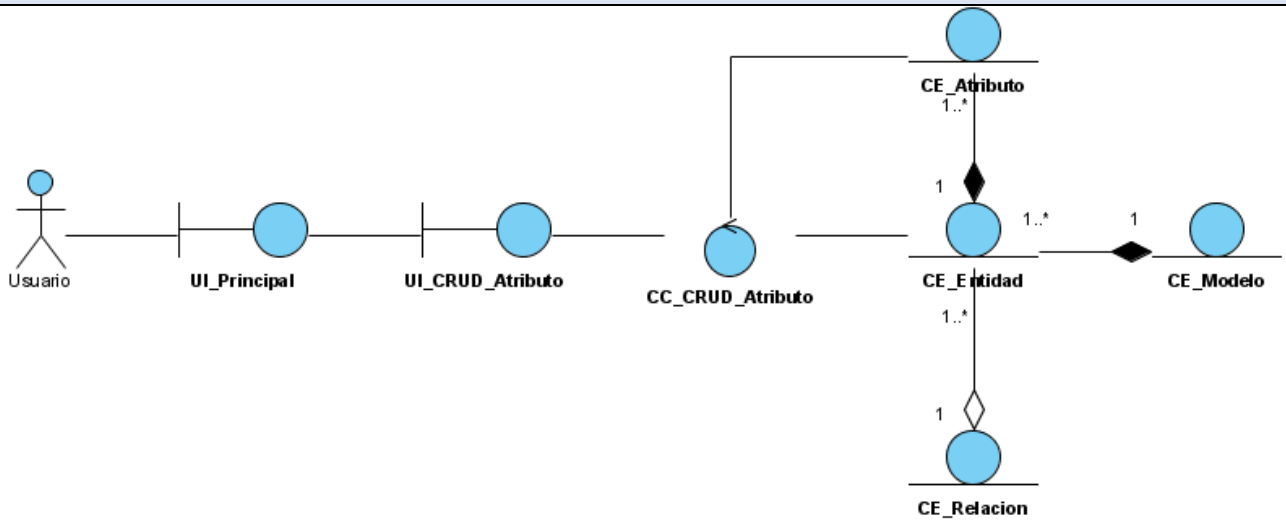
DCA CRUD ENTIDAD



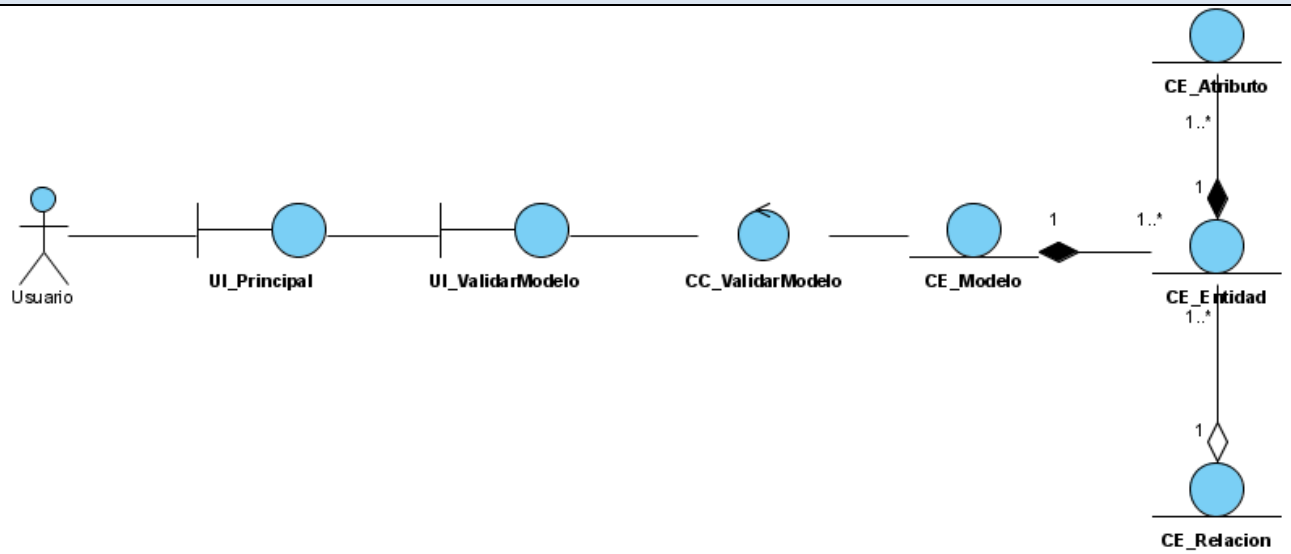
DCA CRUD RELACION



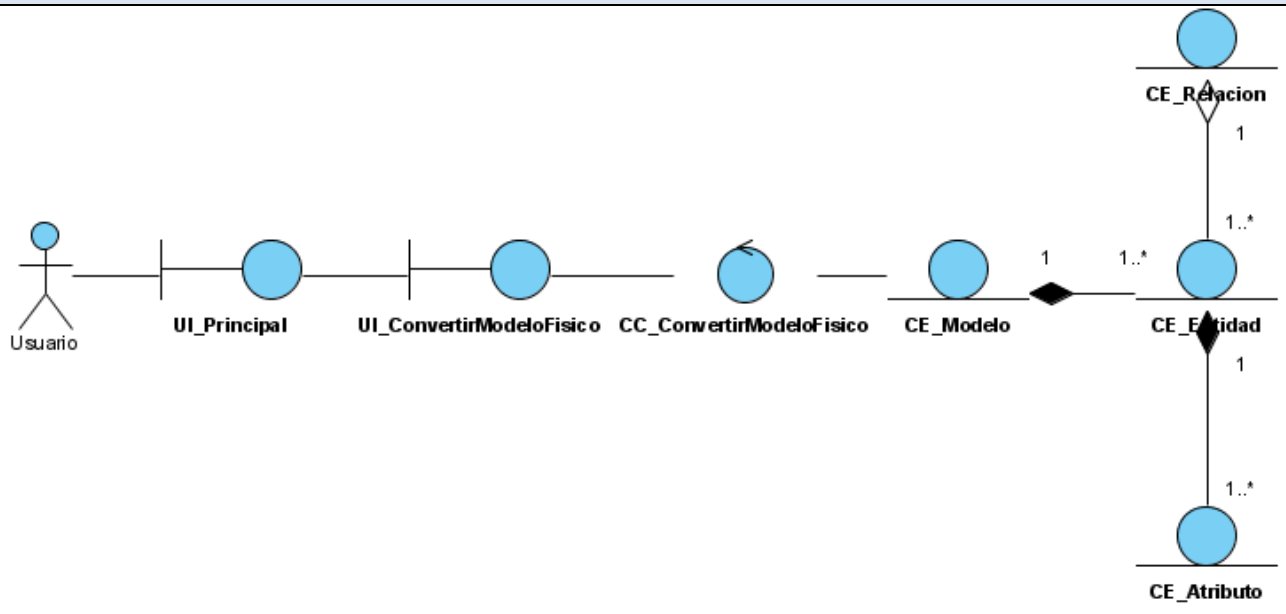
DCA CRUD ATRIBUTO



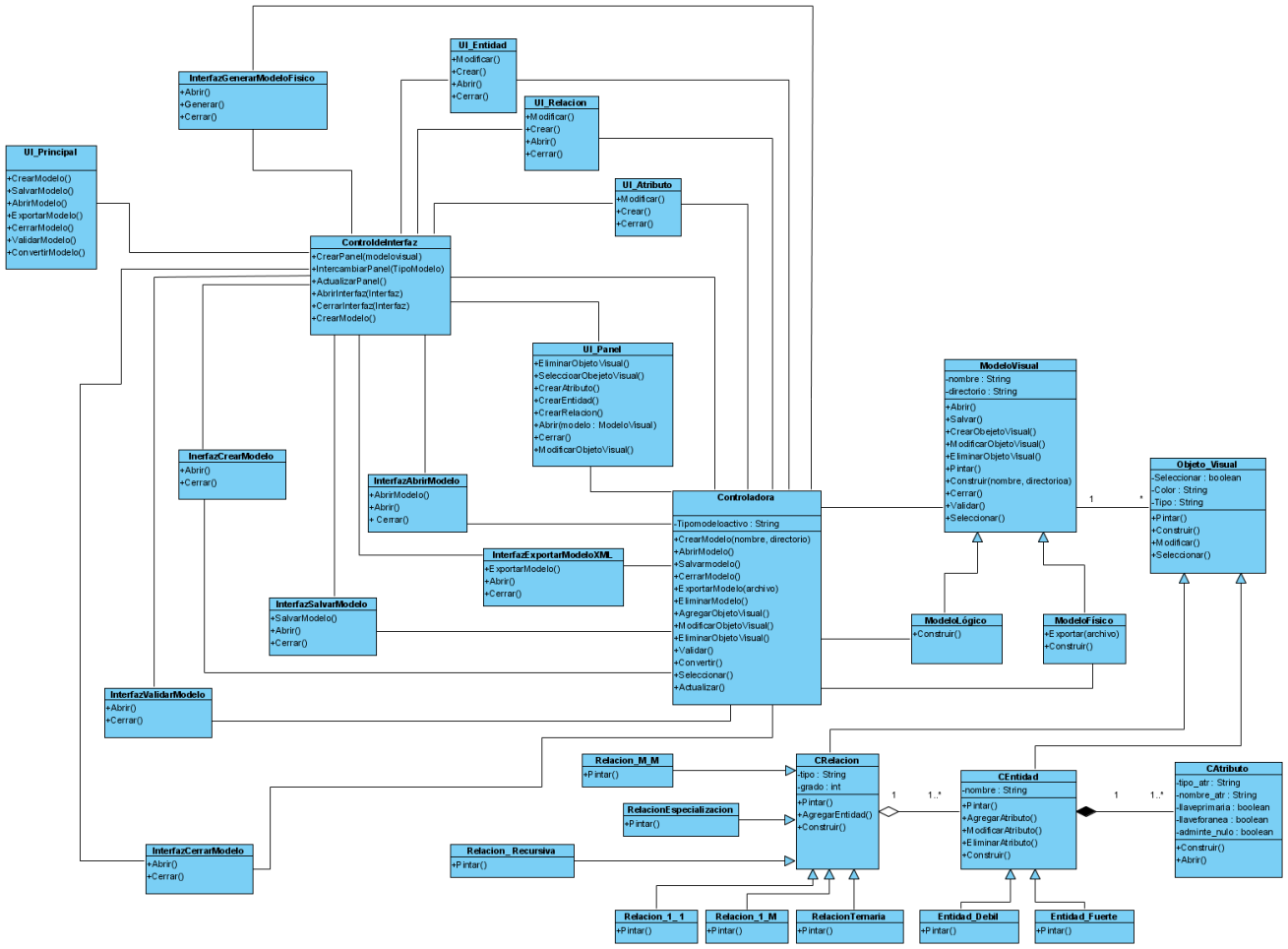
DCA VALIDAR MODELO



DCA CONVERTIR A MODELO FISICO

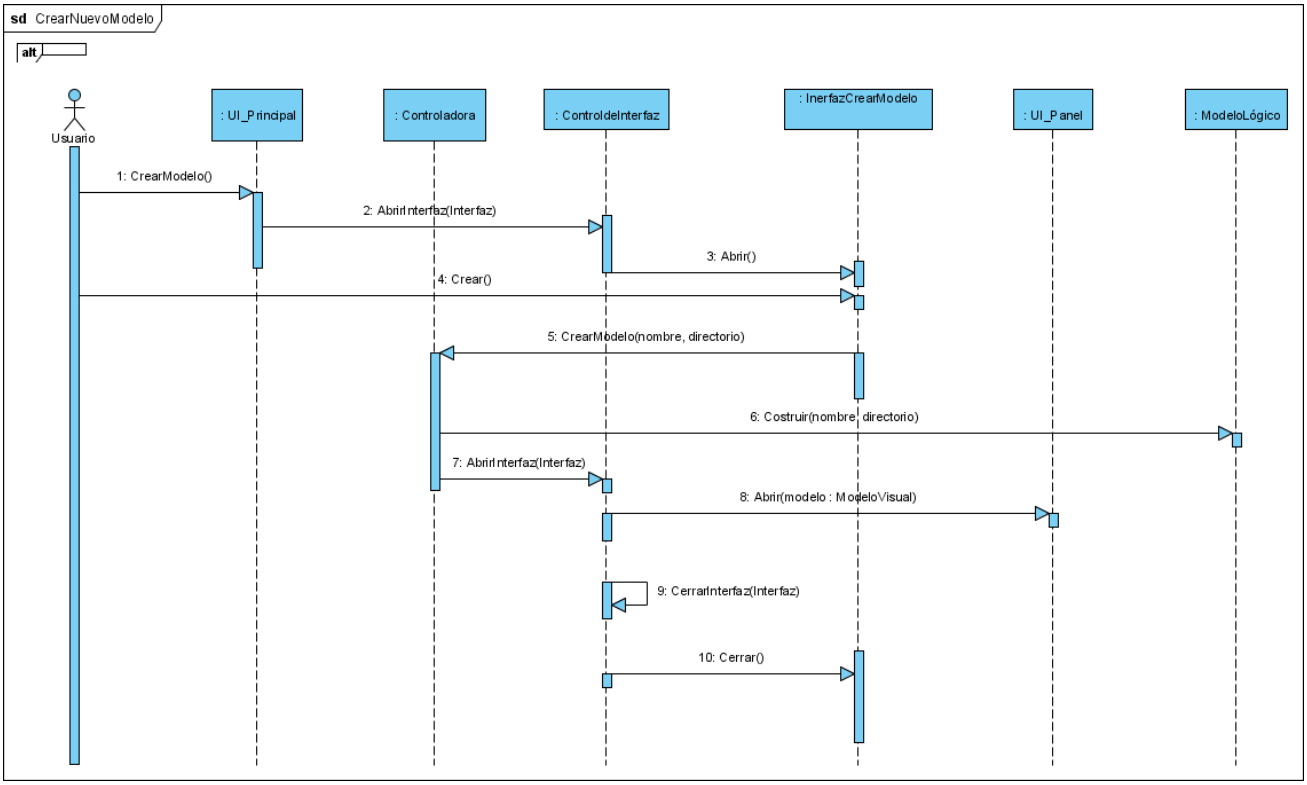


ANEXO II. DIAGRAMA DE CLASE DEL DISEÑO

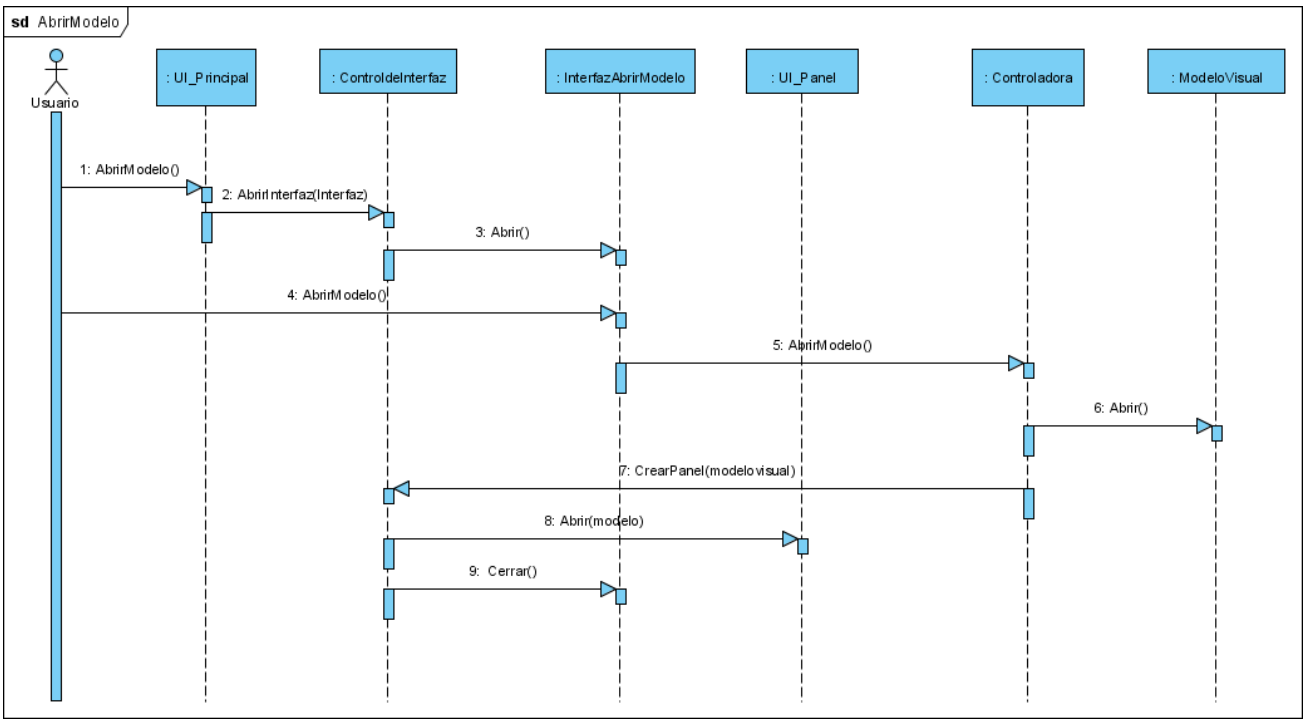


ANEXO III. DIAGRAMAS DE SECUENCIA DEL DISEÑO

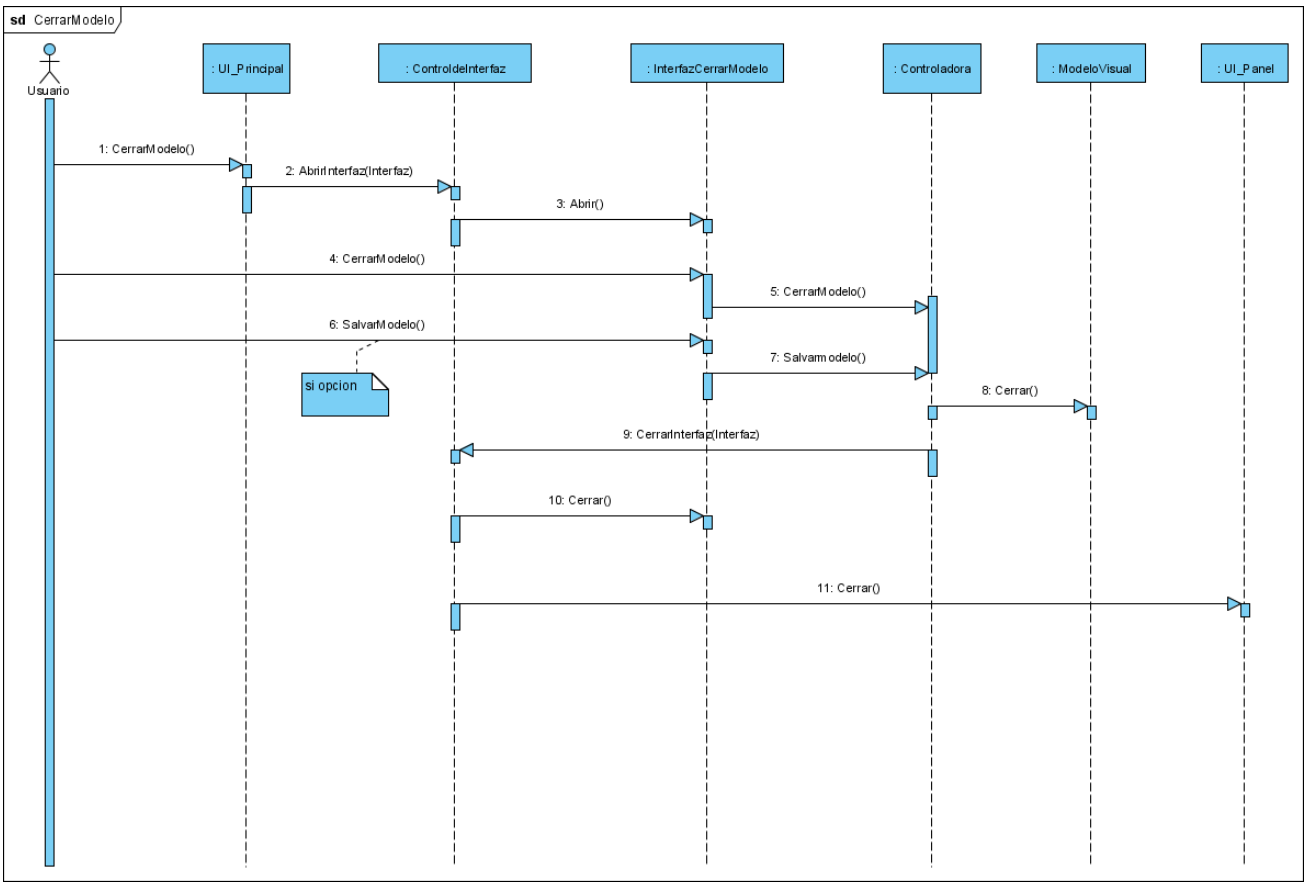
CREAR NUEVO MODELO



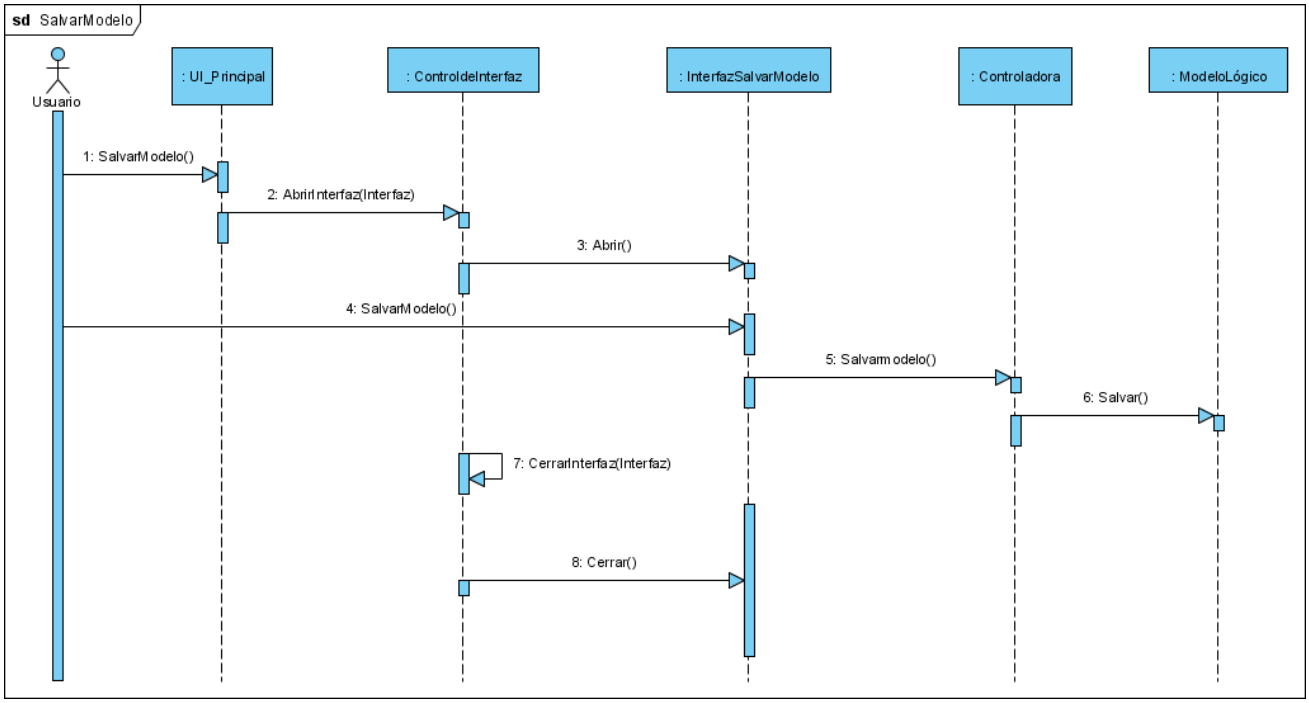
ABRIR MODELO



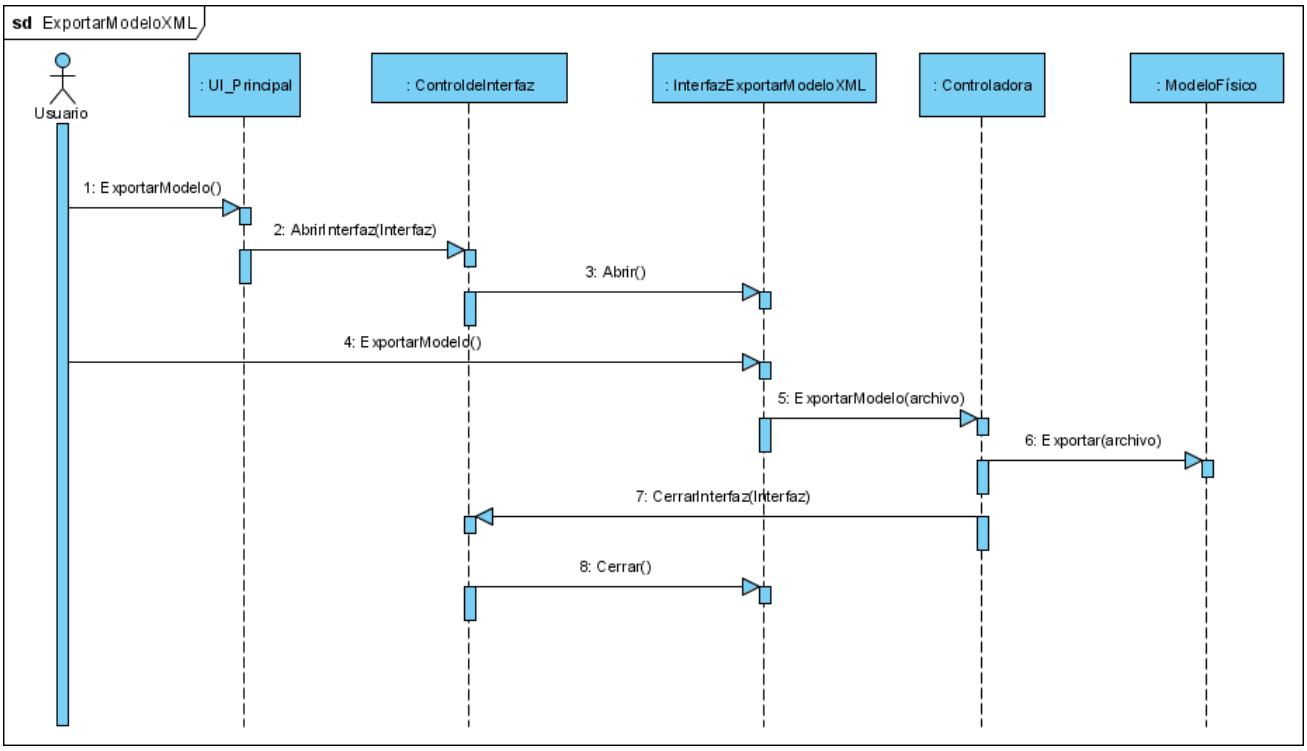
CERRAR MODELO



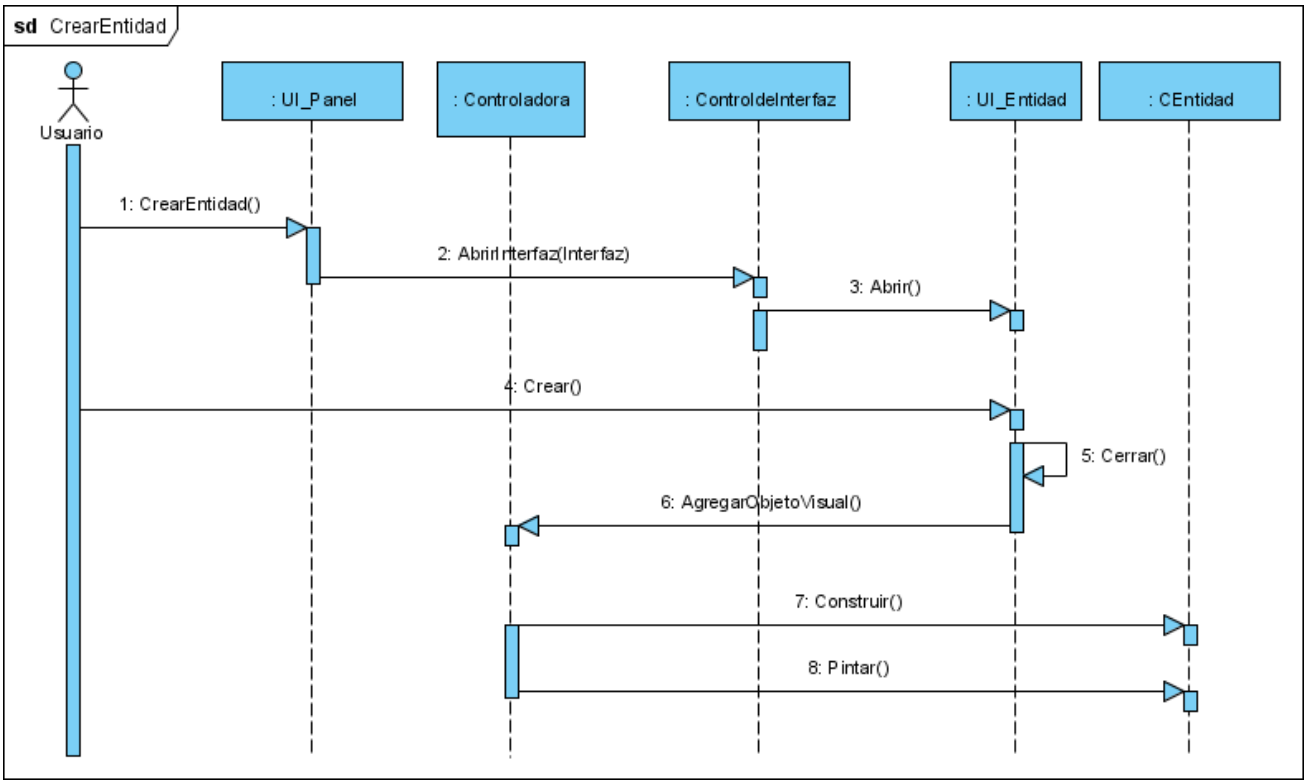
SALVAR MODELO



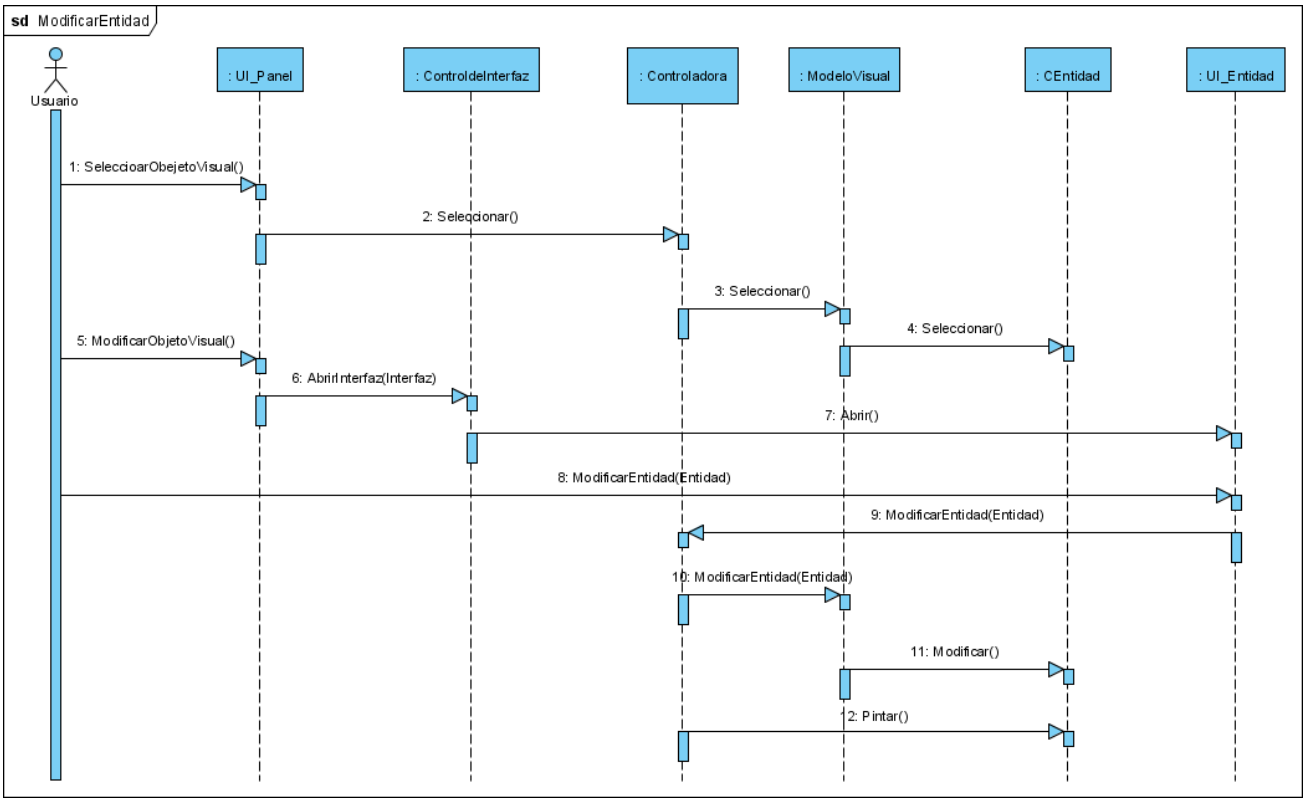
EXPORTAR MODELO A XML



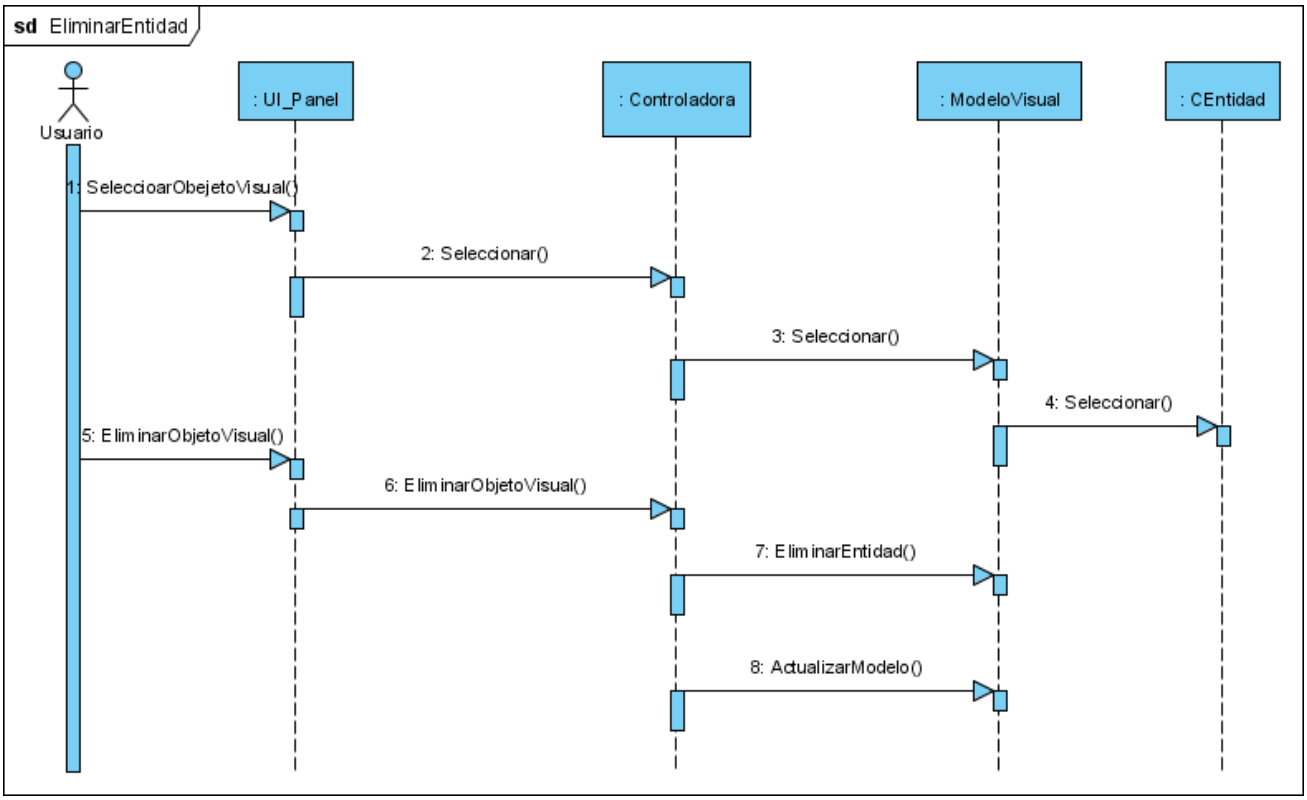
CREAR ENTIDAD



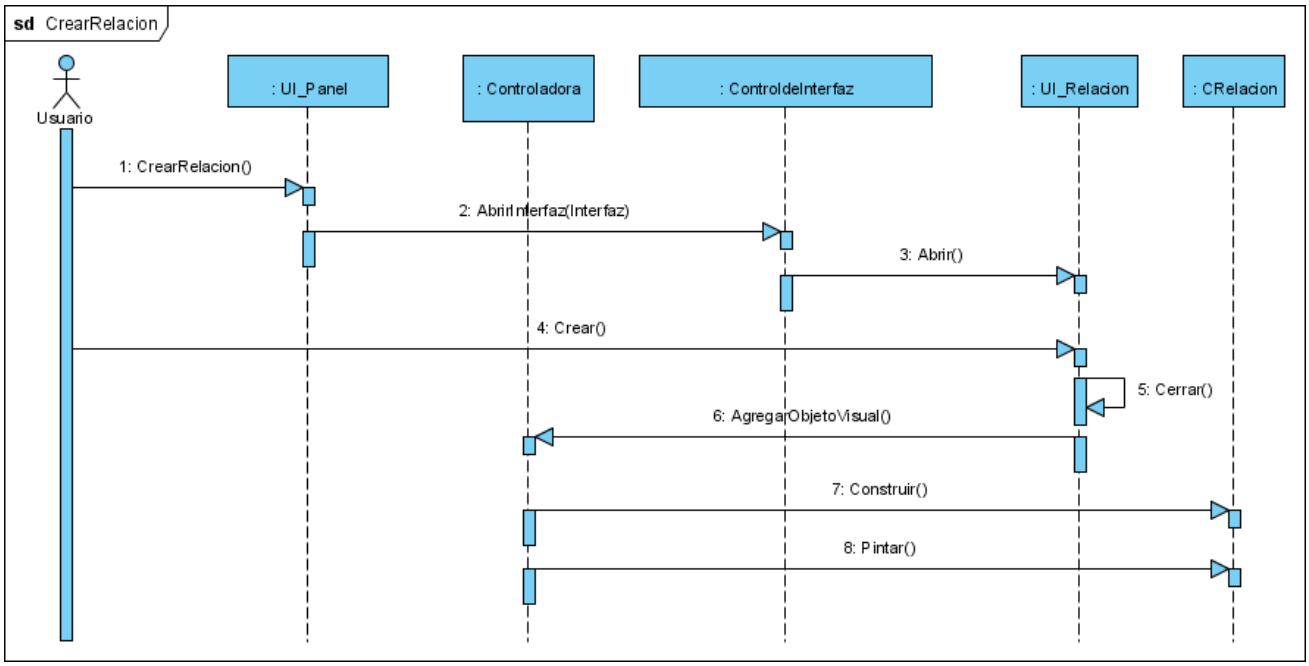
MODIFICAR ENTIDAD



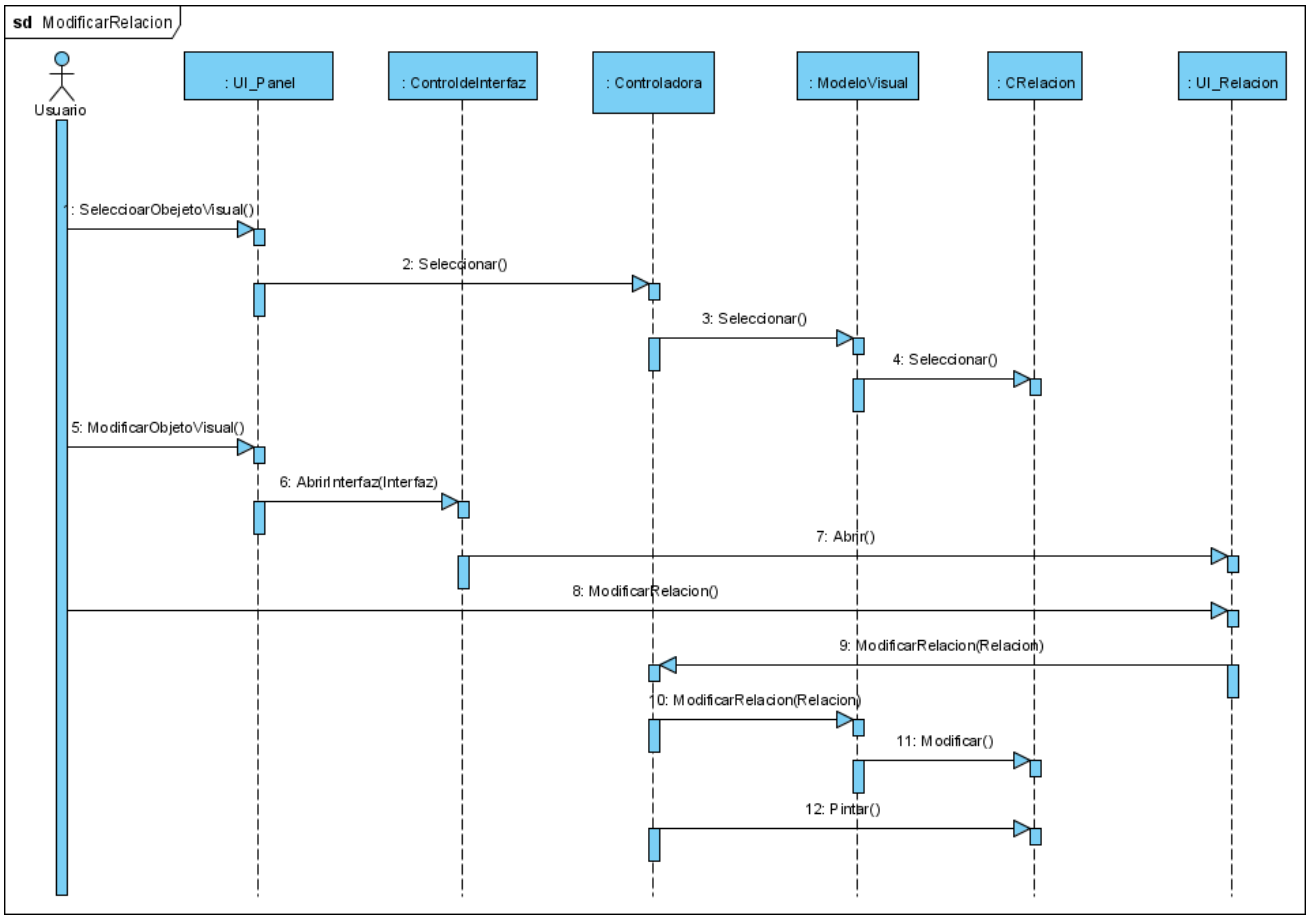
ELIMINAR ENTIDAD



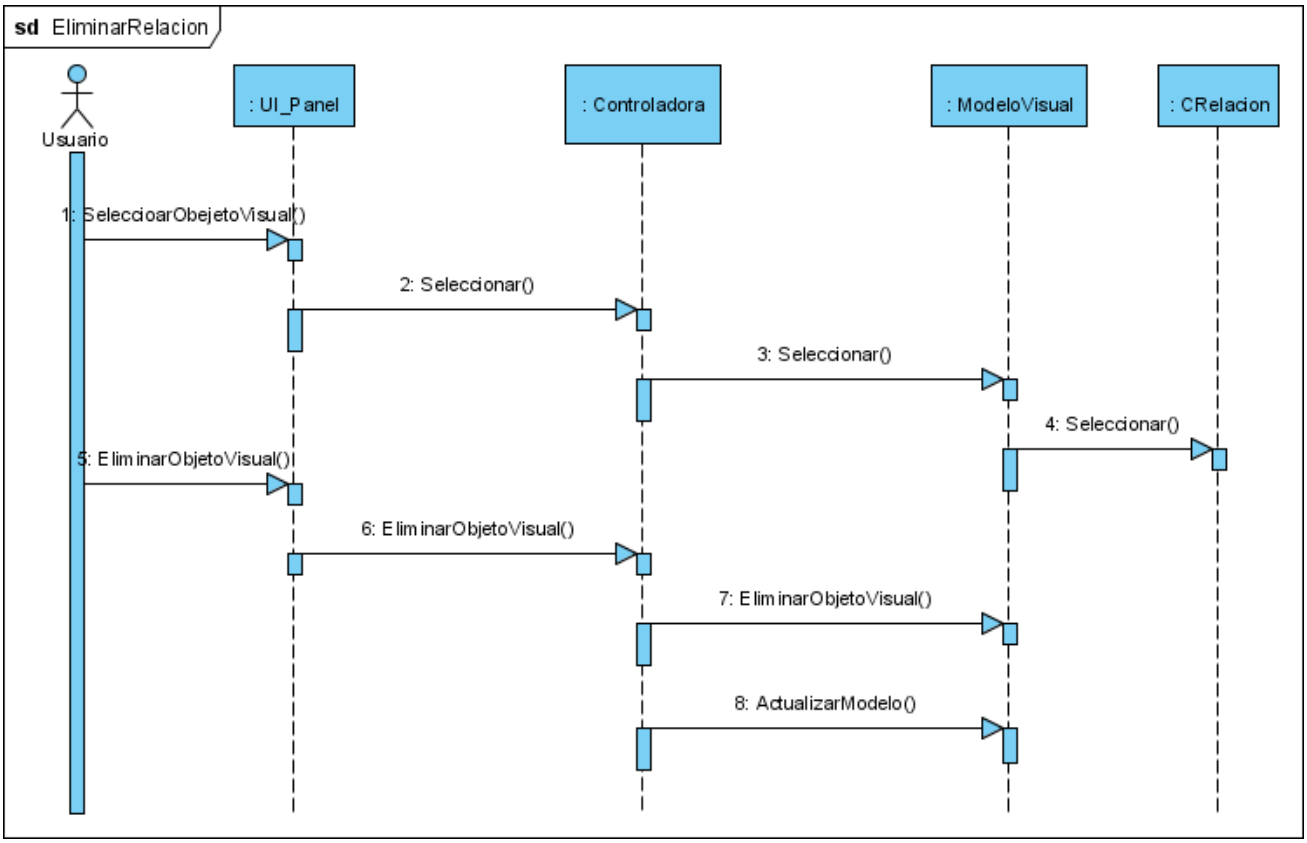
CREAR RELACION



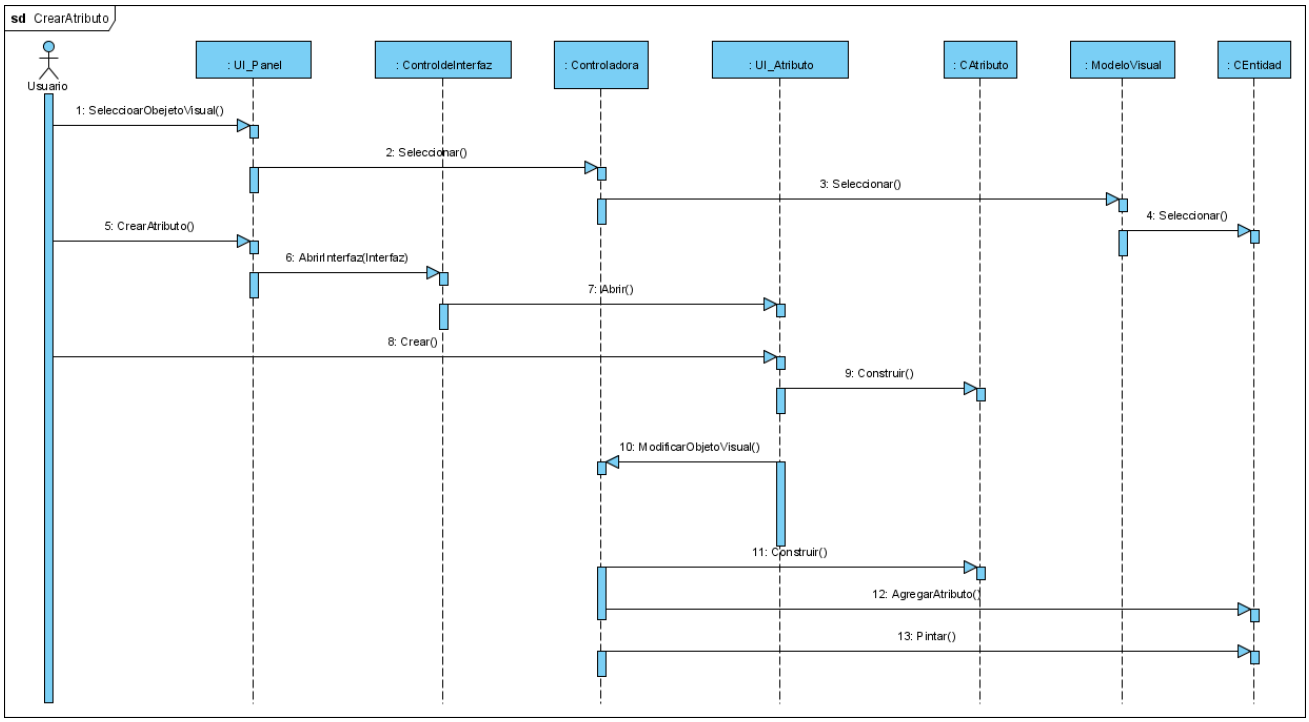
MODIFICAR RELACION



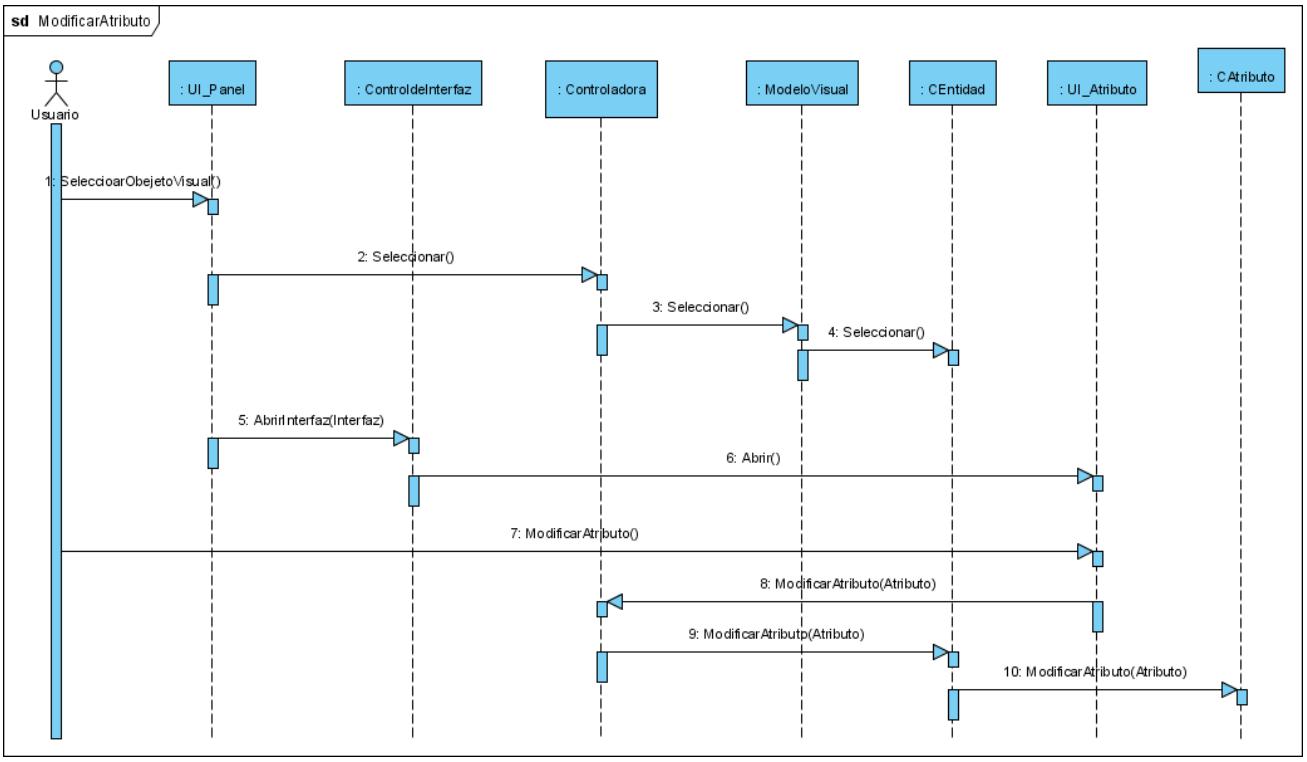
ELIMINAR RELACION



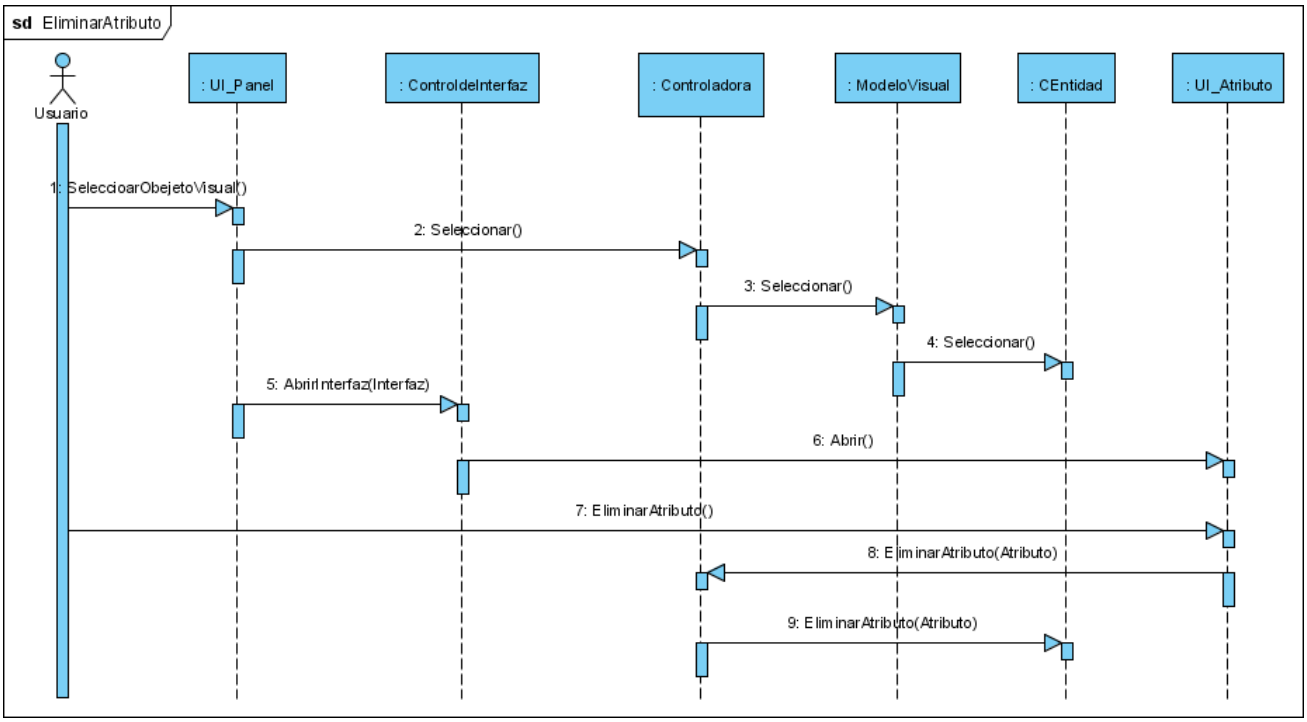
CREAR ATRIBUTO



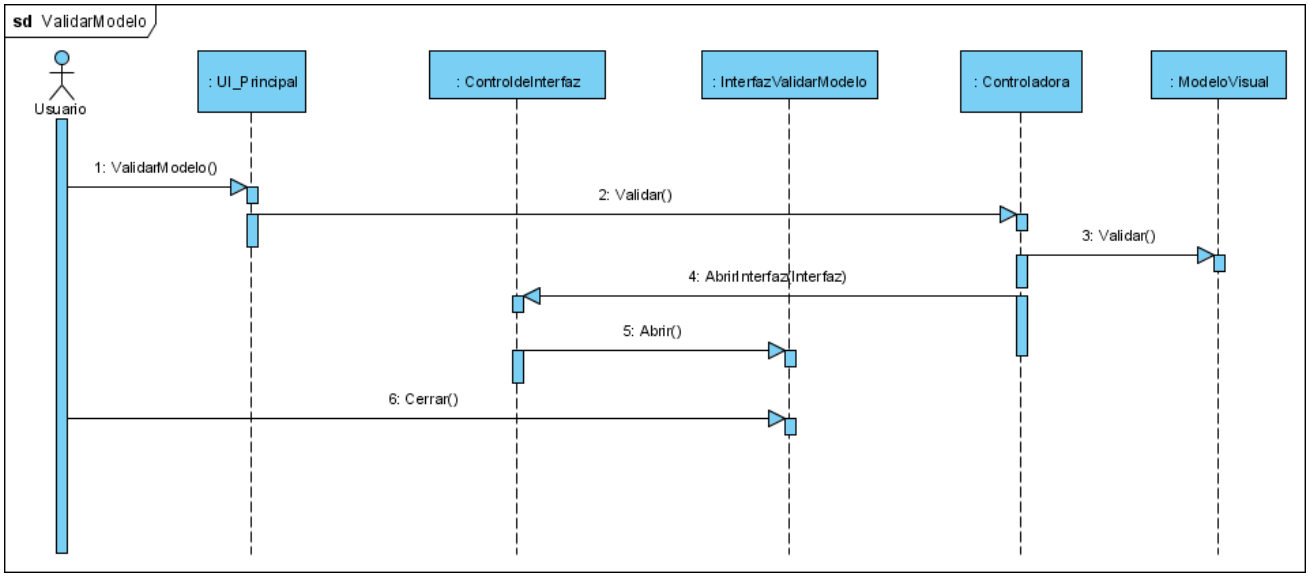
MODIFICAR ATRIBUTO



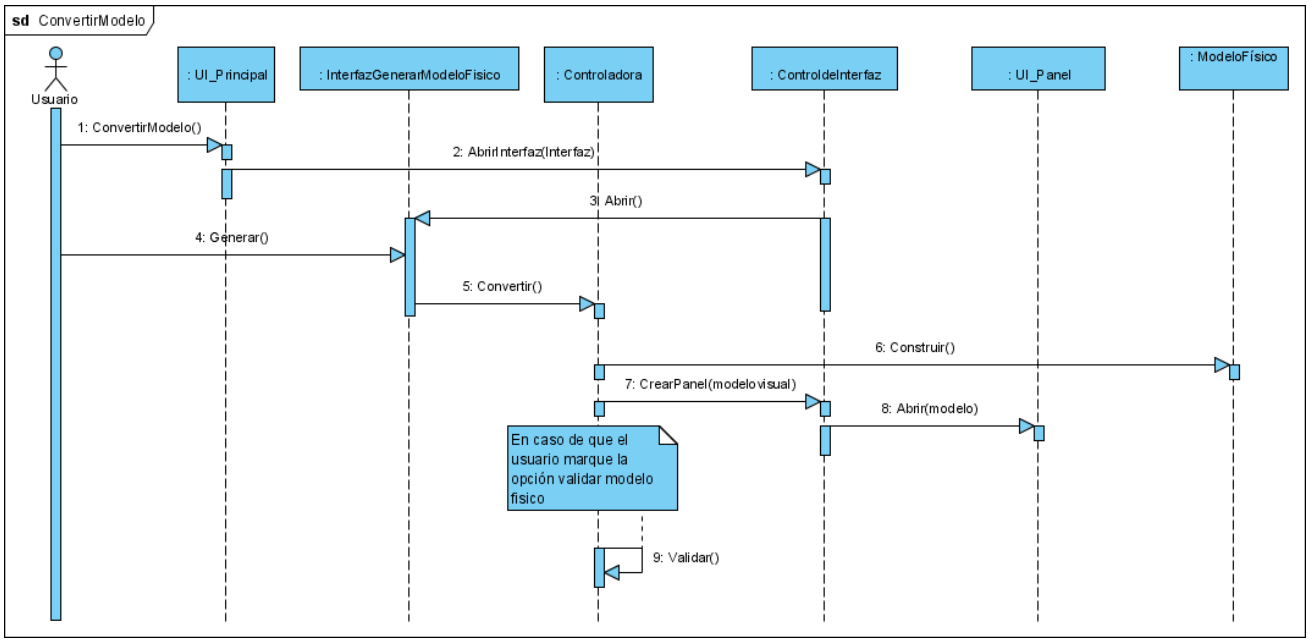
ELIMINAR ATRIBUTO



VALIDAR MODELO



CONVERTIR A MODELO FISICO



GLOSARIO DE SIGLAS Y TERMINOS

CASE: Acrónimo ingles de Computer Aided Software Engineering, que significa Ingeniería de Software Asistida por Ordenador.

UML: Lenguaje Unificado de Modelado (UML, por sus siglas en inglés, Unified Modelling Language) es el lenguaje de modelado de sistemas de software más conocido en la actualidad.

OPEN SOURCE: (Código fuente abierto) Un programa que ofrece al usuario la posibilidad de entrar en su interior para poder estudiarlo o modificarlo, libre acceso al código fuente, que va unido a una serie de conceptos.

RELEASE: (En español, "revisión" o "versión"). Es habitual que una aplicación software sufra modificaciones, mejoras o correcciones. El número de versión suele indicar el avance de los cambios.

DBMS: Sistema de Manejo de Bases de Datos Múltiples.

VP: Visual Paradigm.

XML: Extensible Markup Language (Lenguaje extensible de etiquetas) Es un meta-lenguaje que permite definir lenguajes de marcado adecuado a usos determinados. Se propone como lenguaje de bajo nivel (a nivel de aplicación, no de programación) para intercambio de información estructurada entre diferentes plataformas.

SINGLETONS: (Instancia única). Es un patrón de diseño clásico que dicta el comportamiento de que la propia clase es la responsable de controlar la existencia de una única instancia.

SCRIPTS: Un conjunto de comandos escritos en un lenguaje interpretado para automatizar ciertas tareas de aplicación.

UML: Lenguaje de Modelado Unificado.

PLUGINS: Es un programa adicional que puede ser añadido a una aplicación para aumentar la funcionalidad de esta.

IDE: Son las siglas de Integrated development environment, es decir, un entorno integrado de desarrollo.

ECLIPSE: Es una plataforma de desarrollo de código abierto basada en Java.

INTERFAZ: Parte del programa informático que permite el flujo de información entre el programa y el usuario u otro sistema.

RUP: Rational Unified Process. Proceso Unificado de Desarrollo.

CASOS DE USO: Un casos de uso es una secuencia de transacciones que son desarrolladas por un sistema en respuesta a un evento que inicia un actor sobre el propio sistema. Los diagramas de casos

de uso sirven para especificar la funcionalidad y el comportamiento de un sistema mediante su interacción con los usuarios y/o otros sistemas.

SQL: Server Query Language. Lenguaje de consulta que permite la recuperación de información desde un servidor de bases de datos.

JAVA: Lenguaje de programación desarrollado por Sun para la elaboración de pequeñas aplicaciones exportables a la red (applets) y capaces de operar sobre cualquier plataforma.

UNIX: Sistema operativo interactivo y de tiempo compartido creado en 1969 por Ken Thompson. Reescrito a mitad de la década de los '70 por ATT alcanzó enorme popularidad en los ambientes académicos, y más tarde en los empresariales, como un sistema abierto, robusto, flexible y portable, muy utilizado en los entornos Internet.

TRIGGER: Es un evento que se ejecuta cuando se cumple una condición establecida al realizar una operación de inserción (INSERT), actualización (UPDATE) o borrado (DELETE).

BASES DE DATOS: Es un conjunto integrado de datos junto con una serie de aplicaciones para su manejo accesibles simultáneamente por diferentes usuarios y programas.

SOFTWARE: Se llama así a todos los programas o elementos lógicos que hacen que una computadora funcione, poniéndose en interacción con los componentes físicos de la computadora.

MVC: (Model-View-Controller) Modelo Vista Controlador.

CLASES: En programación orientada a objetos, un tipo de datos definido por el usuario que especifica un conjunto de objetos que comparten las mismas características. Un miembro de la clase (objeto) es un "ejemplo" o caso de la clase. Las clases concretas están diseñadas para citar como ejemplos, mientras que las clases abstractas, para pasar las características por herencia.