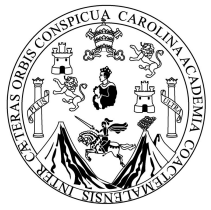


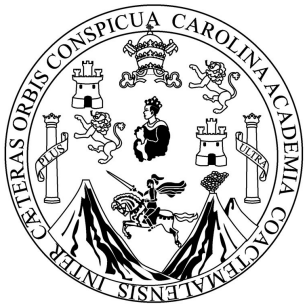
Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería en Ciencias y Sistemas

ANÁLISIS DE SISTEMAS CON HERRAMIENTAS CASE DE MODELADO CON UML

Dany Josué Mínchez Monzón
Asesorado por: Inga. Floriza Avila Pesquera

Guatemala, junio de 2007

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA



FACULTAD DE INGENIERÍA

ANÁLISIS DE SISTEMAS CON HERRAMIENTAS CASE DE MODELADO CON UML

TRABAJO DE GRADUACIÓN

PRESENTADO A LA JUNTA DIRECTIVA DE LA
FACULTAD DE INGENIERÍA
POR

DANY JOSUÉ MÍNCHEZ MONZÓN
ASESORADO POR: INGA. FLORIZA AVILA PESQUERA

AL CONFERÍRSELE EL TÍTULO DE

INGENIERO EN CIENCIAS Y SISTEMAS

GUATEMALA, JUNIO DE 2007

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
FACULTAD DE INGENIERÍA

NÓMINA DE LA JUNTA DIRECTIVA

DECADO:	Ing. Murphy Olympo Paiz Recinos
VOCAL I:	Inga. Glenda Patricia García Soria
VOCAL II:	Inga. Alba Maritza Guerrero de López
VOCAL III:	Ing. Miguel Ángel Dávila Calderón
VOCAL IV:	Br. Kenneth Issur Estrada Ruiz
VOCAL V:	Br. Elisa Yazminda Vides Leiva
SECRETARIA:	Inga. Marcia Ivónne Véliz Vargas

TRIBUNAL QUE PRACTICÓ EL EXAMEN GENERAL PRIVADO

DECANO:	Ing. Murphy Olympo Paiz Recinos
EXAMINADOR:	Ing. Marlon Antonio Pérez Turk
EXAMINADOR:	Ing. Edgar Estuardo Santos Sutuj
EXAMINADOR:	Ing. César Fernández Cáceres
SECRETARIA:	Inga. Marcia Ivónne Véliz Vargas

HONORABLE TRIBUNAL EXAMINADOR

Cumpliendo con los preceptos que establece la ley de la Universidad de San Carlos de Guatemala, presento a su consideración mi trabajo de graduación titulado:

ANÁLISIS DE SISTEMAS CON HERRAMIENTAS CASE DE MODELADO CON UML,

tema que me fuera asignado por la Dirección de la Escuela de Ingeniería en Ciencias y Sistemas, con fecha febrero de 2006.

Dany Josué Mínchez Monzón

Guatemala, Abril de 2007

Ingeniero
Carlos Alfredo Azurdia Morales
Coordinador de Privados y Revisión de Tesis
Escuela de Ciencia y Sistemas

Estimado Ingeniero.

Por medio de la presente, me permito informarle que he asesorado el trabajo de graduación titulado: **ANALISIS DE SISTEMAS CON HERRAMIENTAS CASE DE MODELADO CON UML,** elaborado por el estudiante Dany Josué Mínchez Monzón, a mi juicio el mismo cumple con los objetivos propuestos para su desarrollo.

Agradeciéndole de antemano la atención que le preste a la presente, me suscribo de usted,

Atentamente,

Floriza Avila Pesquera
Ingeniera en Ciencia y Sistemas
Asesor

Universidad San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería en Ciencias y Sistemas

Guatemala, 25 de Abril de 2007

Ingeniero
Marlon Antonio Pérez Turk
Director de la Escuela de Ingeniería
En Ciencias y Sistemas

Respetable Ingeniero Pérez:

Por este medio hago de su conocimiento que he revisado el trabajo de graduación del estudiante **DANY JOSUE MINCHEZ MONZON**, titulado: **"ANALISIS DE SISTEMAS CON HERRAMIENTAS CASE DE MODELADO CON UML"**, y a mi criterio el mismo cumple con los objetivos propuestos para su desarrollo, según el protocolo.

Al agradecer su atención a la presente, aprovecho la oportunidad para suscribirme,

Atentamente,

Ing. Carlos Alfredo Azurdia
Coordinador de Privados
y Revisión de Trabajos de Graduación

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA - INGENIERIA EN CIENCIAS Y SISTEMAS - COORDINACION TESIS

ESCUELA DE CIENCIAS Y SISTEMAS

UNIVERSIDAD DE SAN CARLOS
DE GUATEMALA

FACULTAD DE INGENIERÌA
ESCUELA DE CIENCIAS Y SISTEMAS
TEL: 24767644

"Todo por ti Carolingia mía"
Dr. Carlos Martínez Durán
2006: Centenario de su nacimiento

El Director de la Escuela de Ingeniería en Ciencias y Sistemas de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, luego de conocer el dictamen del asesor con el visto bueno del revisor y del Licenciado en Letras, de trabajo de graduación titulado ***"ANALISIS DE SISTEMAS CON HERRAMIENTAS CASE DE MODELADO CON UML"****, presentado por el estudiante* ***DANY JOSUE MINCHEZ MONZON,*** *aprueba el presente trabajo y solicita la autorización del mismo.*

"ID Y ENSEÑAD A TODOS"

Ing. Marlon Antonio Pèrez Turk
Director, Escuela de Ingeniería en Ciencias y Sistemas

Guatemala, 04 de junio 2007

Universidad de San Carlos
de Guatemala

Facultad de Ingeniería
Decanato

Ref. DTG.175.07

El Decano de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, luego de conocer la aprobación por parte del Director de la Escuela de Ingeniería en Ciencias y Sistemas, al trabajo de graduación titulado: **ANÁLISIS DE SISTEMAS CON HERRAMIENTAS CASE DE MODELADO CON UML,** presentado por el estudiante universitario **Dany Josué Mínchez Monzón,** procede a la autorización para la impresión del mismo.

IMPRÍMASE.

Ing. Murphy Olympo Paiz Recinos
DECANO

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
DECANO
FACULTAD DE INGENIERIA

Guatemala, junio de 2007

/cc

AGRADECIMIENTOS A:

DIOS	Por guiarme, guardarme y llenarme de sabiduría para culminar satisfactoriamente esta carrera universitaria.
MI ESPOSA	Por ser ayuda idónea en cada momento.
HIJOS	Que han sido mi inspiración.
MIS PADRES	Por su constante apoyo y preocupación.
HERMANOS	Por regalarme siempre palabras de apoyo cuando he sentido desfallecer.
FAMILIARES	Con cariño sincero.
COMPAÑEROS Y AMIGOS	Por momentos compartidos y el apoyo incondicional.
FACULTAD DE INGENIERÍA	Por haberme brindado la oportunidad de forjarme del saber al estudiar esta carrera universitaria.
UNIVERSIDAD DE SAN CARLOS DE GUATEMALA	Por permitirme ser fruto del saber, al estudiar esta carrera universitaría.

ÍNDICE GENERAL

ÍNDICE DE ILUSTRACIONES

FIGURAS

TABLAS

RESUMEN

En la actualidad la tecnología avanza día con día y pone a disposición de los profesionales y estudiantes, de ingeniería en Ciencias y Sistemas, las herramientas para desarrollar él análisis y diseño de software con mayor exactitud, rapidez y calidad. Por tal motivo, el fin primordial de este trabajo es exponer el procedimiento del uso de una metodología de desarrollo de sistemas.

Las herramientas CASE de modelado con UML nos permite aplicar la metodología de análisis y diseño orientados a objetos y abstraernos del código fuente, en un nivel donde la arquiterctura y el diseño se tornan más obvios y más fáciles de entender y modificar. A su vez la metodología a exponer, es una potente herramienta que genera código fuente reutilizable, entendible y de fácil modificación. Además, se puede incluir código personalizado al estar generando nuestro proyecto, al mismo tiempo un documentador que puede ser accesado por medio de una intranet corporativa. Tambien nos permite la representación del mundo real, ya que cada objeto tiene identidad propia. Entre las ventajas más importantes que se podria mensionar está el tiempo y costo mucho menor de desarrollo, con respecto a la programacion tradicional.

OBJETIVOS

General

Motivar al estudiante y al profesional de ingeniería en Ciencias y sistemas, para que use la tecnología como herramienta en el análisis y diseño de proyectos, ahorrando tiempo y recursos.

Específicos

1. Presentar una guía útil para el uso de la metodología en el análisis y diseño, documentación e implementación de software.

2. Utilizar la metodología en el análisis y diseño de un proyecto de facturación.

INTRODUCCIÓN

Este trabajo se encuentra dividido en varios capítulos. Ésto le ayudará a comprender los aspectos necesarios, para la utilización de esta metodología. Los capitulos, son:

Capítulo uno - Introducción a herramientas case: aquí se pretende que el lector mire ventajas y desventajas de diferentes herramientas CASE, seleccionando la que mejor responda a sus expectativas.

Capítulo dos - Introducción a la metodología: en este capítulo se indican las características y fases que componen la metodología a utilizar.

Capítulo tres - Casos de uso: contiene conceptos básicos, relaciones y priorización.

Capítulo cuatro - Modelo de objetos: se expone conceptos básicos, relaciones entre entidades, objetos y componentes.

Capítulo cinco - Implementación del proyecto utilizando la metodología: Éste último capítulo está reservado a la implementación de la metodología. La manera de llevar ésto a cabo es asi: se debe tener conocimiento previo en el manejo de herramientas con las que se comunica la metodología, y así poder tener un mayor aprovechamiento del recurso. Así también se indica cómo generar formularios, código fuente y configuración del documentador. Además, indica ventajas y desventajas de la utilización de una herramienta asistida por computadora vrs. programación tradicional.

También se incluye un índice de ilustraciones para facilitar la búsqueda figuras y tablas.

Se espera que este trabajo sea de utilidad y que contribuya al desarrollo del país, en este campo tan dinámico, como lo es la informática.

1 INTRODUCCIÓN A HERRAMIENTAS CASE

1.3 Herramientas CASE

CASE en sus siglas en ingles computer-assisted software engineering (ingeniería de software asistido por computadora, es una herramienta que ayuda al ingeniero de software a desarrollar y mantener software.

CASE son herramientas individuales para ayudar al desarrollador de software o administrador de proyecto durante una o más fases del desarrollo de software (o mantenimiento.) Un CASE es una combinación de herramientas de software y metodologías de desarrollo.

1.3.1 Beneficios

Entre los beneficios de CASE se pueden mencionar:

a. Disminuye tiempo y automatiza tareas
b. Generación (código, formas, reportes, esquemas, documentación)
c. Estandarización y administración visual de la aplicación
d. Facilidad de mantenimiento y reutilización de componentes
e. Facilidad aplicación técnica de desarrollo
f. Portabilidad de aplicaciones
g. Mejora calidad del sistema

1.3.2 Componentes

a. Diccionario o repositorio

Aquí se almacenan los elementos definidos o creados por la herramienta en cuya gestión se realiza mediante el apoyo de un sistema de base de datos o gestión de archivos.

b. Meta-modelo

Constituye el marco para la definición de las técnicas y metodologías que soportan la herramienta, por lo general es extensible por el usuario.

c. Carga y descarga de datos

Lo conforman las facilidades que permiten cargar el repositorio de la herramienta CASE con datos provenientes de otros sistemas o bien generar a partir de la propia herramienta que puedan a su vez alimentar otros sistemas.

d. Comprobación de errores

Permiten realizar un análisis de la exactitud, integridad y consistencia de los esquemas generados por la herramienta.

e. Interfase con el usuario

Consta de editores de texto y herramientas de diseño grafico que permiten mediante la utilización de un sistema de ventanas menús y con la ayuda del mouse, definir los diagramas que incluyen las distintas metodologías.

1.4 Ingeniería de *Software* y herramientas CASE

La evolución de las herramientas CASE está ligada a la evolución de la ingeniería de software como disciplina.

El ciclo de vida del software es entendido como la secuencia de fases por las cuales atraviesa un proyecto de desarrollo de software desde su concepción hasta el fin del uso del producto de software obtenido, pasando por su construcción y mantenimiento.

Una metodología se define como el conjunto de métodos que se siguen en una investigación científica o en una exposición doctrinal. Una metodología de *software* usualmente identifica las principales actividades, por ejemplo, análisis, diseño, codificación y pruebas a ser realizadas e indica qué personas (usuarios, administradores, técnicos) deben estar involucrados en cada actividad y qué papel desempeña en ellas. Las metodologías a menudo describen criterios de entrada, por ejemplo, condiciones para comenzar una fase, criterios de salida y puntos de revisión.

Un método se define como un procedimiento que se sigue en las ciencias para hallar la verdad y enseñarla. Un método es entendido como un enfoque técnico para ser utilizado en toda o parte de una metodología. Unas de las actividades típicas incluidas en el ciclo de vida del software son: planificación del proyecto, gestión del proyecto, análisis, diseño, codificación, pruebas, documentación, mantenimiento, validación y verificación.

1.4.1 Evolución de las herramientas CASE

Las primeras herramientas para apoyar el proceso de desarrollo de software fueron los editores y procesadores de texto, los cuales fueron usados para escribir programas y su documentación. Así también algunos programas de dibujo comenzaron a incorporar las notaciones gráficas de técnicas para diseño de programas.

La consolidación de metodologías se desarrolló integrando diferentes técnicas e impulsó la aparición de paquetes de propósito más amplio. Surgió la necesidad de un diccionario de datos del sistema que almacene las definiciones usadas en las diferentes fases del desarrollo (este diccionario es lo que comúnmente se denomina repositorio), esto contribuyó a implementar funciones de integración y verificación de consistencia entre técnicas asociadas a distintas actividades en el desarrollo. Actualmente, todos los desafios y los correspondientes enfoques de solución están normalmente concebidos y practicados dentro del contexto de CASE.

1.4.2 Expectativas del uso de una CASE

El propósito de una herramienta CASE es dar soporte automatizado para la aplicación de todas o algunas técnicas usadas por una o varias metodologías.

Entre las expectativas se mencionan:

a. Enfrentar el proceso de desarrollo de software como un proyecto de ingeniería de software.
b. Usar una herramienta CASE para apoyar la aplicación de los métodos utilizados.
c. Aplicar una o varios métodos de forma integrada cubriendo todas las actividades del ciclo del software.

1.4.3 Clasificación de herramientas CASE

a. De Gestión

Encargada de la planificación, estimación y del seguimiento del proyecto.

b. Técnicas

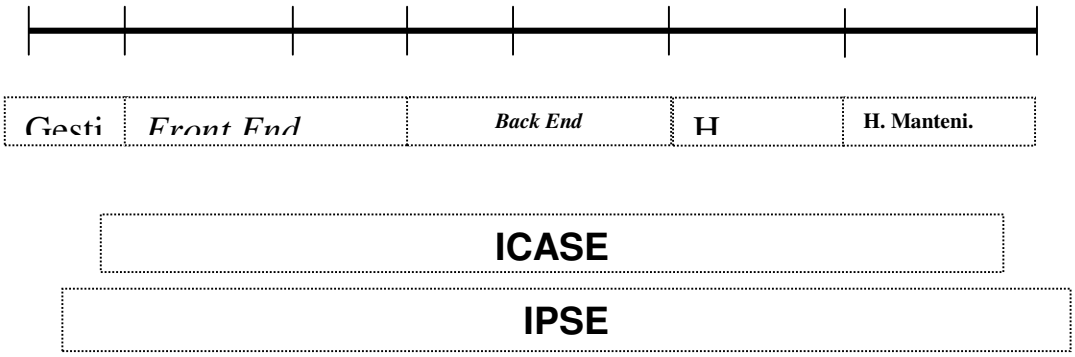
- *Front end*: abarca las primeras fases del ciclo del software (determinación de requerimientos, análisis y diseño)
- *Back end*: fases finales del ciclo del *software* (diseño detallado, generación de código)
- Otras: de prueba y mantenimiento.
- Soporte: Incluye repositorio, control, configuración y seguridad.

1.4.4 Términos para clasificar herramientas CASE

a. ICASE: Case integral, *integran Front* y *back end*. Esta incluye toda la etapa de desarrollo.
b. IPSE: Ambiente de soporte programa integrado, tiene a todas las categorías.

1.4.5 Etapa de desarrollo

Inv. Gestión Preliminar Análisis Diseño Construcción Implementación Mantenimiento



1.4.6 Herramientas CASE y sus características

1.4.6.1 Herramientas de gestión

a. AXIOMSYS

Axiomsys es una herramienta CASE que combina los métodos de análisis estructurado con las extensiones de real-time y arquitectura que planean para proporcionar una solución completa a los problemas de los sistemas a planear.

1.4.6.2 Herramientas técnicas

1.4.6.2.1 *Front End*

a. MULTIBASE-EASYCASE

EasyCASE es una herramienta CASE que cubre las fases de análisis y diseño. (diagrama de flujo de datos, modelo entidad relación, diagrama de estructuras, historia de la vida de la entidad.)

b. VISIBLE ANALIST/INGENIERIA DB

Es versión de visible analyst se dirige a administradores de bases de datos, modeladores de datos , arquitectos de datos y analistas de datos. Esta especializada con las capacidades de visible analyst en modelado de datos e ingeniería de base de datos y contiene únicamente los métodos, técnicas, diagramas y funcionalidades relacionadas directamente con modelado de datos y diseño de base de datos. Compite favorablemente con otras herramientas especializadas en diagramas entidad relación (ERD) y diseño de base de datos.

1.4.6.2.2 *Back end*

a. PLATINUM ERWIN

Es una poderosa herramienta fácil de usar, para el diseño de base de datos. Con él, aumentará la productividad en el diseño y mantenimiento de bases de datos de gran calidad.

Tiene un nuevo soporte para el establecimiento de estándares de diseño y reutilización, propiedades meta expandibles y modelado dimensional para el diseño modelado de data warehouse. Además de otras aplicaciones basadas en base de datos relacionales en las cuales se requiere un diseño iterativo e inteligente entre sus niveles físicos y lógicos

b. WINA&D *DESKTOP*

Es un software que diseña herramienta OO, análisis estructurado y plan o datos los proyectos modelados. Apoya al mismo proceso, datos y modelos de la clase como la edición normal, la ventaja agrega al estado, objeto y modelos de la estructura. Los rasgos adicionales en la edición de desktop incluyen, generación de código, plantillas del documento, readonly y vuelve a abrir órdenes para el acceso del documento flexible, construir en archivos del silbido para los documentos del proyecto, los gráficos de herencia instante, costumbre HTML informa y una aplicación de programa.

1.4.6.3 Herramientas de soporte

a. EXCELERATOR

Funciones más importantes de excelerator: gráficas, xldiccionario, plantas y reportes, documentación, análisis, interfaces y utilerias. Excelerator incluye un sistema de contraseñas para impedir el acceso a la información por usuarios que no tienen autorización para hacerlo. El acceso se controla en el ámbito de cada proyecto.

b. VERSIÓN 7.6 DE *VISIBLE ANALYST CORPORATE EDITION*

- Una barra de control con funciones añadidas par modificar más fácilmente las características de los diagramas.
- Un interfase con el repositorio completamente rediseñado que utiliza diálogo de manera mucho más eficaz.
- Generación de los diagramas en formato JPE y publicación de informes a través de internet mediante formato html.
- Actualizada la generación de SQL para consultas.
- Generación de vistas para informix, oracle, DB2.

1.4.7 Herramientas ICASE

a. eZway

Es una herramienta fácil de usar que ayuda en la planeación enlazando Microsoft proyect, análisis enlazando visio (basado en UML), diseño generando formas, integrando menús, generando código en visual basic y /o ASP, la documentación técnica generada en paginas HTML, que pueden ser publicadas en una Intranet corporativa, facilita también el establecimiento de estándares en el proceso de desarrollo y documentación, facilita el mantenimiento del sistema y la actualización de su documentación, facilita la planificación y gestión del proyecto informático.

b. MULTIWAY

Esta es una herramienta abierta, con esta puede desarrollar sobre cualquiera de las bases de datos más extendidas del mercado (Oracle, informix, ingres, DB2/6000.) Casi todo el desarrollo del ciclo de vida del software la realiza esta herramienta, por eso la hace una de las herramientas más importantes dentro del mercado informatico.

c. EAY CASE PROFESIONAR

Con esta herramienta se pueden realizar los distintos diagramas de flujo de datos y entidad relación sus funciones principales son:

- Introducción al desarrollo de aplicaciones informáticas
- Análisis de requisitos y diseño de módulos
- Diseño de interfases de usuario y diseño físico de datos
- Descripción de programas y gestión de desarrollo e implantación de aplicaciones

1.4.8 Herramientas IPSE

a. *ORACLE DESIGNER*

Esta herramienta que tiene todos los módulos de las herramientas CASE integrado con los que podrá realizar:

- Elaborar programas que cumplan las especificaciones establecidas en el diseño con bajo coste de mantenimiento.
- Integrar y enlazar programas y rutinas siguiendo las especificaciones establecidas en el diseño.
- Realizar pruebas funcionales de programas atendiendo a las especificaciones establecidas en el diseño.

- Elaborar y mantener documentación descriptiva de programas y pruebas que permitan la consulta y actualizaciones por terceras personas.
- Efectuar cambios en programas de a acuerdo con los nuevos requerimientos establecidos.

b. VISIBLE *ANALYST* CORPORATIVA

Esta versión es la más poderosa de todas las versiones de visible analyst, incluye la totalidad de las características mencionadas en las versiones anteriores. Es la herramienta indicada para amplios proyectos con múltiples grupos de usuarios, con avanzada ingeniería de base de datos, controles para el acceso multiusuario y transición a métodos orientados objetos.

2 INTRODUCCIÓN A LA METODOLOGÍA UTILIZADA

3.1 Qué es eZway

eZway es una herramienta que viene a resolver los retos que enfrentan las empresas en el desarrollo de *software* tales como recursos limitados, la necesidad de reaccionar y evolucionar ante los cambios, así como lograr la incorporación de las más nuevas tecnologías manteniendo la integración con los sistemas existentes en un negocio.

Dicha herramienta implementa una metodología que guía al desarrollador por un proceso interactivo e incremental, con base en la notación del lenguaje modelado unificado, (UML -por sus siglas en inglés-) para el análisis y diseño de aplicaciones en ambientes distribuidos de múltiples capas, que a partir de una base del conocimiento que refleja la realidad, genera un código de programación en múltiples capas, así como toda la documentación técnica asociada. Esto hace que se reduzca el tiempo de desarrollo y mantenimiento de aplicaciones.

La herramienta eZway realiza la administración de proyectos complejos en equipos de desarrollo, así como una adecuada gestión del riesgo del producto que se va a desarrollar. Además habilita rápidamente a las empresas para Internet sin duplicar esfuerzos.

3.1 Características de eZway

2..1 Ser iterativo e incremental

Esto significa que no existe una secuencia de fases rígida, en la que para iniciar una debe completarse la anterior.

La idea es hacer un poco de exploración, un poco de planificación, un poco de construcción y, si se obtiene una versión funcional, proceder a su instalación, luego el ciclo se repite. El ciclo se repetirá tantas veces como versiones se hayan previsto en el plan de versiones del producto a desarrollar.

2..2 Orientarse con base en los casos de uso

Esto quiere decir que el ciclo iterativo e incremental, está determinado por el desarrollo de los casos de uso. Los casos de uso se priorizan de acuerdo con varios criterios, y luego se procede a un desarrollo paulatino. Al no existir mas casos de uso, se da por terminado el desarrollo del sistema.

2..3 Centrarse en la gestión de riesgos

Esto significa que el líder del proyecto identifica, controla y reduce al máximo los riesgos en cada ciclo del desarrollo, de tal manera que al finalizar cada ciclo se habrán reducido los riesgos del proyecto y se deberá evaluar la situación para encontrar nuevos riesgos.

2..4 Sustentado en la arquitectura

Durante el desarrollo de un nuevo sistema, se escoge una arquitectura idónea para modelar tanto lógica como físicamente el producto, esta arquitectura, organiza la conceptualización del sistema a distintos niveles de abstracción. eZway utiliza la arquitectura en tres capas como modelo arquitectónico principal.

3.1 Fases de la metodología utilizada

Las fases de la metodología utilizada están orientadas al análisis, diseño, construcción e implementación de aplicaciones de software, las cuales son:

- Fase de exploración
- Fase de planificación y análisis de riesgos
- Fase de construcción
- Fase de instalación

2..1 Fase de exploración

El objetivo principal de la fase consiste en recolectar la información necesaria para poder hacer un plan de proyecto suficientemente convincente. Para tal efecto, se debe producir un modelo lógico preliminar e incompleto, con el fin de facilitar la identificación y priorización de riesgos, priorización de casos de uso, así como una estimación de esfuerzo y recursos necesarios que deberá presentarse por medio de un cronograma basado en ciclos y una distribución general de responsabilidades.

Esta versión preliminar del modelo lógico debería incluir:

a. Una lista de funciones (y restricciones) a dos niveles de detalle, es decir, un nivel general y un nivel más específico en que cada función se descompone en funciones más detalladas
b. Uno o dos diagramas con los principales casos de uso
c. Una especificación general de estos casos
d. Un modelo de objetos con los objetos más importantes o significativos para comprender el dominio de aplicación, algunos de sus atributos y relaciones más características.

Finalmente es importante señalar que esta fase se debe repetir cada vez que se desee desarrollar una nueva versión del producto, lo cual es consistente con el modelo de ciclo de desarrollo en espiral. En consecuencia, tanto el modelo lógico producido en esta fase, como el propio plan de proyecto evolucionarán durante el desarrollo mismo del producto, no deben nunca considerarse documentos terminados.

2..2 Fase de planificación y análisis de riesgos

El único objetivo de esta fase consiste en que el futuro líder de proyecto, o alguna otra persona con suficiente experiencia previa en la dirección de proyectos, elabore un plan de proyecto. Para esto deberá basarse en el modelo lógico preliminar construido en la fase anterior de exploración.

La elaboración de un plan para cada proyecto nuevo, permitirá garantizar un uso óptimo de los recursos disponibles en la empresa, una negociación más atinada sobre los plazos de entrega con los clientes, una visualización a mediano y largo plazo del desarrollo del producto, y un proceso de aprendizaje organizacional, que en el mediano plazo, mostrará sus frutos con una capacidad cada vez mayor de predicción sobre el esfuerzo y los recursos necesarios para el desarrollo de un proyecto. También mejorará paulatinamente la gestión de riesgos en cada proyecto.

La estimación de esfuerzo es fundamental para poder hacer una buena planificación. La estimación es difícil, requiere sobre todo de experiencia pero también del conocimiento del dominio de la aplicación y de la tecnología que se utilizará en la solución.

La construcción de casos de uso, acompañados del diseño preliminar de la interfaz humano sistema (IHS) puede ser un recurso muy valioso a la hora de estimar, de hecho técnicas modernas como el cálculo de puntos de función se basa, en un diseño preliminar de la IHS, por lo menos para las principales funciones del sistema.

2..3 Fase de construcción

La fase de construcción también es un ciclo iterativo e incremental. No se debe pensar en que se trata de un proceso lineal, dónde se hace todo el análisis, el diseño, la programación y prueba. Más bien, la idea es hacer un poco de análisis, un poco de diseño, un poco de programación y un poco de prueba.

Durante una fase de construcción se pretende alcanzar una versión del producto, de acuerdo con el plan de versiones elaborado. En cada repetición se amplía la funcionalidad del producto, hasta que poco a poco se van alcanzando los requerimientos de la versión meta.

En cada repetición de la fase de construcción se trabaja con base en un conjunto de casos de uso. Los casos de uso han sido previamente priorizados. De acuerdo con esta priorización de los casos de uso, se escogen los más prioritarios para desarrollarlos en el siguiente ciclo de construcción. Cada ciclo de construcción termina cuando se han logrado programar los casos de uso correspondientes.

Asimismo, del análisis se produce un modelo lógico más completo. En cada repetición del diseño se produce un modelo físico más completo. Ninguno de estos documentos es definitivo, hasta que se logra una versión comercializable. Incluso en este caso, se le confiere a los modelos la característica de línea base. Esto significa que se constituyen en documentos estables que sirven de referencia para posibles actividades de reutilización, mantenimiento o reingeniería futuras.

La fase de construcción se subdivide en:

- Análisis
- Diseño
- Prueba
- Programación

2.3.3.1 Análisis

La actividad de análisis es la primera de cada ciclo de construcción. La meta en cada ciclo de construcción es desarrollar completamente alguna parte de la funcionalidad del sistema. Cada parte queda delimitada por un conjunto de casos de uso que han sido identificados y seleccionados.

Tomando como base la arquitectura en tres capas, el análisis se concentra en la construcción de la capa de objetos del negocio.

La actividad de análisis dentro de un ciclo de construcción consiste en:

- Especificar casos de uso, aquí se trata de listar y detallar los casos de usos a utilizar. Se podría agregarse el diseño de la interfaz con el usuario.
- Afinar diagramas de casos de uso, en alguno de los casos se podrá introducir nuevos casos de uso que extienden o usan a los existentes.
- Afinar el modelo de objetos, a partir del conjunto de casos de uso que se pretenden desarrollar se deberá: identificar nuevos objetos, identificar nuevas relaciones entre objetos, identificar nuevos atributos y métodos para los objetos del modelo.

2.3.3.2 Diseño

El objetivo de toda actividad de diseño es ampliar o detallar la arquitectura física para que concuerde con la arquitectura lógica que se tiene como insumo principal. Al final de un ciclo de construcción, la arquitectura física debe quedar lista para que la programación pueda desarrollarse de manera fluida.

Algunas actividades típicas de diseño son:

- Afinar el modelo de objetos y la especificación de objetos
- Empaquetar los objetos en componentes
- Diseño de la base de datos, es decir tablas, índices y relaciones
- Establecer las relaciones de dependencia entre componentes y partes de la base de datos
- Construir una arquitectura física análoga a la arquitectura lógica, empaquetar componentes y partes de la base de datos
- Detallar la Interfaz Humano-Sistema
- Especificar detalladamente cómo será la comunicación entre el sistema y cualquier otro sistema externo.

2.3.3.3 Programación

Si las actividades de análisis y diseño se han hecho con cuidado, la programación será un proceso relativamente fácil. Esto significa que habrán pocas o ninguna corrección qué hacer sobre los resultados del análisis y diseño a partir de la programación.

Actividades típicas de esta fase son:

- Codificación de los métodos de los objetos y cualquier otro tipo de funciones que sean necesarias.
- Introducción de nuevos atributos para representar relaciones entre objetos. Es posible que hayan algunos atributos que no sean importantes a nivel de análisis o diseño, pero al llegar a la programación son imprescindibles. Esto sucede con la representación de relaciones entre objetos, que en el modelo de objetos es representado sin importar cómo han de implementarse.
- La codificación debe seguir las reglas definidas en eZway para la nominación de variables, archivos, forms, tablas, así como la inserción de comentarios para la documentación interna del código.
- La inserción de código para el manejo de errores y excepciones (vistas como errores inesperados).
- Puede ser que sea necesario introducir nuevos objetos en la capa de base de datos.
- Es conveniente establecer un orden en la implementación de objetos y componentes de tal manera que se faciliten las pruebas modulares y de integración. Por esta razón conviene programar primero aquellos objetos y componentes que dependan menos de otros. Lo mismo se aplica a la construcción de tablas de la base de datos.

Es posible que la programación genere cambios en la arquitectura, lógica o física, pero estos podrán hacerse inmediamente o postergarse para el siguiente ciclo de construcción.

2.3.3.4 Prueba

Lo más importante en una actividad de prueba de software es la mentalidad de quien hace las pruebas. El objetivo debe ser encontrar errores y no demostrar que el software está correcto. Se debe partir del supuesto de que el *software* tiene errores, así que lo único correcto es esforzarse por encontrarlos lo más pronto posible y de la manera más eficiente. Por esta razón, en general se recomienda que la prueba de software sea ejecutada por personas distintas a aquéllas que lo produjeron.

Tipos de prueba que se pueden aplicar a un sistema:

2.3.3.4.4 Pruebas modulares

Se concentran en la búsqueda de errores en un objeto, una función o un grupo pequeño de objetos y/o funciones que están fuertemente interrelacionados. Existen básicamente dos grandes tipos de técnicas, las llamadas pruebas de caja negra que se basan en las especificaciones de un módulo y las técnicas de caja blanca que se basan en la estructura del código de un módulo.

2.3.3.4.5 Pruebas de integración

Se concentran en la búsqueda de errores asociados a la interacción de los componentes de un sistema o subsistema.

2.3.3.4.6 Pruebas funcionales

Se concentran en la búsqueda de errores en la implementación de la funcionalidad deseada para el sistema. Es posible que algún aspecto de la funcionalidad no se haya implementado o que sólo se haya implementado parcialmente. Incluso es posible que se haya implementado pero de una forma que no es satisfactoria para el cliente. Por lo que hay dos elementos básicos en las pruebas funcionales, la especificación de la funcionalidad que debió quedar plasmada en los casos de uso y la participación del cliente o usuario.

2..4 Fase de instalación

En el caso de sistemas críticos para el cliente, será necesario mantener el sistema actual y el nuevo en modo paralelo y funcionando por un tiempo, a fin de minimizar el riesgo de pérdidas para el cliente.

En el caso de sistemas a la medida, conviene elaborar un documento final que describa la distribución de componentes tal como quedó en la infraestructura del cliente, esto podría facilitar el mantenimiento futuro del sistema.

2.4 Introducción a UML

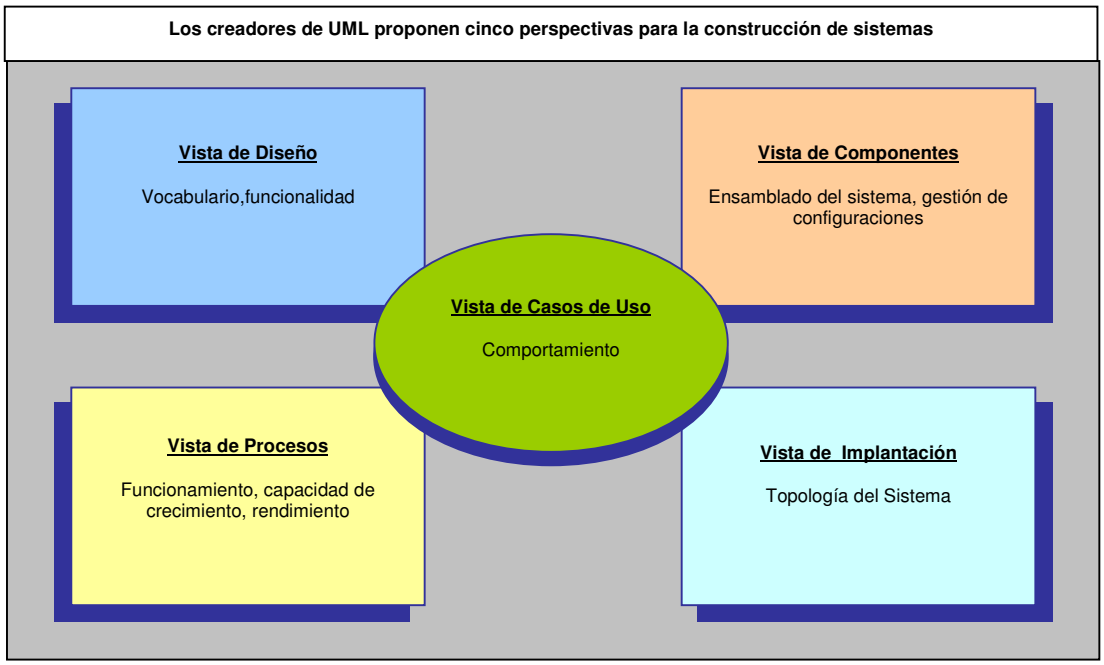
2.4.4 En qué consiste

El lenguaje unificado de modelado, es un lenguaje gráfico que facilita el modelaje de sistemas de *software*, se compone de un conjunto de bloques de construcción (gráficos) y un conjunto de reglas para interrelacionarlos. Este lenguaje se identifica como UML (por sus siglas en ingles Unified Modeling Language).

2.4.5 Para qué sirve

- Modelar aplicaciones específicas, utilizando un lenguaje sencillo de interpretación.
- Visualizar el diseño de tal modo que facilite la comunicación entre los desarrolladores y el trabajo en equipo.
- Especificar el diseño con descripciones que puedan ser fácilmente comprendidas por todos los miembros de un equipo de desarrollo y así construir el diseño a partir de las especificaciones precisas que se pueden elaborar con UML.
- Documentar el diseño por medio de documentos de UML.

Figura 1. Muestra la visión arquitectónica del sistema bajo la metodología UML



Cada una de estas vistas contemplan dos aspectos básicos:

a. Aspectos estáticos: son los que se refieren a documentos que representa objetos (diagramas de casos de uso, de objetos) y,

b. Aspectos dinámicos: son documentos que representan actividades, procesos o relaciones entre casos de uso u objetos.

2.4.6 Vista de casos de uso

Las vistas de casos de uso indican lo que debe hacer el sistema, abarcando los requerimientos del sistema descritos en forma de casos de uso, tal y como son vistos por los usuarios finales, los analistas y los encargados de las pruebas.

Los aspectos estáticos se especifican mediante diagramas de casos de uso y los aspectos dinámicos se especifican mediante diagramas de secuencias de eventos principalmente. Las vistas son las siguientes:

a. Vista de diseño

Esta vista permite visualizar cómo se soporta la funcionalidad del sistema, es decir los requerimientos. Los aspectos estáticos se especifican mediante los diagramas de objetos y de instancias, los aspectos dinámicos se especifican mediante los diagramas de secuencias de eventos y de colaboración.

b. Vista de procesos

Abarca los procesos e hilos que conforman los mecanismos de sincronización y concurrencia del sistema modelado. Los aspectos estáticos se especifican mediante los diagramas de objetos y de instancias, pero se hace énfasis en los “objetos activos”, los aspectos dinámicos se especifican mediante los diagramas de secuencias de eventos y de colaboración, pero se hace énfasis en los “objetos activos”. Los objetos activos son los que representan a procesos e hilos de un sistema.

c. Vista de componente

Abarca los archivos que se utilizan para construir el sistema, así como los módulos a que dan origen después de la compilación y el *linking*.

Esta vista también permite la gestión de la configuración del sistema pues se pueden representar versiones de archivos y módulos y correlacionarlos con las versiones del sistema.

Los aspectos estáticos se especifican por medio de los diagramas de componentes, los aspectos dinámicos se pueden especificar mediante diagramas de secuencias de eventos y colaboración, pero raramente se usan en este contexto.

d. Vista de implantación

Abarca la topología del hardware sobre el cual se implanta el sistema, así como la distribución de componentes entre los distintos nodos de la topología del hardware.

Los aspectos estáticos se especifican mediante los diagramas de implantación, los aspectos dinámicos se especifican mediante los diagramas de secuencias de eventos y de colaboración.

3 CASOS DE USO

3.1 Conceptos básicos

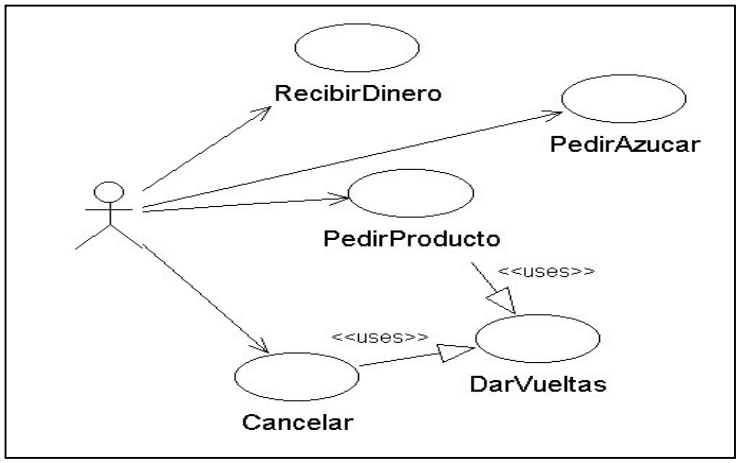
3.1.1 Diagrama de caso de uso

Un diagrama de caso de uso es un gráfico donde intervienen actores, es decir, un conjunto de casos de uso que se relacionan en el entorno de un sistema, teniendo comunicación y participación entre ellos.

Los diagramas de casos de uso son creados para visualizar la iteración de un sistema en el mundo exterior y muestra las distintas operaciones que se esperan de una aplicación o sistema y cómo se relaciona con su entorno (usuarios u otras aplicaciones).

Se muestra como ilustración los casos de uso de solicitar una taza de café.

Figura 2. Muestra casos de uso solicitar una taza de café



3.1.2 Caso de uso

Un caso de uso es:

a. La representación de un proceso o interacción en un sistema y se representa gráficamente con una elipse.
b. Describe un proceso importante para el negocio, así como la interacción entre actores y el sistema. También representa requerimientos del sistema en términos de un proceso de interacción.
c. Una secuencia de acciones que el sistema ejecuta, que produce un resultado observable para un actor en particular y denota un requerimiento solucionado por el sistema.

Cada caso de uso es una operación completa desarrollada por los actores y por el sistema en un diálogo.

El conjunto de casos de uso representa la totalidad de operaciones desarrolladas por el sistema y va acompañado de un nombre significativo.

En el caso del ejemplo anterior se tienen como casos de uso de la cafetería: recibir dinero, pedir azúcar, pedir producto, dar vueltas y cancelar.

3.1.2.1 Parámetros para la construcción de un caso de uso

Un caso de uso debe ser simple, inteligible, claro y conciso. Generalmente hay pocos actores asociados a cada caso de uso.

Entre las preguntas clave se puede mencionar:

a. ¿Cuáles son las tareas del actor?
b. ¿Qué información crea, guarda, modifica, destruye o lee el actor?
c. ¿Debe el actor notificar al sistema los cambios externos?
d. ¿Debe el sistema informar al actor de los cambios internos?

3.1.2.2 Descripción del caso de uso

La descripción del caso de uso comprende:

a. El inicio: ¿cuándo y qué actor lo produce?
b. El fin: ¿cuándo se produce y qué valor devuelve?
c. La interacción actor-caso de uso:¿qué mensajes intercambian ambos?
d. Objetivo del caso de uso: ¿qué lleva a cabo o intenta?
e. Cronología y origen de las interacciones
f. Repeticiones de comportamiento: qué operaciones son iteradas?
g. Situaciones opcionales: ¿qué ejecuciones alternativas se presentan en el caso de uso?

3.1.3 Actor

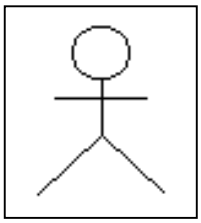
Un actor es el usuario mismo cuando desempeña un papel con respecto al sistema.

Los actores llevan a cabo casos de uso y un mismo actor puede realizar muchos casos de uso.

Los actores no necesariamente son seres humanos, también pueden ser sistemas externos que necesita información del sistema actual.

El actor es un usuario del sistema que necesita o usa algunos de los casos de uso y va acompañado de un nombre significativo si es necesario, que se representa mediante un símbolo que lo identifica como tal.

Figura 3. Muestra la representación simbólica de un actor



3.1.4 Relaciones típicas entre casos de uso

Entre los elementos de un diagrama de casos de uso se pueden presentar tres tipos de relaciones representadas por líneas dirigidas entre ellos, del elemento dependiente al independiente.

*3.1.4.1 Comunica (*communicates*)*

Relación entre un actor y un caso de uso. Denota la participación del actor en el caso de uso determinado.

En el diagrama de ejemplo todas las líneas que salen del actor denotan este tipo de relación.

*3.1.4.2 Usa (*uses*)*

Relación entre dos casos de uso y denota la inclusión del comportamiento de un escenario en otro.

En el caso del ejemplo el caso de uso cancelar incluye en su comportamiento dar vueltas y pedir producto, incluye también dar vueltas

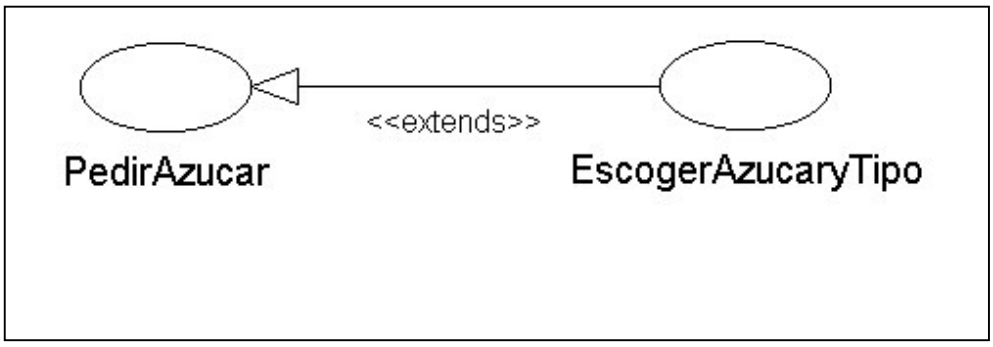
*3.1.4.3 Extiende (*extends*)*

Relación entre dos casos de uso y denota cuando un caso de uso es una especialización de otro.

Por ejemplo, podría tenerse un caso de uso que extienda la forma de pedir azúcar, parta que permita escoger el tipo de azúcar (normal, dietético moreno) y además la cantidad en las unidades adecuadas para cada caso (cucharaditas, bolsitas o cucharaditas, respectivamente).

Un posible diagrama se muestra a continuación:

Figura 4. Muestra las relaciones típicas entre casos de uso



Aparte de la descripción que acompaña a cada caso de uso, un diagrama de casos de uso llevará asociada una descripción textual, en forma de flujos de eventos, de cada caso de uso representado.

Se debe tener en consideración:

a. Flujo de eventos principal

Se trata de una descripción de los eventos que van aconteciendo en el uso habitual, es decir, cuando no se presenta ningún tipo de problema (es el denominado happy path).

b. Flujo de eventos excepcional

Podemos encontrar tantos apartados de este tipo como situaciones excepcionales se puedan plantear, siendo que para cada uno de estos escenarios atípicos se definirá el flujo de eventos correspondiente.

Tabla I. Muestra ejemplo caso de uso: hacer pedido

Flujo de eventos principal:

- Include(Validar cliente)
- El sistema muestra una lista con los datos de una serie de productos seleccionables.
- El cliente selecciona los items que desea comprar y sus respectivas cantidades.
- El cliente valida la selección.
- El sistema recoge la lista de items seleccionados por el cliente.
- Establecer prioridad
- El sistema envía los datos del pedido para su proceso.
- Fin del caso de uso.

Flujo de eventos excepcional:

- El cliente valida un pedido en que no ha seleccionado ningún producto.
- El sistema vuelve a mostrar la lista de productos seleccionables.
- El cliente valida un pedido en que la cantidad seleccionada para un producto excede de la disponible.
- El sistema lo notifica al cliente y muestra la lista de productos seleccionados dando opción a cambiar la cantidad del producto.

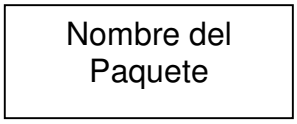
3.1.4.4 Paquetes de casos de uso

Los casos de uso se empaquetan de acuerdo a su funcionalidad, por lo que es conveniente agruparlos por tema, cada agrupamiento se debe asociar con un paquete.

Un sistema de casos de uso puede contener paquetes de casos de uso, casos de uso y relaciones entre casos de uso. Los paquetes de casos de uso, se pueden aprovechar para organizar jerárquicamente los diagramas de casos de uso.

Los paquetes se representan en un diagrama de casos de uso con el símbolo.

Figura 5. Muestra la representación de los paquetes en un diagrama



3.1.4.5 Nomenclaturas y reglas de estilo

Los nombres de los casos de uso deben reflejar que se trata de procesos del negocio, por lo que es conveniente usar verbos en su forma infinitiva (comprar, usar, obtener).

Los nombres de los actores deben representar tipos de personas (encargado, jefe, digitador) o a algún sistema específico u organización con que el sistema interactúa. Un actor se representa con el estereotipo del *stick man* ubicando el nombre del actor en la parte inferior

Para identificar y documentar los casos de uso se pueden seguir los siguientes métodos:

a. Por medio de los actores

Identificar primero los actores relacionados con el sistema o la organización. Para cada actor identificar los procesos en que participan o inician.

b. Por medio de los eventos

Identificar los eventos externos a los que el sistema debe responder, relacionarlos con actores y luego con casos de uso.

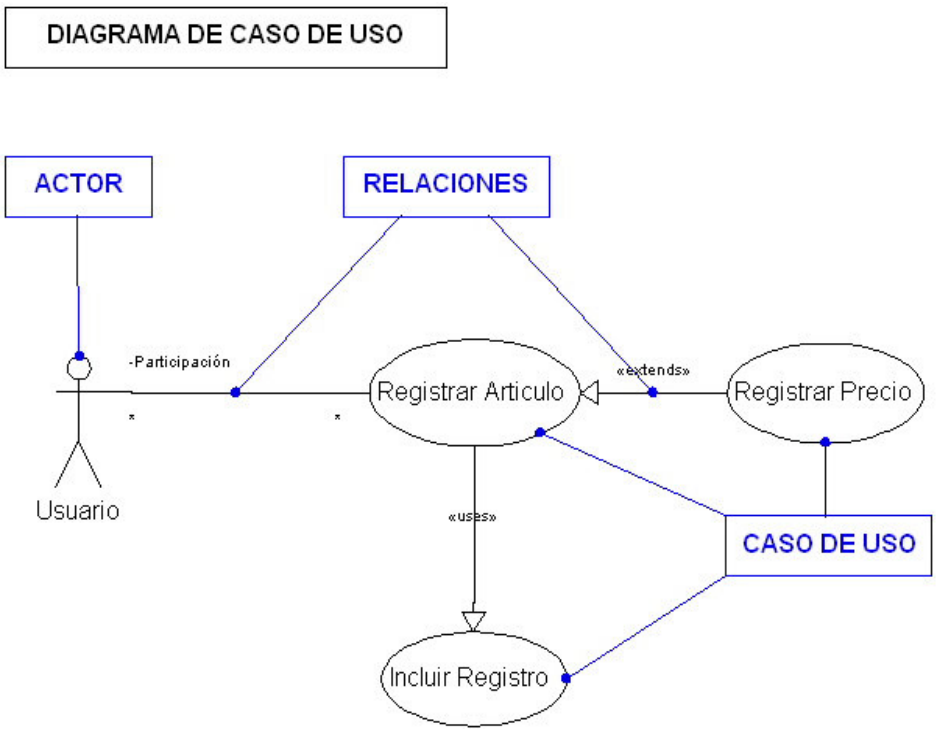
c. Especificación de casos de uso

- Nombre del caso de uso.
- Actores involucrados.
- Propósito.
- Descripción general.
- Funciones asociadas.
- Curso típico de eventos.
- Diseño de pantallas y formas.

Se especificará un caso con una secuencia de cuatro documentos electrónicos unidos por hipervínculos:

- Un diagrama de casos de uso.
- Un texto descriptivo general (nombre, actores, propósito, descripción y funciones).
- Un texto describiendo el curso típico de eventos.
- Un gráfico con el diseño de la pantalla.

Figura 6. Muestra ejemplo de un caso de uso utilizando relaciones



3.1.4.6 Especificación de casos de uso

- Nombre del caso de uso: registrar artículos.
- Actores involucrados: usuario.
- Propósito: mostrar la interacción del sistema al incluir un artículo.
- Descripción general: el usuario requiere ingresar un nuevo artículo al catálogo.

3.1.4.7 Algunas funciones asociadas

- Registrar artículos usando un lector de código de barras.
- Desplegar la descripción y el precio de cada artículo comprado.

3.1.4.8 Curso típico de eventos

El curso típico de eventos toma en cuenta las acciones de un actor y las acciones del sistema.

Tabla II. Muestra acciones del actor y del sistema

Acción de un actor	**Acción del sistema**
El usuario solicita al sistema la pantalla para registrar artículos	El sistema muestra la pantalla de mantenimiento de artículos
El usuario selecciona la opción de Inclusión	Habilita los campos necesarios para completar la información y posiciona el cursor en el primer control de lectura (TextBox)
El usuario digita el código del artículo	El sistema valida el valor digitado para identificar cualquier error
El usuario digita el nombre del articulo	El sistema valida el valor digitado para identificar cualquier error
El usuario continua digitando los demás valores	El sistema evalúa y responde con diferentes mensajes de validación
El usuario solicita salvar la información	El sistema revalida todos los datos y salva la información en la base de datos

3.1.4.9 Priorización de casos de uso

En la priorización de casos de uso se revisa con el inicio de cada ciclo de construcción, de tal manera que puede variar de acuerdo con la retroalimentación constante que se da durante el proceso.

Se utilizará el siguiente de método de priorización de los casos de uso.

a. Se debe construir una tabla de categorías de priorización cuyos rangos deben ir entre 4, como valor mínimo y 16 como valor máximo, sin importar el número de categorías.
b. Se realiza una tabla como ejemplo, que contiene 3 categorías de priorización (tabla 3):

Tabla III. Muestra categorías de priorización

Categoría	**Rango inicial**	**Rango final**
Alta	16	12
Media	11	8
Baja	7	4

c. La priorización de los casos de uso, se realiza en base a los siguientes criterios:

Tabla IV. Muestra criterios de la priorización de los casos de uso

Criterio	Descripción
Diseño	Impacto del caso de uso en el diseño del sistema en términos de cantidad de clases o de tablas adicionales en la base de datos.
Riesgo	El caso de uso está asociado con funciones cuyos tiempos de respuesta son críticos, o son muy complejas, o son muy importantes para el cliente o tienen asociados riesgos.
Conoci-miento	El caso de uso está asociado con el uso de tecnología poco conocida o que todavía se encuentra en fase de diseño experimental o de pruebas preliminares.
Esfuerzo	El caso de uso permite profundizar mucho en la comprensión del diseño del sistema con un esfuerzo relativamente pequeño.

Tabla V. Muestra integración de las Categorías de priorización con los Casos de Uso

CASO DE USO	DISEÑO	RIESGO	CONO-CIMIENTO	COSTO	SUMA	PRIORIDAD
Registrar Artículos	4	4	4	4	16	**ALTA**
Registrar Precios	2	2	3	2	9	**MEDIA**
Incluir Registro	1	1	1	1	4	**BAJA**

4 MODELO DE OBJETOS

4.1 Conceptos básicos

4.1.1 Modelo de objetos

El modelo de objetos o clases (en adelante modelo de objetos) representa las entidades, conceptos, objetos, personas, organizaciones y procesos, así como sus relaciones, con el sistema en construcción. Algunos de los objetos del modelo representan entidades con existencia física, otros pueden ser invención del analista / programador para estructurar un diseño óptimo.

En un sistema de apoyo administrativo, los objetos típicamente representan las entidades cuya información debe almacenarse en alguna base de datos. Sin embargo, para mantener una arquitectura en tres capas, es necesario diferenciar las tablas de la base de datos, de los conceptos representados en el modelo de objetos.

Esto implica que debe implementarse una clase para cada objeto del modelo que sea independiente de la tabla correspondiente de la base de datos. La clase estará encargada del manejo de los datos del objeto en memoria principal, y se basará en la tabla para asegurar su persistencia.

Un modelo de objetos generalmente se construye a partir de casos de uso, sin embargo, es frecuente que ambos documentos se elaboren y afinen en forma paralela. Para cada objeto se puede representar sus atributos, métodos, inclusive se puede indicar si son públicos, privados o protegidos. Para cada relación se puede especificar los roles, así como el número asociado a cada rol.

4.1.2 Objeto

Es una unidad lógica de diseño, una unidad de modelaje (Empleado, Estudiante, Cliente o Sistema de Punto de Venta), representa algún concepto relevante del dominio de la aplicación o algún concepto técnico.

El objeto se implementa con estructuras de datos y funciones que existen en memoria RAM durante la ejecución de un programa, no representa una entidad en un modelo entidad-relación y tampoco representa una tabla de una base de datos relacional. Con frecuencia se empaqueta junto con otros para constituir un componente que se ubicará en algún servidor para un sistema distribuido.

4.1.3 Componente

Es una unidad física de diseño que provee un conjunto pequeño de funcionalidades en el contexto de un sistema, las funcionalidades se representan como interfaces y se implementa con estructuras de datos y funciones que existen en memoria RAM durante la ejecución de un programa.

Algunas de sus características son:

- Representa en un sistema distribuido, por ejemplo, a un componente-COM
- No representa una entidad en un modelo entidad-relación.
- No representa una tabla de una base de datos relacional, una tabla es un tipo específico de componente.

Con frecuencia incluye varios objetos fuertemente acoplados, y se ubicará en algún servidor para un sistema distribuido.

4.1.4 Entidad

Es una unidad lógica de diseño de una base de datos relacional, representa agrupaciones de objetos: empleados, estudiantes, clientes, representa algún concepto relevante del dominio de la aplicación que requiere persistencia. Sus características son:

- No se implementa con estructuras de datos y funciones que existen en memoria RAM durante la ejecución de un programa.
- Se representa mediante tablas de una base de datos relacional, dichas tablas son tipos específicos de componentes (estereotipables), aunque puede tener asociados procedimientos almacenados (store-procedures).
- No constituye un módulo de software como los componentes.

4.1.5 Atributos

Son las propiedades o datos que componen un objeto. Un ejemplo de atributos se ilustra en la siguiente tabla.

Tabla VI. Muestra ejemplo de atributos

ATRIBUTO	DESCRIPCION
Sistema Operativo	Ambiente operativo de la aplicación.
Tiempo de Respuesta	Lapso de Tiempo de procesamiento esperado.
Interfaz	Forma en las que se visualizan las pantallas.
Plataforma	Plataforma sobre la cual se ejecuta la aplicación.
Base de Datos	Motor de base de datos a utilizar.
Seguridad	Esquema de seguridad básica de acceso al sistema y control de la información.

4.1.6 Métodos

Los métodos representan el comportamiento del objeto.

4.1.7 Abstracción

La abstracción son las características esenciales que distinguen a un objeto de otros tipos de objetos, definiendo fronteras conceptuales, relativas al observador.

La abstracción surge del reconocimiento de similitudes entre ciertos objetos, situaciones o procesos en el mundo real. Así mismo La abstracción decide concentrarse en las similitudes e ignorar las diferencias y enfatiza detalles con significado para el usuario, suprimiendo aquellos detalles que, por el momento, son irrelevantes o distraen de lo esencial.

La abstracción debe seguir el principio de mínimo compromiso, que significa que la interfase de un objeto provee su comportamiento esencial, y nada más que eso. Además también el principio de mínimo asombro: capturar el comportamiento sin ofrecer sorpresas o efectos laterales.

4.1.8 Encapsulación

Es el proceso de compartir los elementos de una abstracción que constituyen su estructura y comportamiento. La encapsulación sirve para separar la interfase de una abstracción y su implementación.

Es un concepto complementario al de abstracción, la encapsulación esconde la implementación del objeto que no contribuye a sus características esenciales.

La encapsulación da lugar a que las clases se dividan en dos partes:

a. Interfase

Captura la visión externa de una clase, abarcando la abstracción del comportamiento común a los ejemplos de esa clase.

b. Implementación

Comprende la representación de la abstracción, así como los mecanismos que conducen al comportamiento deseado.

La encapsulación se conoce también como ocultamiento o privacidad de la información.

4.1.9 Modularidad

Es la propiedad que tiene un sistema que ha sido descompuesto en un conjunto de módulos cohesivos y vagamente conexos.

Cada módulo se puede compilar separadamente, aunque tengan conexiones con otros módulos. En un diseño estructural, modularización comprende el agrupamiento significativo de subprogramas. En diseño orientado a objetos, la modularización debe ceñirse a la estructura lógica elegida en el proceso de diseño, dividir un programa en componentes individualizados reduce en alguna manera su complejidad.

4.1.10 Jerarquización

Es una clasificación u ordenación de las abstracciones, por jerarquía se denota el orden de relación que se produce entre abstracciones diferentes.

Los tipos de jerarquía más útiles son:

a. Herencia (generalización / especialización, padre / hijo, jerarquía del tipo es un...), una clase (subclase) comparte la estructura o comportamiento definido en otra clase, llamada superclase.
b. Herencia múltiple, una clase comparte la estructura o comportamiento de varias superclases.

Agregación comprende relaciones del tipo -es parte de- al realizar una descomposición, relaciones entre los conceptos asociados al modelo de objetos.

Los conceptos de abstracción y encapsulación son conceptos complementarios: la abstracción hace referencia al comportamiento observable de un objeto, mientras encapsulación hace referencia a la implementación que la hace alcanzar este comportamiento.

4.1.11 Tipificado

Tipificar es la imposición de una clase a un objeto, de tal modo que objetos de diferentes tipos no se puedan intercambiar, o se puedan intercambiar solo de forma restringida.

Tipo es una caracterización precisa de las propiedades estructurales y de comportamiento que comparten una colección de entidades, grosso modo, tipo y clase pueden considerarse sinónimos.

4.1.12 Concurrencia

Es la propiedad que distingue un objeto activo de uno no activo. Concurrencia permite que diferentes objetos actúen al mismo tiempo.

4.1.13 Persistencia

Es la propiedad por la cual la existencia de un objeto trasciende en el tiempo (esto es, el objeto sigue existiendo después de que su creador deja de existir) o en el espacio (esto es, la localización del objeto cambia respecto a la dirección en la que fue creado).

4.2 Relaciones entre entidades, objetos y componentes

Las entidades constituyen el modelo lógico de la base de datos que da persistencia a los objetos.

La relación entre objetos y entidades no siempre es uno a uno y los objetos complejos (compuestos por varios objetos) posiblemente necesiten varias tablas.

Identificar entidades para el diseño de una base de datos no es lo mismo que identificar objetos para un modelo de objetos y esto puede pasar con las relaciones.

La identificación y conformación de componentes obedece más a criterios de diseño físico que de diseño lógico.

4.3 Nomenclaturas y reglas de Estilo

Para construir modelos de objetos se deben utilizar los siguientes símbolos, representados en las figuras:

Figura 7. Muestra símbolo de objeto

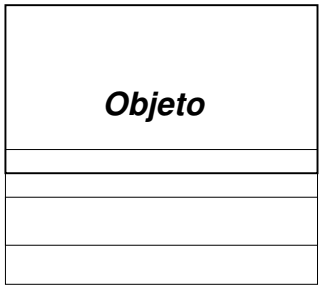


Figura 8. Muestra símbolo de componente

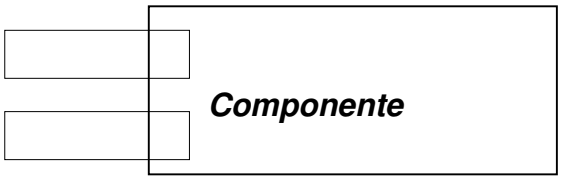
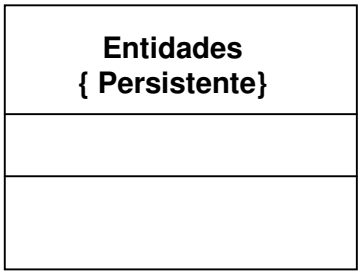
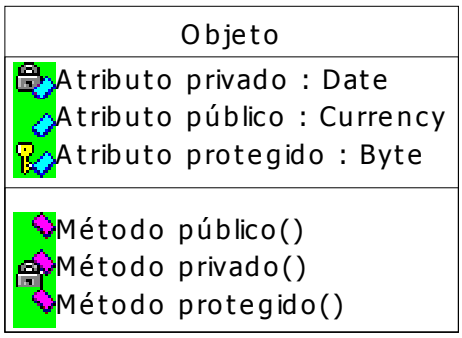


Figura 9. Muestra símbolo de entidad



Para un objeto se pueden especificar los atributos y métodos públicos y privados protegidos:

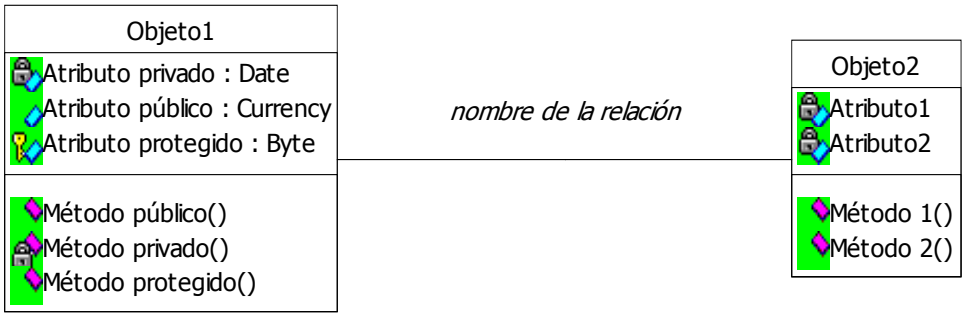
Figura 10. Muestra especificación de atributos y métodos



4.4 Relaciones o asociaciones entre objetos

Una asociación entre objetos se representa por medio de una línea continua.

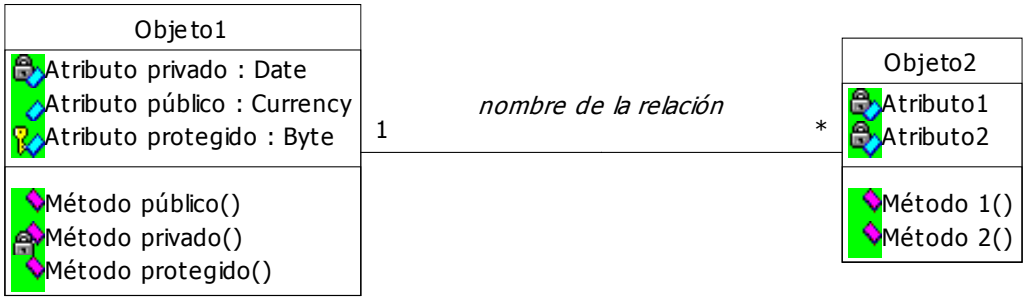
Figura 11. Muestra modelo de relaciones o asociaciones



4.5 Multiplicidad

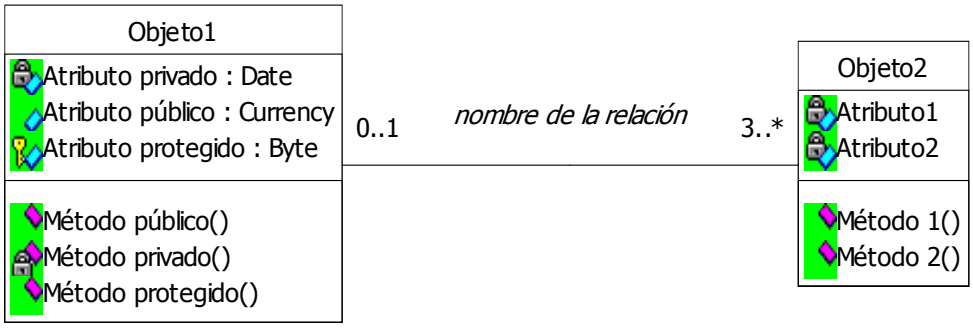
Se puede indicar la multiplicidad de la asociación en ambos lados con un solo carácter.

Figura 12. Muestra modelo de relaciones o asociaciones con un solo carácter



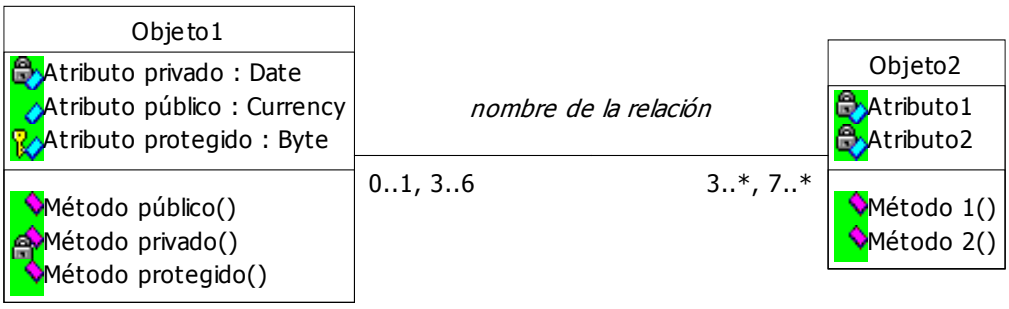
Se puede indicar la multiplicidad de la asociación en ambos lados con un rango.

Figura 13. Muestra modelo de asociación en ambos lados con un rango



Se puede indicar la multiplicidad de la asociación en ambos lados, con varios rangos.

Figura 14. Muestra modelo de asociación en ambos lados con varios rangos



Se pueden especificar los roles de los objetos en una asociación. El rol 1, es el papel que juega el objeto 1 en la asociación y el rol 2, es el papel del objeto 2.

Figura 15. Muestra roles de los objetos en una asociación

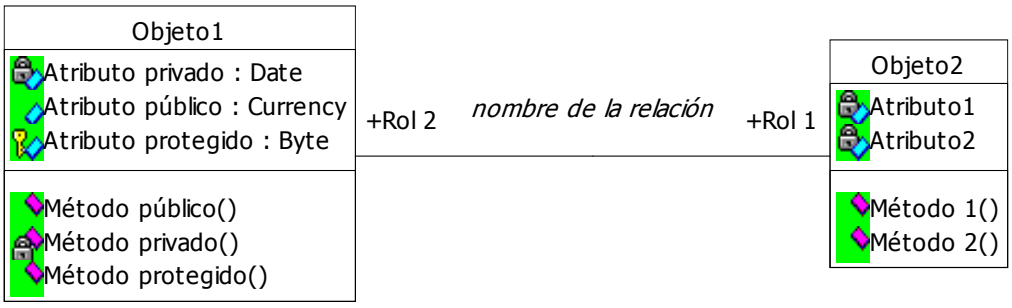
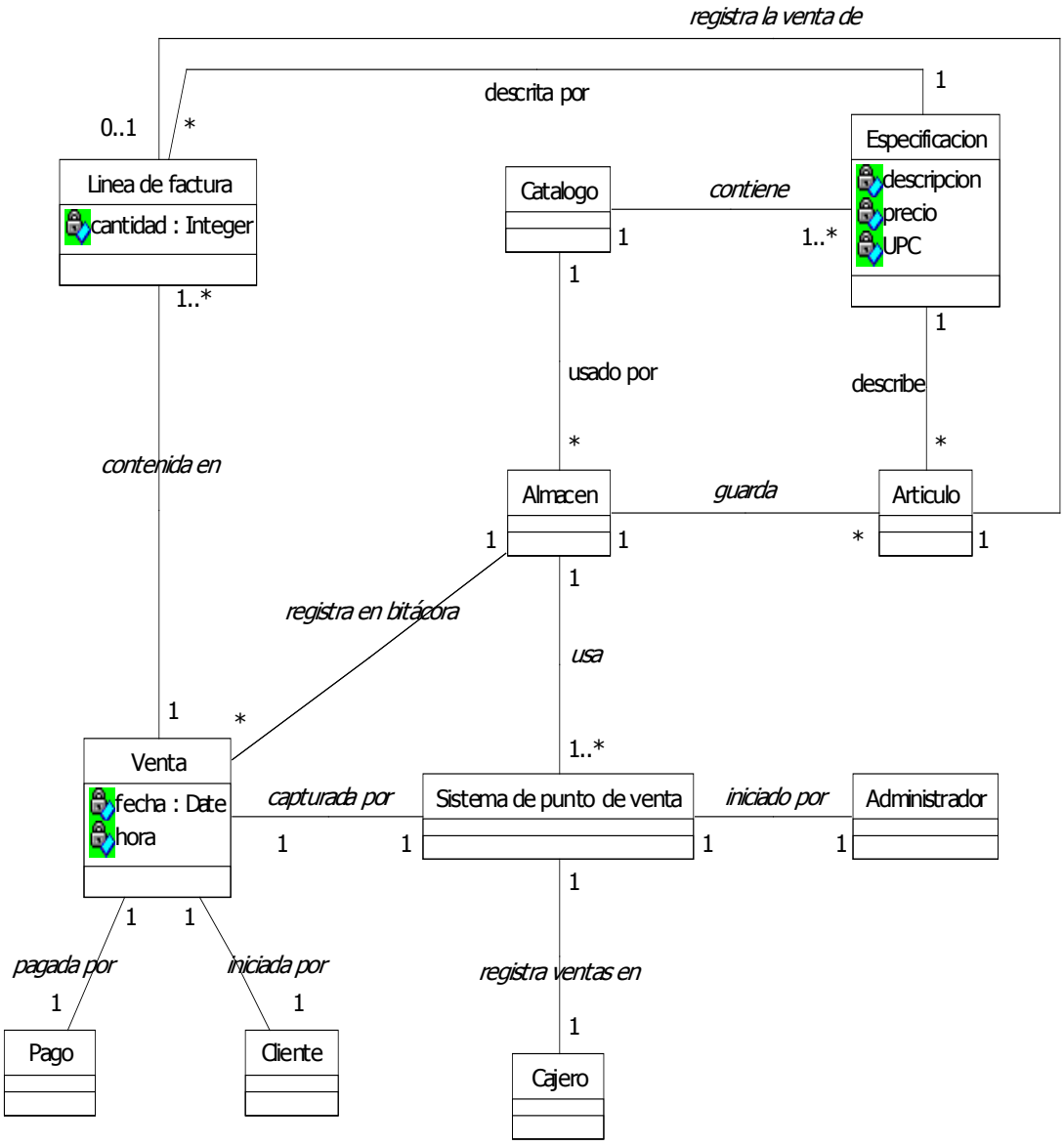


Figura 16. Muestra modelo de un objeto



5 IMPLEMENTACIÓN DEL PROYECTO UTILIZANDO EZWAY

5.1 Requisitos minimos de implementación

Para un mejor aprovechamiento y comprensión es recomendable tener conocimientos previos en microsoft *visual basic*, microsoft visio *professional*, sistema operativo windows, MS *project professional*, case *studio* y SQL server .

En *hardware* las estaciones de trabajo deben contar con: procesador pentium III o superior, memoria RAM 128 Mb., espacio disponible en disco 30 Mb. (para Instalación) y unidad de CD-ROM. El servidor debe tener las características procesador pentium III o superior, memoria RAM 256 Mb., espacio disponible en disco 10 Gb. (para instalación) y unidad de CD-ROM.

En *software* el sistema operativo debe contar con microsoft windows SP 3, 4, *option pack*), microsoft windows 98 (*second edition*) o microsoft windows 2000 *professional*. microsoft *Visual Studio* 6.0 con *service pack* 5 ó microsoft *visual basic* 6.0 con *service pack* 4 como herramientas de programación.

El servicio de transacciones debe estar en Windows NT 4.0 Microsoft *transaction server* 2.0 (o*ption pack*), en Windows 98SE Microsoft *transaction server* 2.0 y en Windows 2000.

Propiedades de la base de datos: microsoft SQL *server* 2000. Como *software* complementario: microsoft internet *explorer* ó netscape , microsoft *data access components* o Ezway *connection manager*.

Para el desarrollo web en windows 98SE microsoft personal *web Server*, en windows 2000: *Internet information server* (IIS).

Como *software* opcional: microsoft visio *professional*, microsoft *project manager*, microsoft *visual modeler* o *rational rose*.

5.2 Descripción del proyecto de aplicación

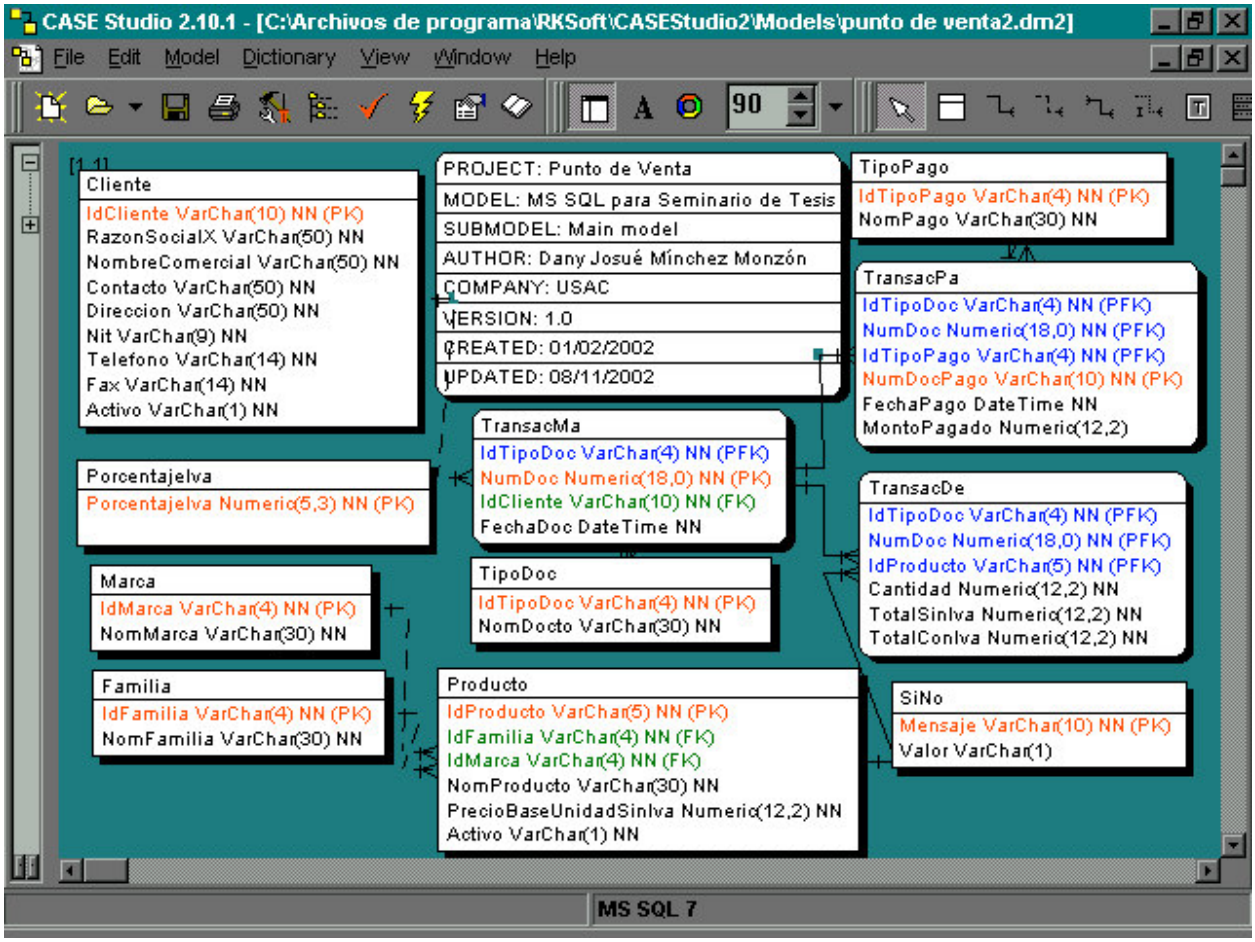
El sistema consiste en un punto de venta donde se llevará el control de clientes, porcentaje de IVA, marcas, familias, tipos de documentos, productos, tipos de pago, facturas y pagos por factura. Además, se creará toda la documentación técnica necesaria utilizando freeway, visio *professional* y *project professional*, siendo la documentación generada con dichas herramientas, será desplegada en un intranet corporativa.

5.3 Herramienta Case Studio

A continuación se encontrarán los pasos a realizar en la herramienta case estudio.

a. Realizar el diagrama entidad-relación completo del proyecto a implementar.

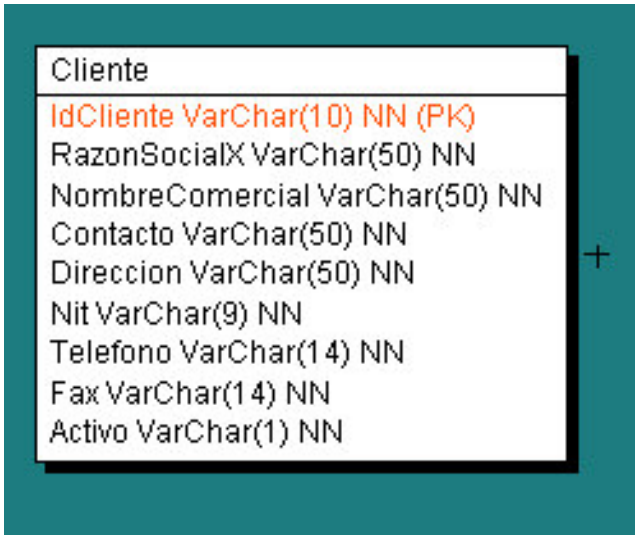
Figura 17. Muestra diagrama Entidad-Relación



Fuente. **Case estudio versión 2.10.1**

b. Incluir un nombre de campo erróneo (cliente.razonsocialx)

Figura 18. Muestra indicar el nombre de campo

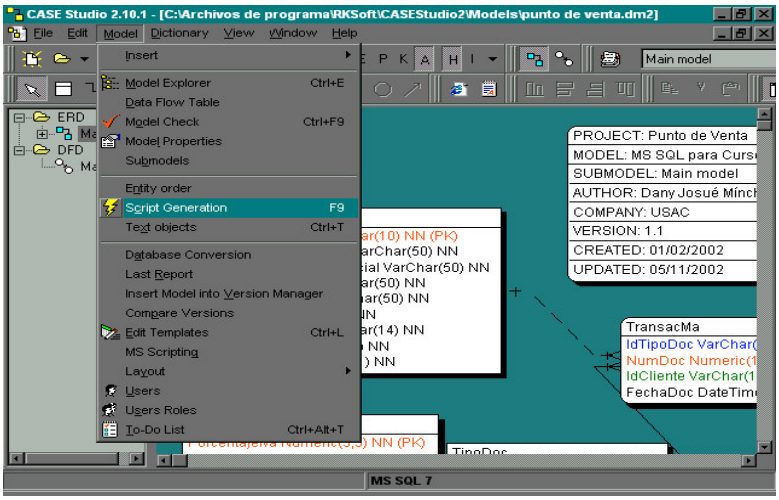


Fuente. **Case estudio versión 2.10.1**

El campo erróneo se coloca para verificar la facilidad con que se actualiza el repositorio.

c. Generar el *Script* y cargar en SQL server.

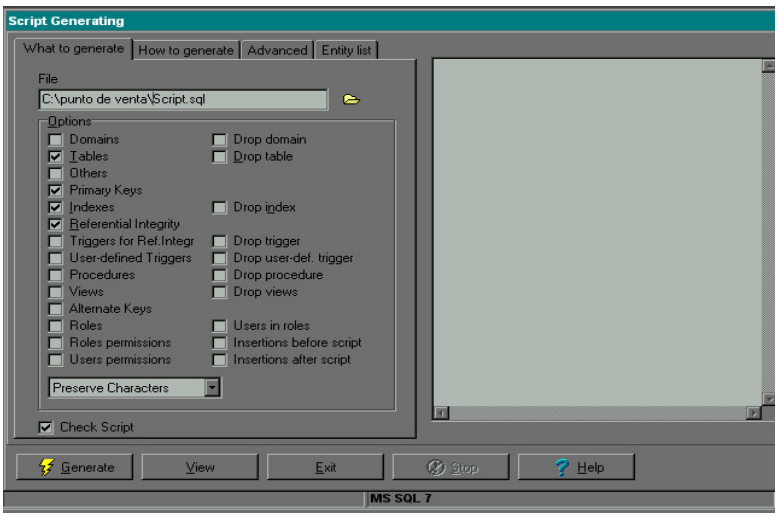
Figura 19. Muestra generación de Script



Fuente. **Case estudio versión 2.10.1**

d. Indicar el directorio y nombre del archivo donde será generado el *script.*

Figura 20. Indicar el directorio y nombre para generar el Script



Fuente. **Case estudio versión 2.10.1**

e. Crear los directorios en el disco personal

C:\Archivo de programa\EzDocumenter\DiccionarioDato\Facturación\)

f. Generar documentación en HTML

(Directorio:C:\Archivosdeprograma\EzDocumenter\DiccionarioDato\ Facturacion\)

5.4 Pasos a realizar en MS SQL

a. Crear la base de datos del proyecto a desarrollar en IIS (Internet *information server*)

b. Crear el directorio virtual para la intranet \ (proyecto tipo ASP)

- http://localhost/facturacion/
- C:\Documents and Settings\dminchez\Mis documentos\My Projects\Facturación\

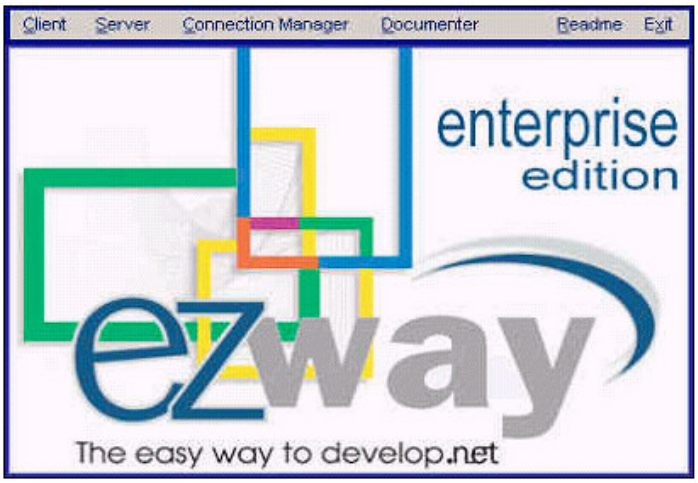
c. Redireccionar el IIS default para ver la documentación (*ezdocumenter*)

- C:\Archivos de programa\EzDocumenter\Asp

5.5 Instalación de eZway

Aquí se pretende introducir a la instalación de eZway, indicando como realizarla.

Figura 21. Muestra ventana de instalación de eZway

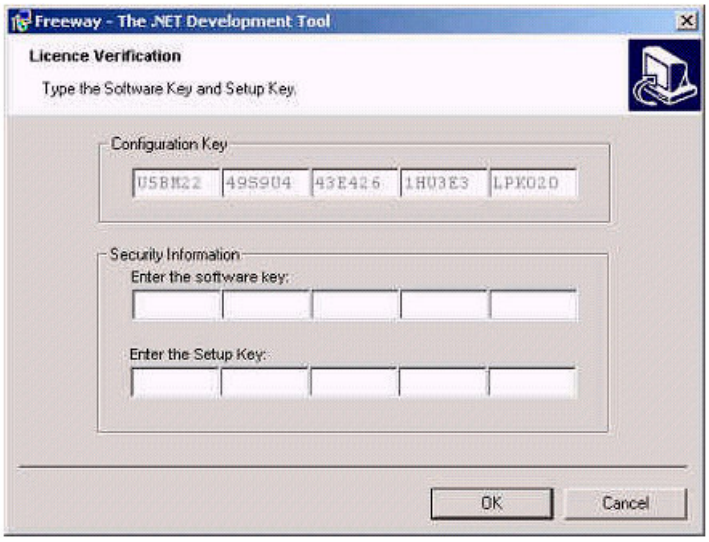


Fuente. **Ezway Connection Manager**

5.5.1 Pasos de la instalación del cliente

- Proceda a ejecutar la opción *client* de la ventana de instalación.
- Indique las llaves de instalación del producto. La llave *configuration key* es una llave generada específicamente para cada equipo donde se instale eZway. La llave *software key* va adjunta con el producto en la caja del CD y por último, la llave *setup key*, deberá ser solicitada a freeway *development Corp.*

Figura 22. Muestra ventana de verificación de licencias



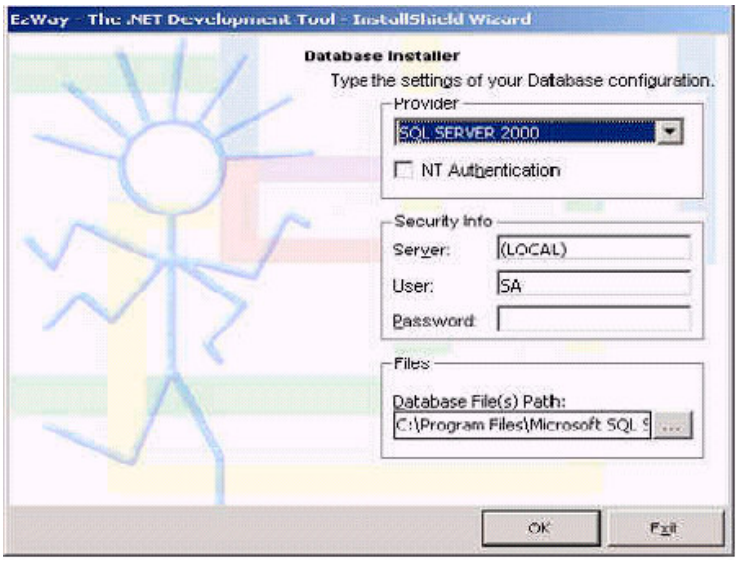
Fuente. **eZway Enterprise Edition**

- Una vez ingresadas las llaves de instalación, presione *ok*, lea los términos que incluye la instalación del producto, e indique si acepta los términos, de ser aceptados, presione *Next*, luego indique la información acerca del usuario que podrá tener acceso a la herramienta, seleccione instalación completa, presione *Next* y por último presione *Install*. El proceso de instalación seguirá automáticamente a partir de este punto.

5.5.2 Pasos para la instalación del Server

- En este proceso únicamente se instala el repositorio de datos de eZway, por lo que sí el producto se va a utilizar en máquinas clientes conectadas a un servidor, esta opción debe ejecutarse en el servidor de bases de datos, en caso de que cada estación requiera utilizar eZway en forma local, entonces ejecute esta parte de la instalación en la máquina cliente.
- Presione click sobre la palabra *server* de la ventana de instalación

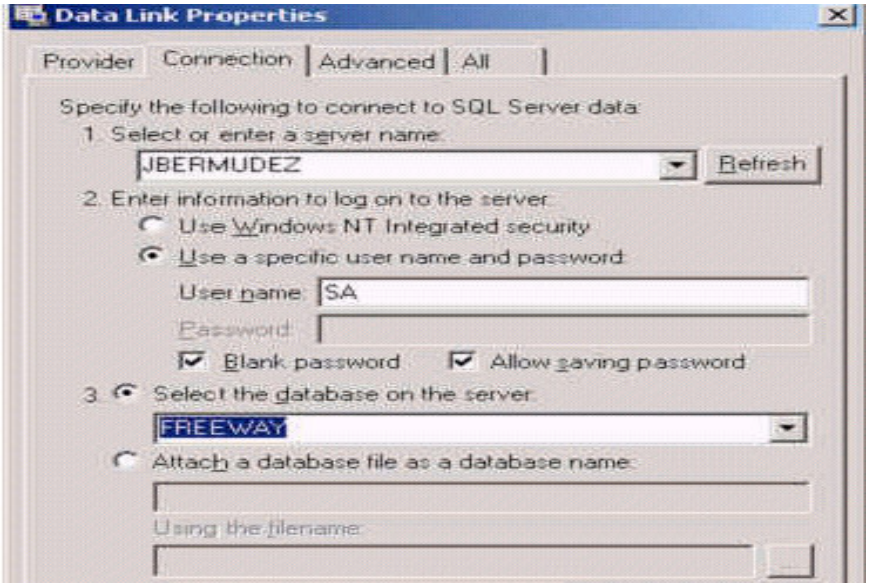
Figura 23. Muestra ventana de instalación del repositorio de eZway



Fuente. **eZway Enterprise Edition**

- El instalador del *server* ubica el repositorio donde serán almacenados los proyectos modelados de acuerdo a los datos indicados en la pantalla desplegada. Indique el proveedor de servicio de base de datos y versión, además, indique si su servidor actualmente está configurado para reconocer la autenticación vía sistema operativo o no. Luego indique el nombre del servidor, el *login* y el *password*. Por último indique la ubicación física de los archivos del repositorio.
- La primera vez que ingrese a eZway, este le preguntará a que repositorio desea conectarse con la siguiente pantalla, para lo cual seleccione en proveedor, el proveedor de la base de datos en donde fue instalado el repositorio y en la cejilla de conexión, indique la información de su servidor y o su sistema de archivos. Recuerde que el repositorio normalmente se llama freeway.

Figura 24. Muestra ventana de configuración de la conexión del repositorio de eZway

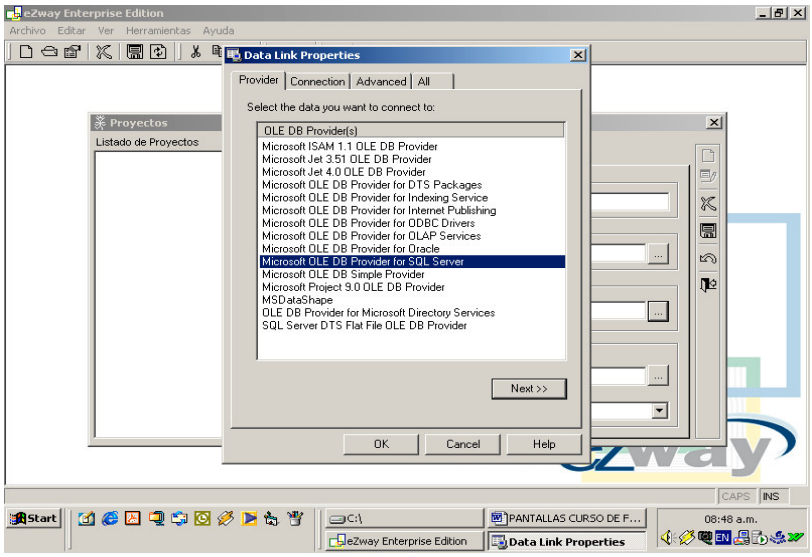


Fuente. **eZway Enterprise Edition**

5.6 ODBC direccionado a facturación

Aquí se indica como instalar el ODBC llamado facturación, dentro de eZway.

Figura 25. Muestra donde colocar el nombre del ODBC

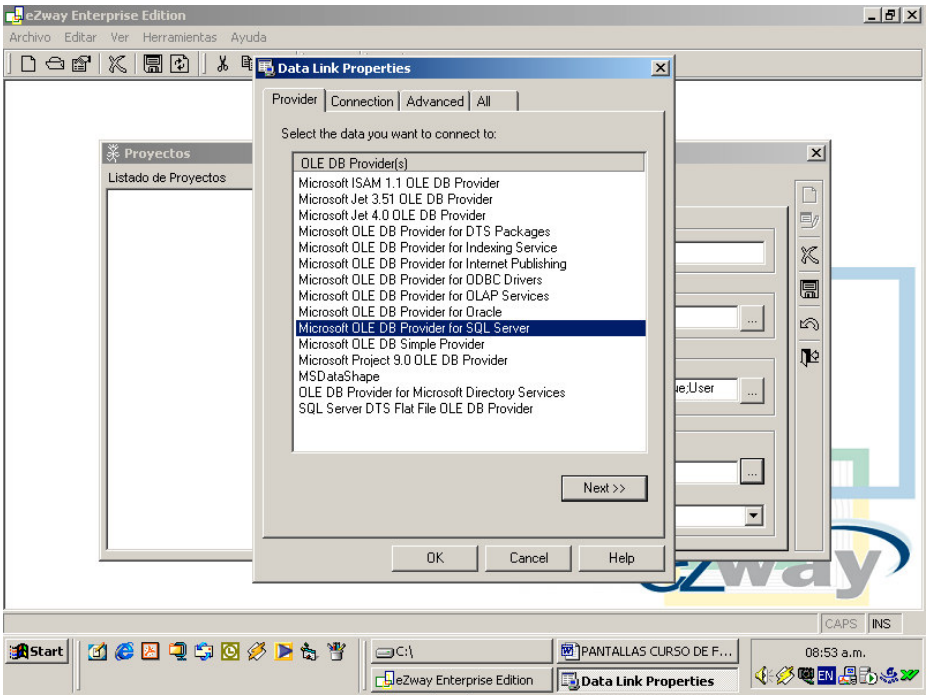


Fuente. **eZway Enterprise Edition**

5.7 ODBC direccionado a FREEWAY

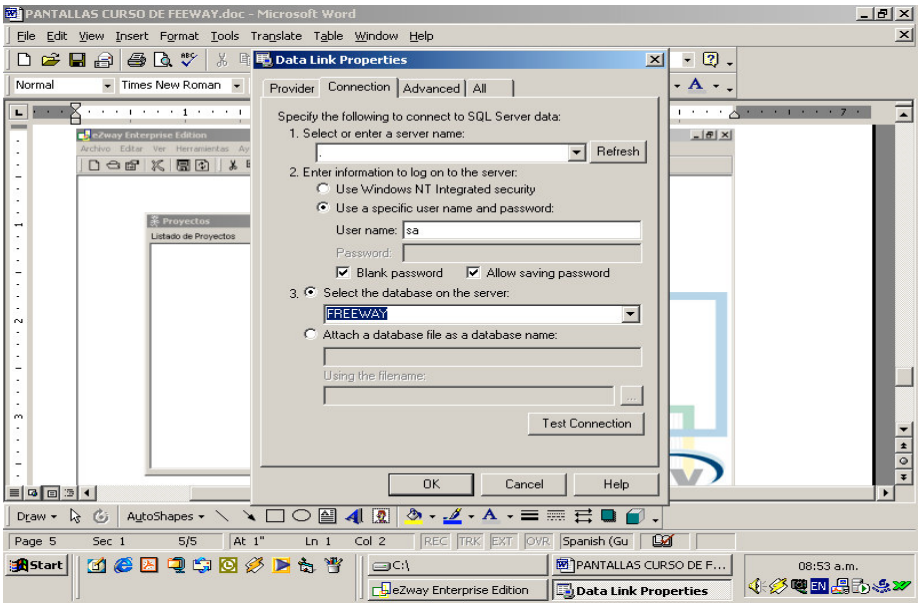
Aquí se indica como instalar el ODBC, llamado freeway que nos servirá como repositorio de eZway.

Figura 26. Muestra donde colocar el nombre FREEWAY-1



Fuente. **eZway Enterprise Edition**

Figura 27. Muestra donde colocar el nombre FREEWAY -2



Fuente. **eZway Enterprise Edition**

5.8 Configuración de los servicios de componentes

La configuración de los servicios de componentes se realiza de la siguiente manera:

Inicio->ConFiguración Panel de Control->Herramientas Administrativas->Servicios de Componentes-> Servicios de Componentes->Equipos->Mi equipo, clic derecho->Propiedades->Seguridad Predeterminada->(Permisos de acceso predeterminados)Modificar predeterminados...->Agregar->Todos. (Next, Next, Upsss)

5.9 Ingeniería inversa a VISIO

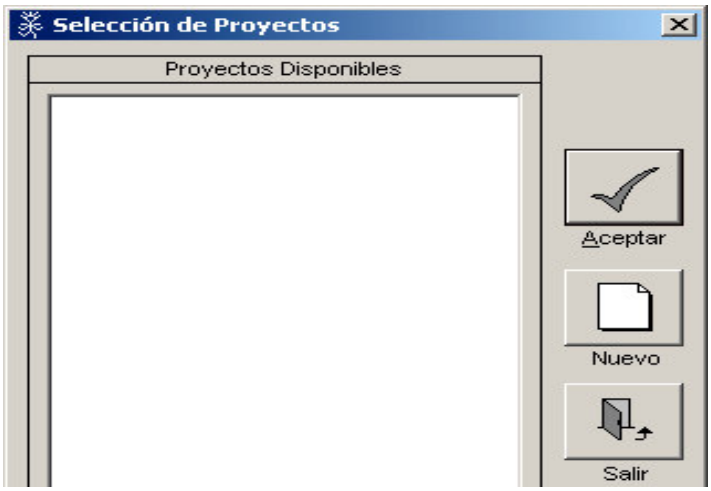
- Corregir el nombre de campo erróneo (Cliente.RazonSocialX)
- Actualizar el repositorio

5.9.1 Nuevo proyecto y selección

Al realizar un nuevo proyecto se debe realizar lo siguiente:

- Colocar la ruta de http para la Intranet (http://localhost/facturacion/**)**
- Direccionar a la BD de facturación a la base de datos Facturación
- Direccionar a la BD de eZway a la base de datos freeway
- Direccionar el ODBC para Project (Nombre de ODBC=freeway)

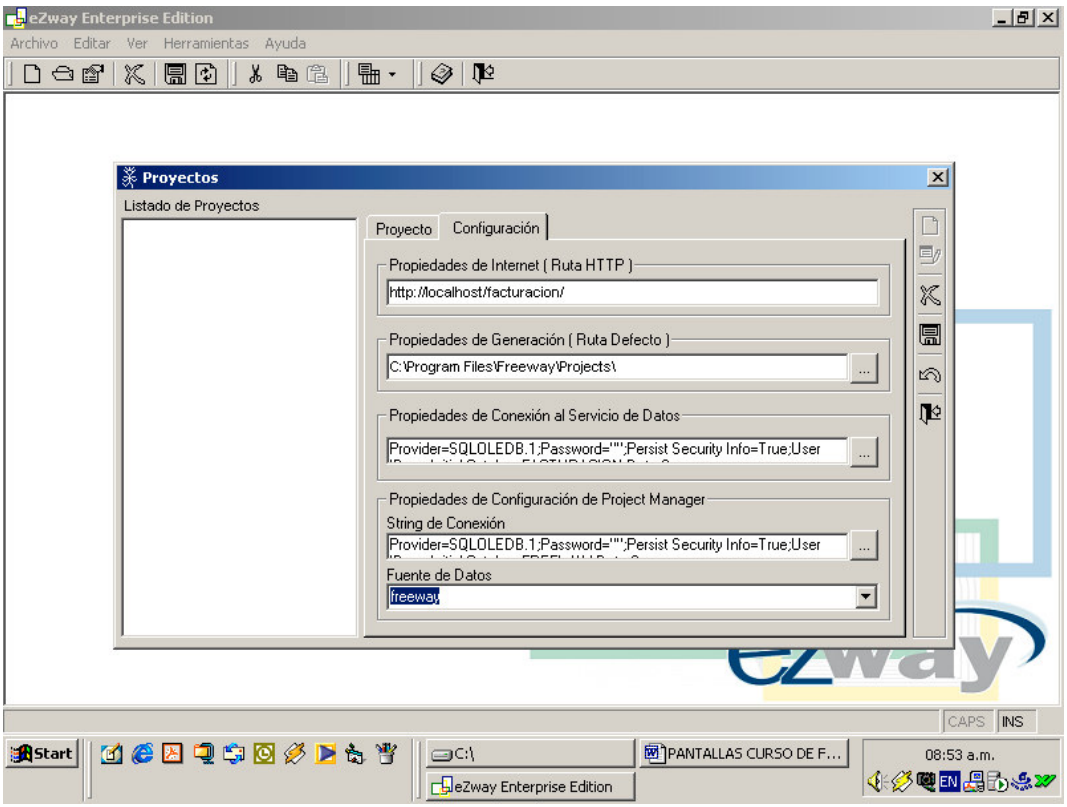
Figura 28. Muestra selección de proyectos



Fuente. **eZway Enterprise Edition**

Se encontraran también los proyectos disponibles (realizados anteriormente).

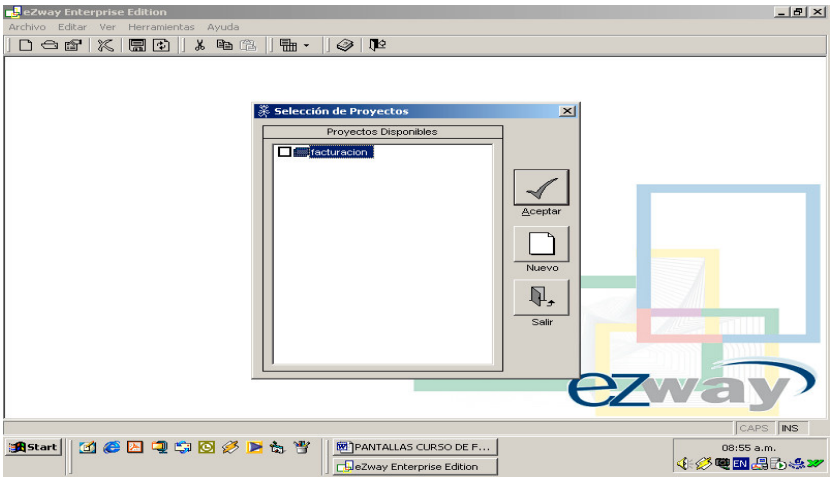
Figura 29. Muestra pantalla de configuración de proyectos.



Fuente. **eZway Enterprise Edition**

Se indica la dirección en la intranet, propiedades de generación, propiedades de conexión, fuente de datos.

Figura 30. Muestra presentación de la pantalla después de ingresado el nuevo proyecto.



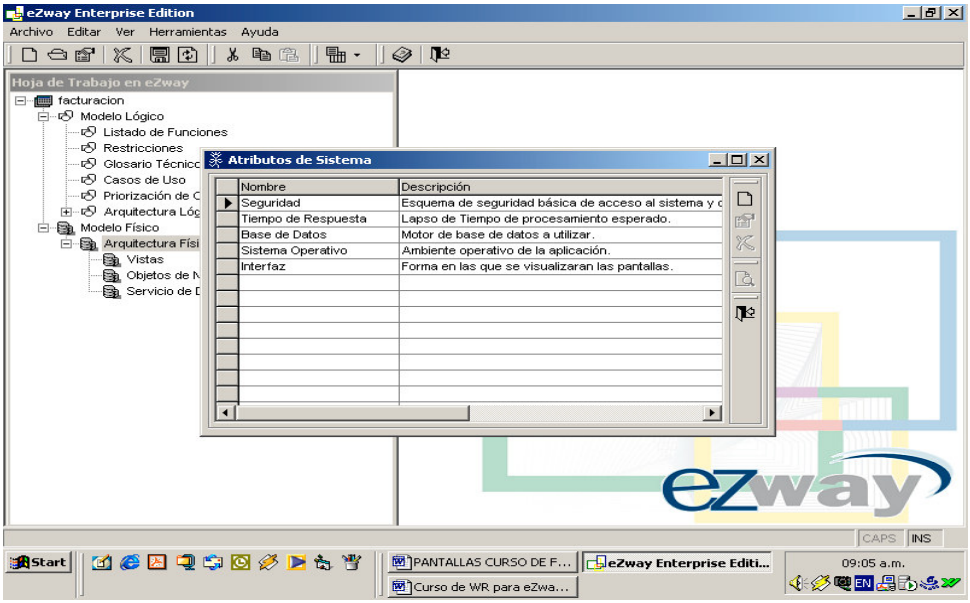
Fuente. **eZway Enterprise Edition**

5.10 Estructura eZway

Tabla VII. Muestra herramientas y atributos del sistema

Atributo	Descripción
Sistema Operativo	Ambiente operativo de la aplicación.
Tiempo de Respuesta	Lapso de Tiempo de procesamiento esperado.
Interfaz	Forma en las que se visualizaran las pantallas.
Plataforma	Plataforma sobre la cual se ejecuta la aplicación.
Base de Datos	Motor de base de datos a utilizar.
Seguridad	Esquema de seguridad básica de acceso al sistema y control de la información.

Figura 31. Muestra ingreso y actualización de atributos del sistema del proyecto a desarrollar

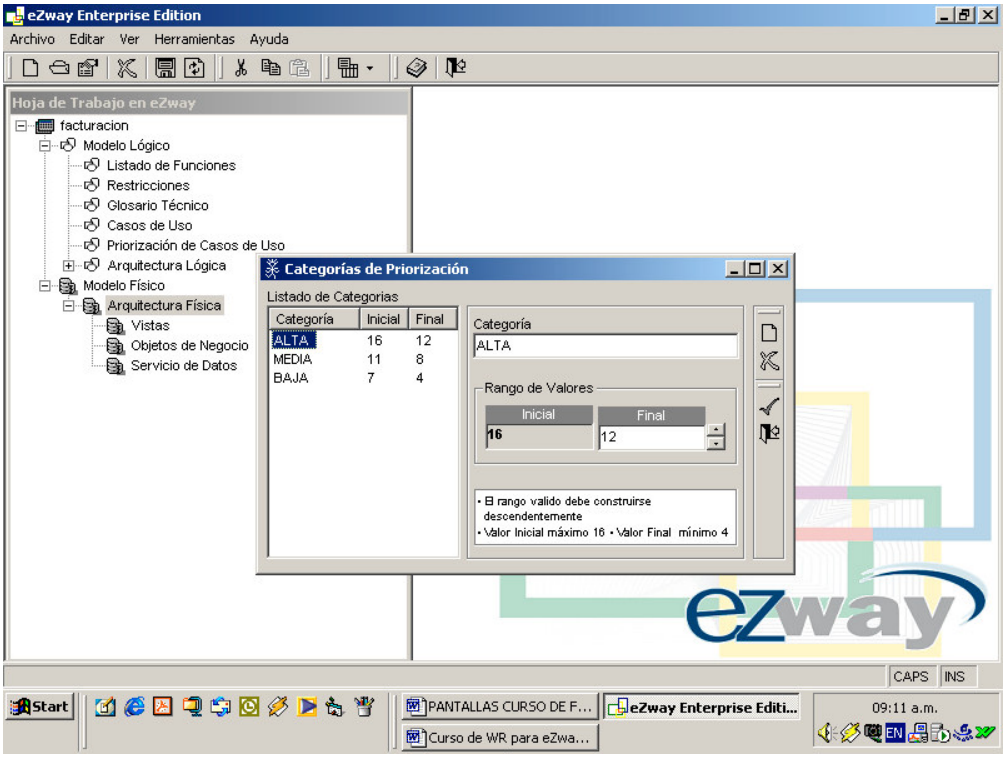


Fuente. **eZway Enterprise Edition**

Tabla VIII. Muestra categorías de priorización de actividades

Categoría	Valor inicial	Valor final
Alta	16	12
Media	11	8
Baja	7	4

Figura 32. Ingreso y actualización de categorías de priorización del sistema a desarrollar



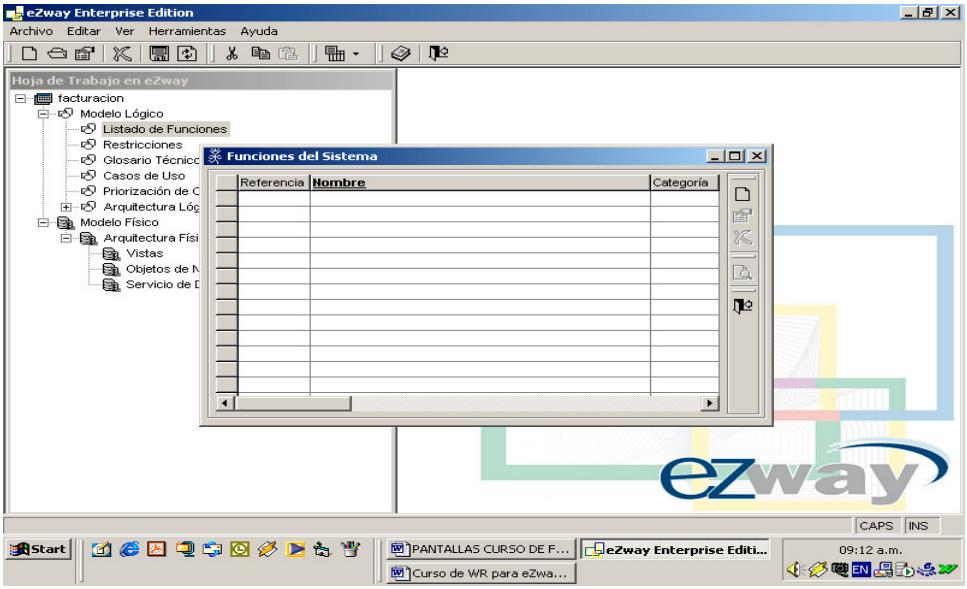
Fuente. **eZway Enterprise Edition**

5.10.1 Modelo lógico

Tabla IX. Muestra lista de funciones

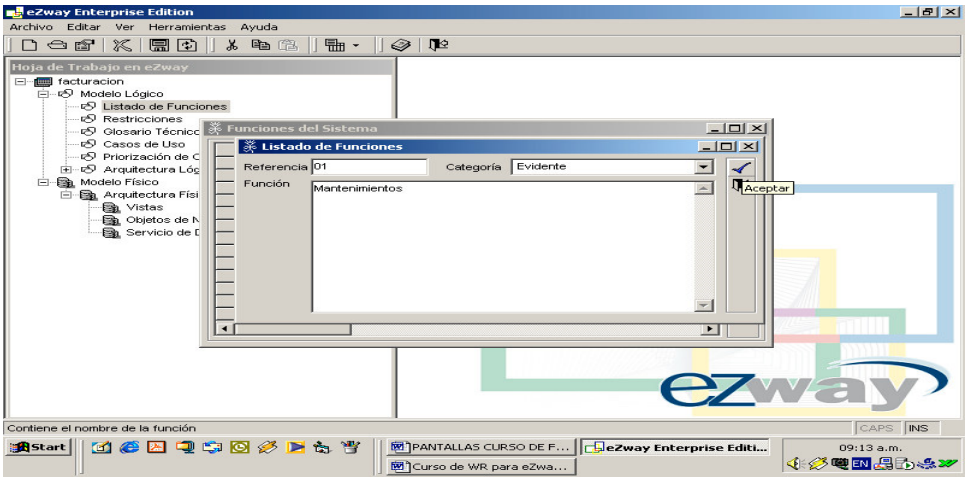
01	Mantenimientos	Evidente
01.01	IVA	Evidente
01.02	Marca	Evidente
01.03	Familia	Evidente
01.04	Tipo de documento	Evidente
01.05	Tipo de pago	Evidente
01.06	Cliente	Evidente
02	Mantenimiento maestros poblados	Evidente
02.01	Productos	Evidente
03	Mantenimiento de transacciones	Evidente
03.01	Encabezado de factura	Evidente
03.02	Detalle de factura	Evidente
03.02.01	Calculo total de factura	Oculto
03.02.02	Calculo saldo	Oculto
03.03	Detalle de pago	Evidente
03.03.01	Calculo total del pago	Oculto
03.03.02	Calculo de saldo	Oculto
04	Reportes	Evidente
05	Otros	Evidente

Figura 33. Muestra ingreso y actualización de funciones del proyecto a desarrollar



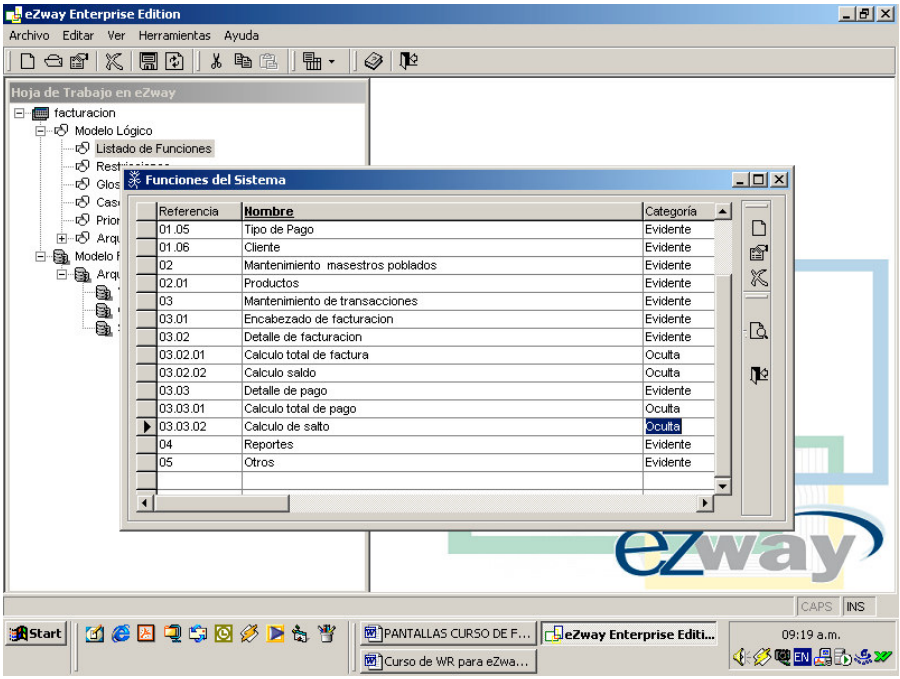
Fuente. **eZway Enterprise Edition**

Figura 34. Muestra pantalla de actualización e ingreso del listado de funciones



Fuente. **eZway Enterprise Edition**

Figura 35. Muestra lista de funciones ingresadas del nuevo proyecto

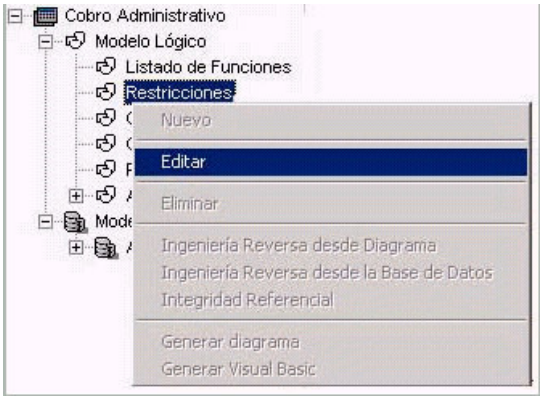


Fuente. **eZway Enterprise Edition**

Tabla X. Muestra restricciones

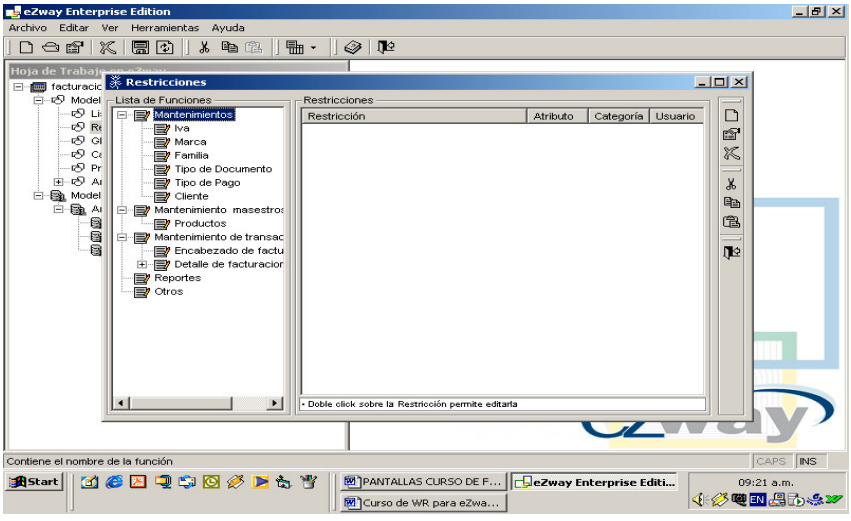
Función	Ref.	Atributo	Categoría	Descripción
01	1	Sistema Operativo	Opcional	Windows 2000
01	2	Seguridad	Obligatorio	Administrador y Supervisor
02	1	Seguridad	Obligatorio	Administrador y Supervisor
03	1	Tiempo de Respuesta	Obligatorio	No mayor a 2 segundos
03	2	Seguridad	Obligatorio	Cajero
04	1	Plataforma	Opcional	Internet Informatión Server

Figura 36. Muestra pantalla de ingreso y actualización de restricciones



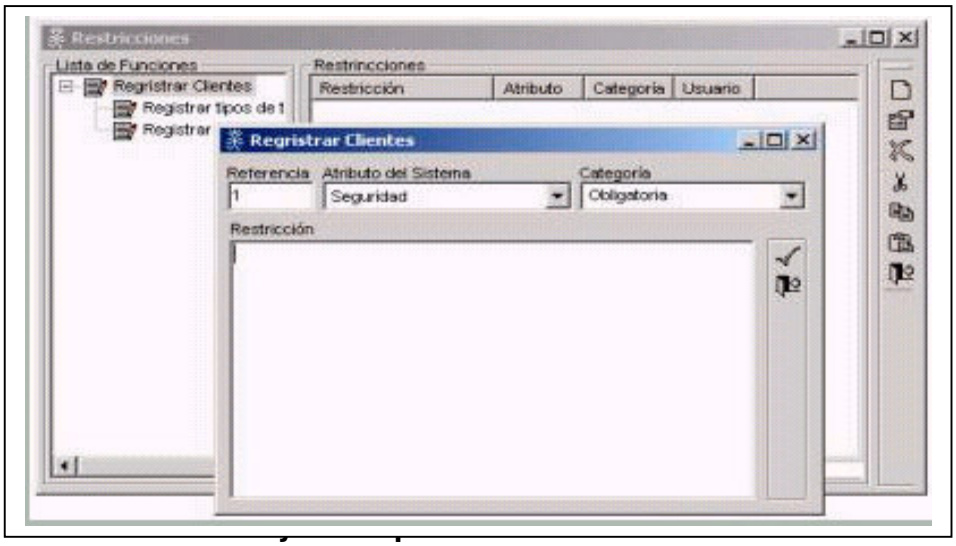
Fuente. **eZway Enterprise Edition**

Figura 37. Muestra pantalla referente a la definición de restricciones



Fuente. **eZway Enterprise Edition**

Figura 38. Pantalla de ingreso de restricciones

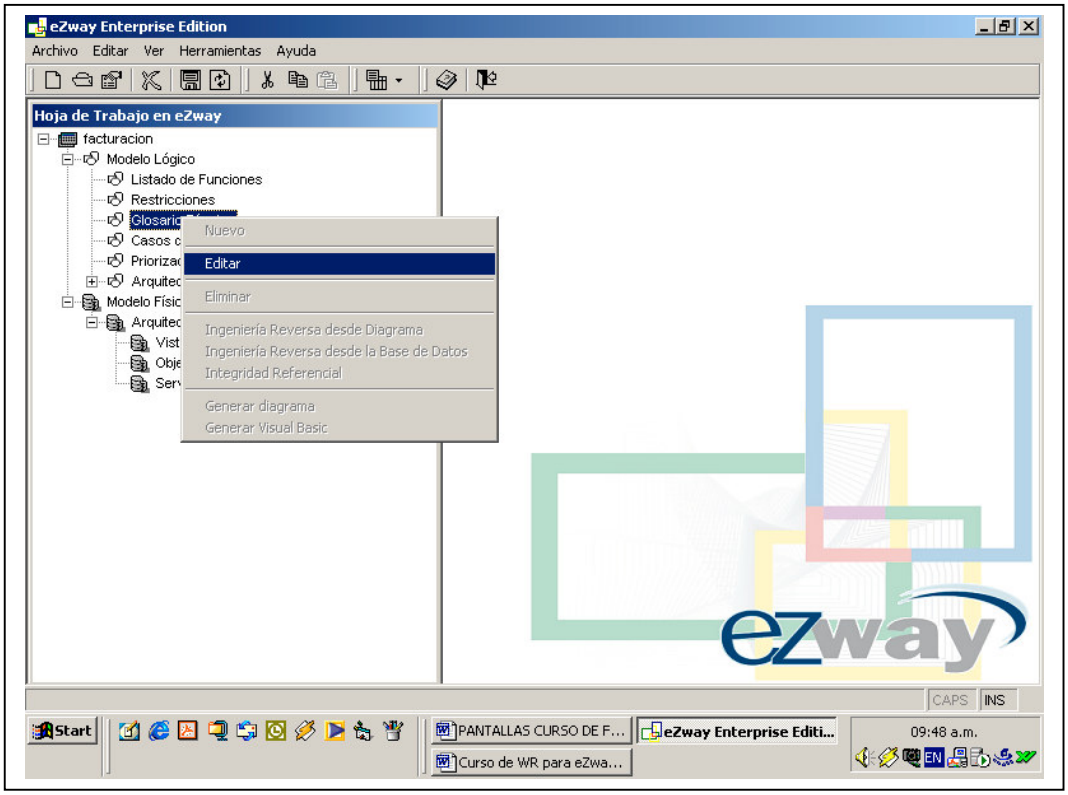


En el glosario técnico se ingresan términos y su definición respectiva para su posterior consulta en la documentación del proyecto.

Tabla XI. Muestra glosario técnico

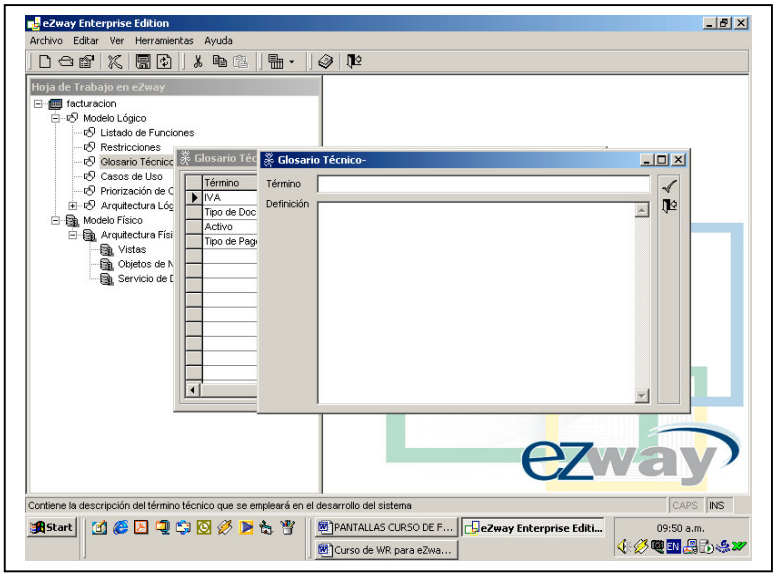
Término	Definición
Tipo de Documento	Tipo de factura (computarizada, manual, etc.)
Tipo de Pago	Forma de abonar una factura (efectivo, cheque, etc.)
IVA	Porcentaje del Impuesto al Valor Agregado
Activo	Indica si esta habilitado o no determinado registro

Figura 39. Muestra pantalla de ingreso de definición de términos para el glosario técnico.



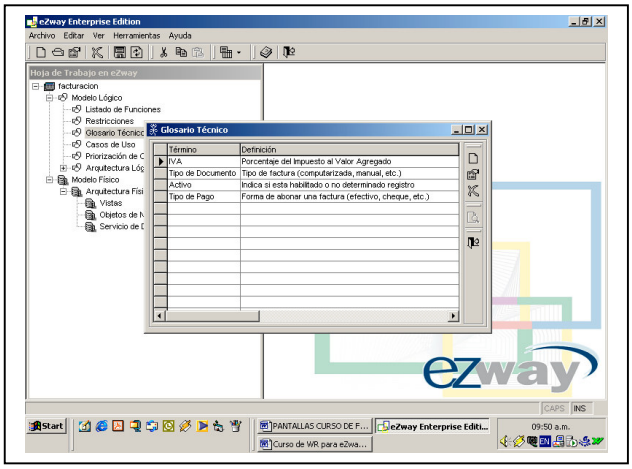
Fuente. **eZway Enterprise Edition**

Figura 40. Muestra pantalla de ingreso de términos para el glosario.



Fuente. **eZway Enterprise Edition**

Figura 41. Muestra pantalla de términos ya ingresados.

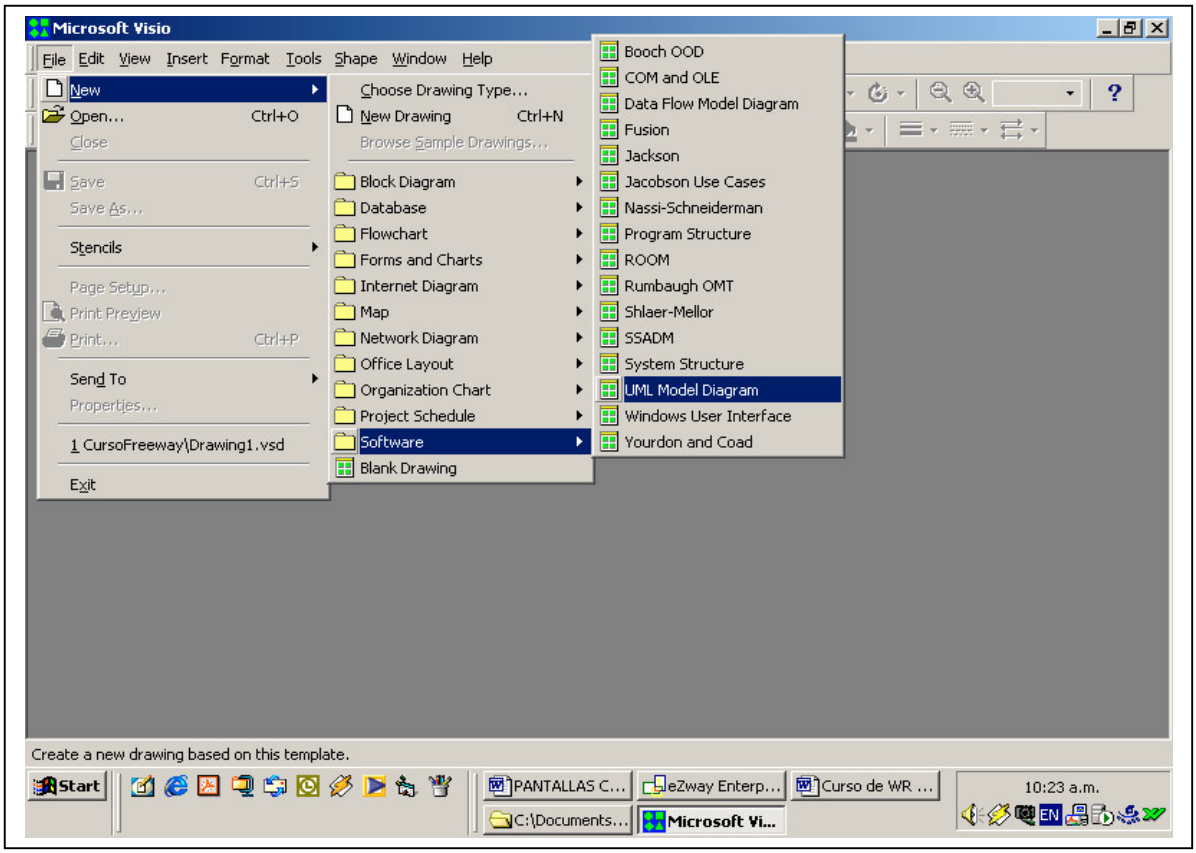


Fuente. **eZway Enterprise Edition**

5.11 Visio

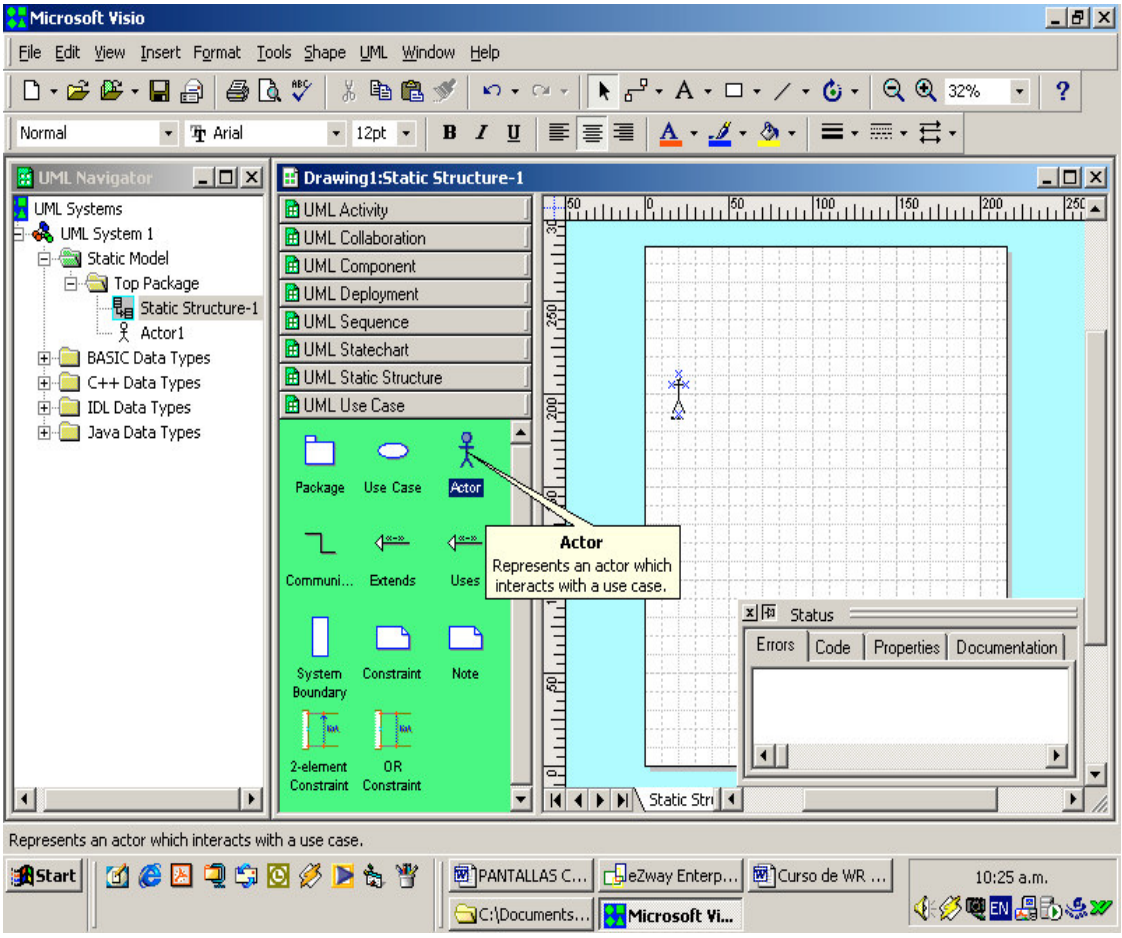
5.11.1 Diagrama UML

Figura 42. Muestra pantalla principal de Microsoft visio, para realizar el diagrama UML



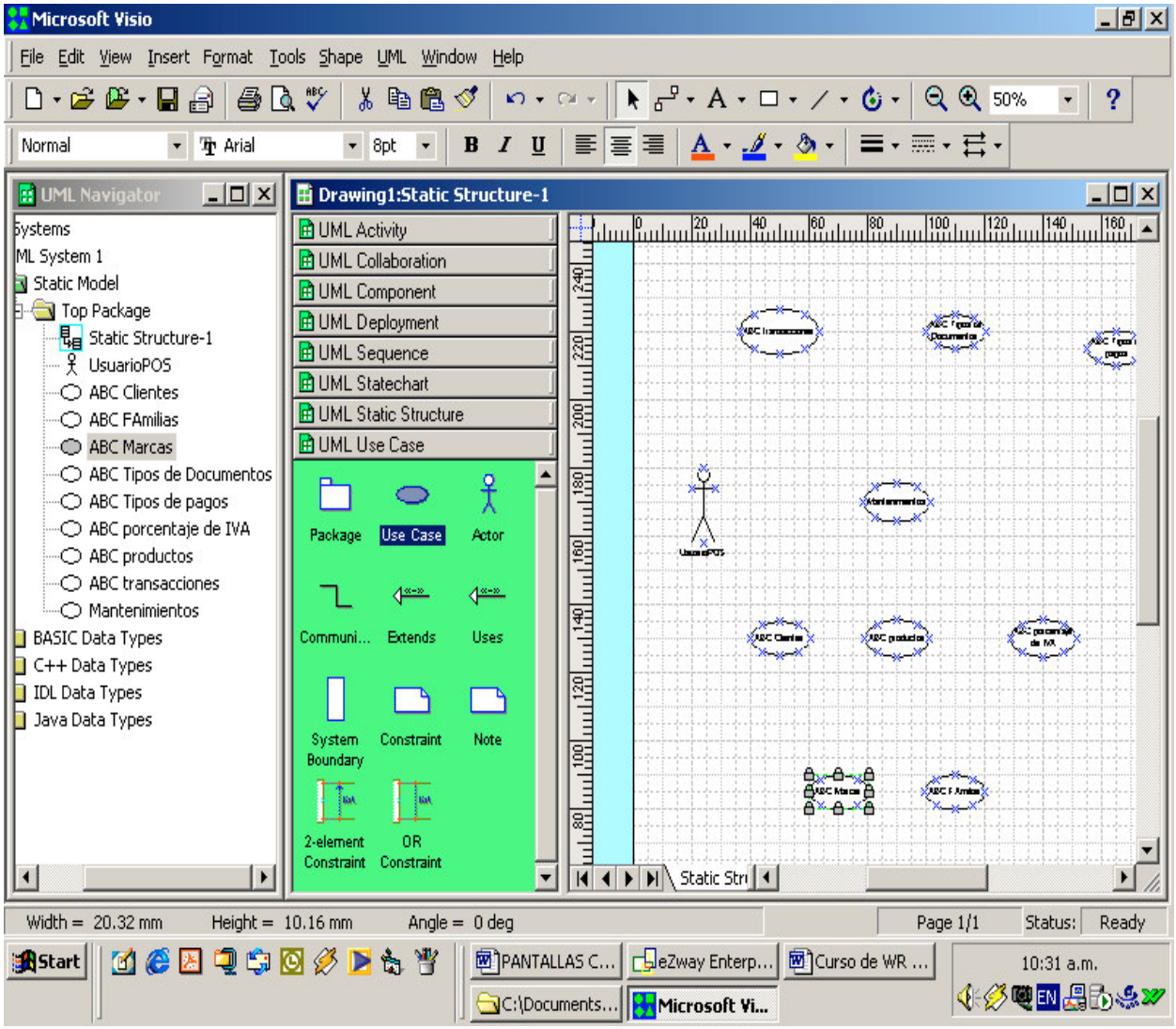
Fuente. **Microsoft Visio Professional**

Figura 43. Muestra pantalla para realizar diagrama UML



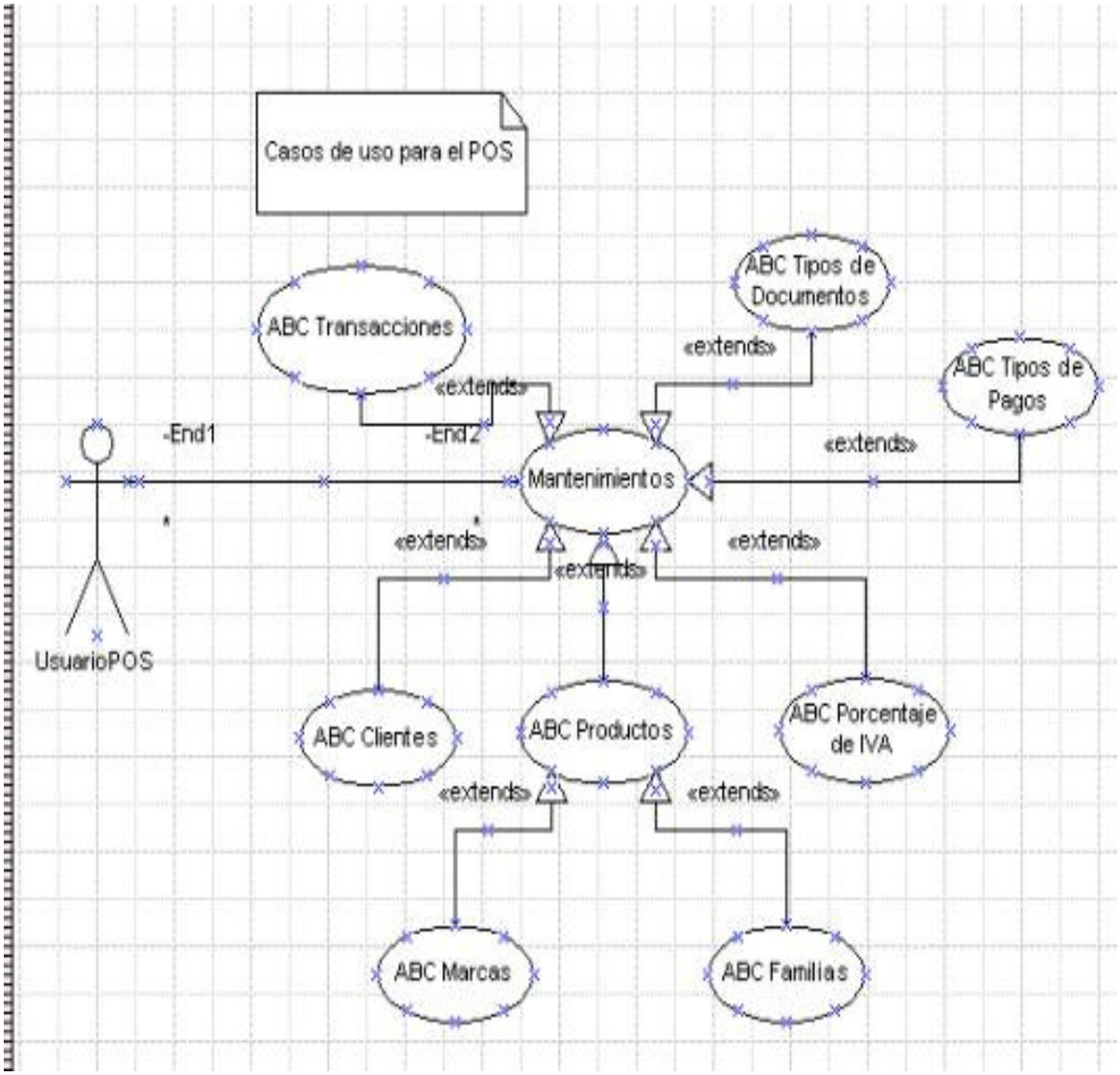
Fuente. **Microsoft Visio Professional**

Figura 44. Muestra pantalla con casos de uso.



Fuente. **Microsoft Visio Professional**

Figura 45. Muestra pantalla de realización de relaciones con caso de usos

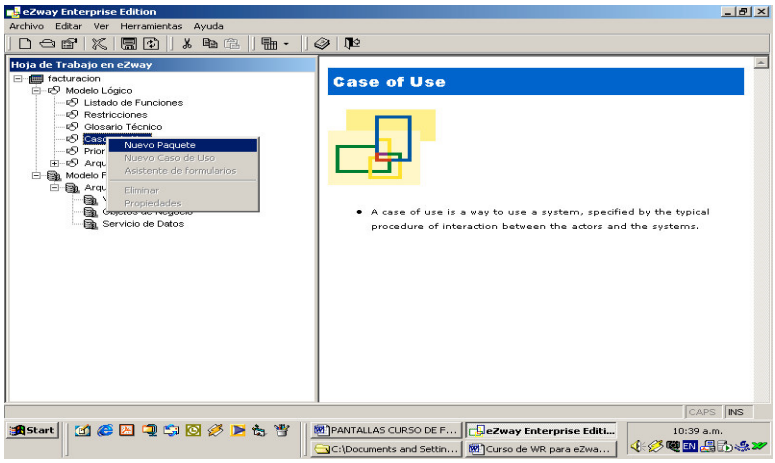


Fuente. **Microsoft Visio Professional**

5.12 Casos de uso

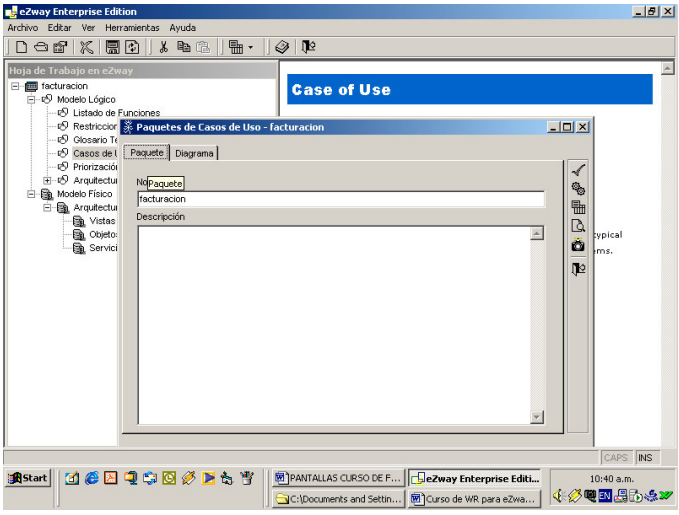
5.12.1 Nuevo paquete (Facturación)

Figura 46. Muestra pantalla inicial de obtención de caso de uso realizado en Visio.



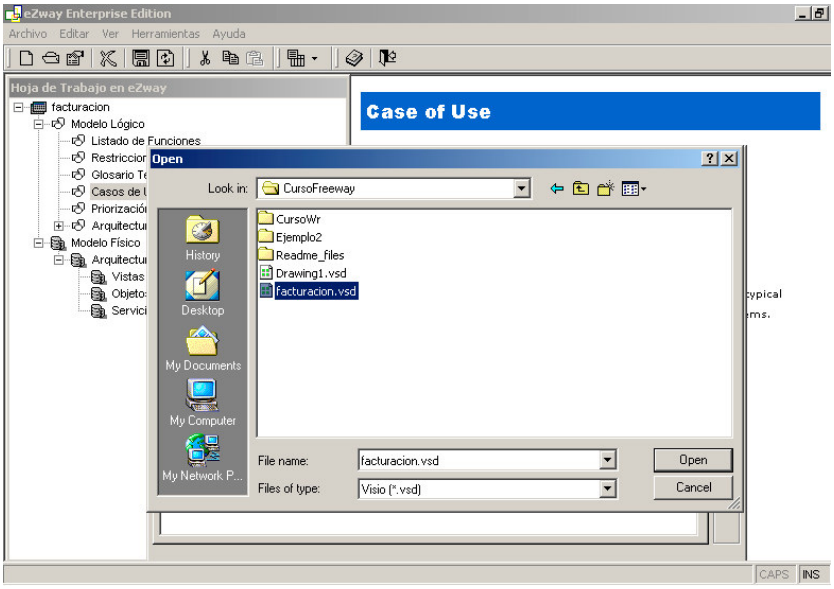
Fuente. **eZway Enterprise Edition**

Figura 47. Muestra ingreso del paquete de casos de uso -Facturación



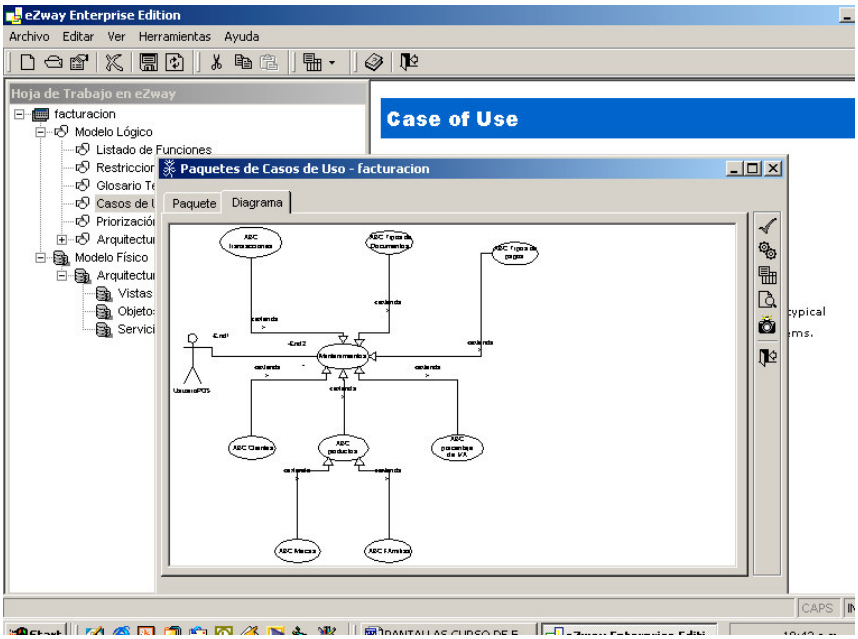
Fuente. **eZway Enterprise Edition**

Figura 48. Muestra pantalla de búsqueda del diagrama UML



Fuente. **eZway Enterprise Edition**

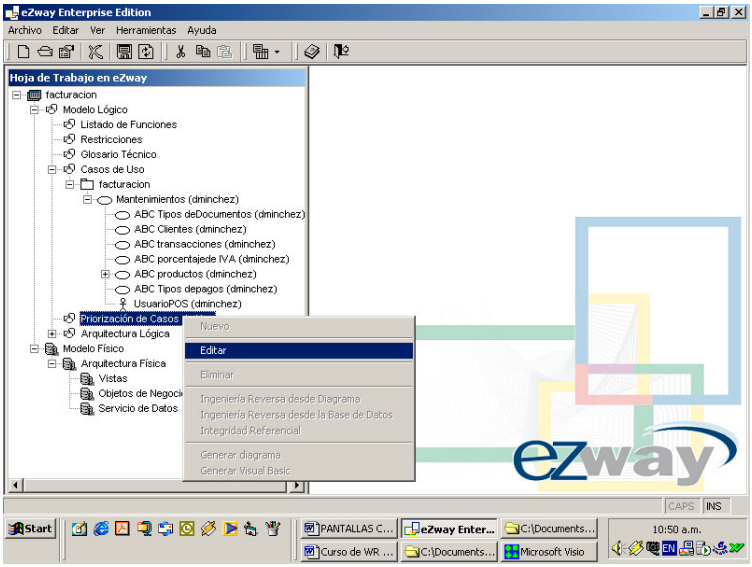
Figura 49. Muestra caso de uso presentado en eZway



Fuente. **eZway Enterprise Edition**

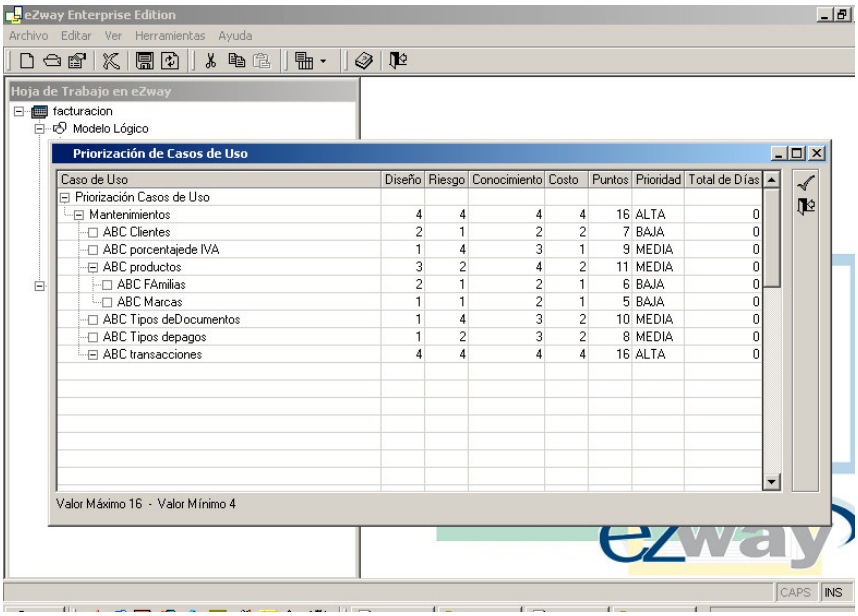
5.12.2 Priorización de casos de uso

Figura 50. Muestra ingreso y actualización de priorización de casos de uso



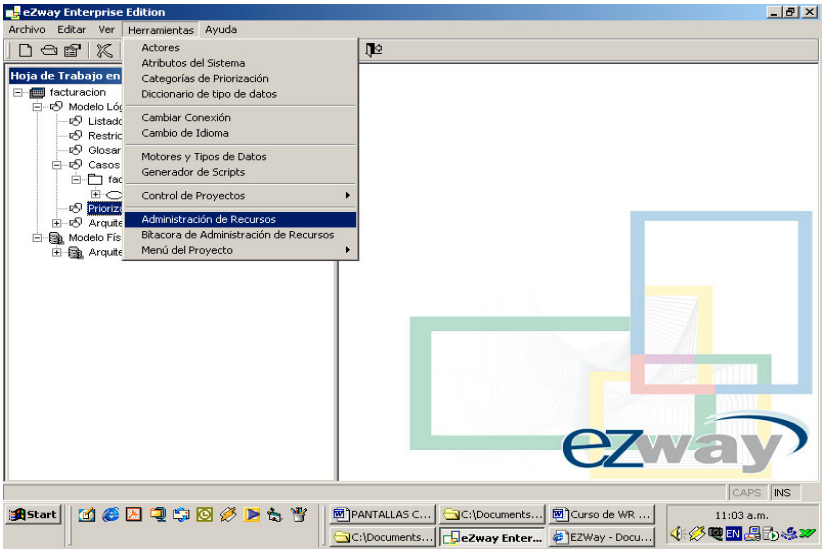
Fuente. **eZway Enterprise Edition**

Figura 51. Muestra casos de uso ingresados para el Nuevo proyecto



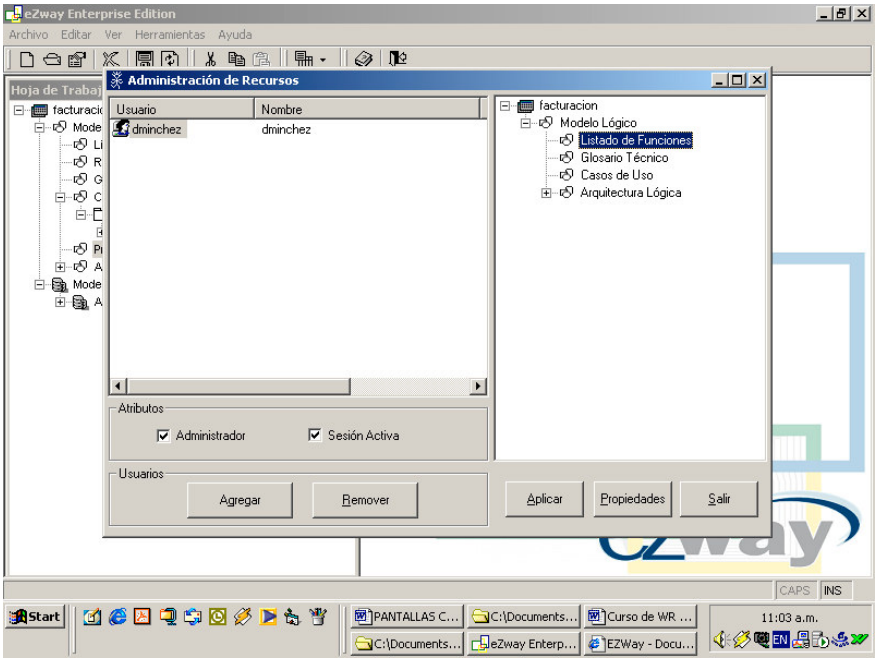
Fuente. **eZway Enterprise Edition**

Figura 52. Muestra ventana principal de administración de recursos



Fuente. **eZway Enterprise Edition**

Figura 53. Muestra usuario y los recursos que puede administrar

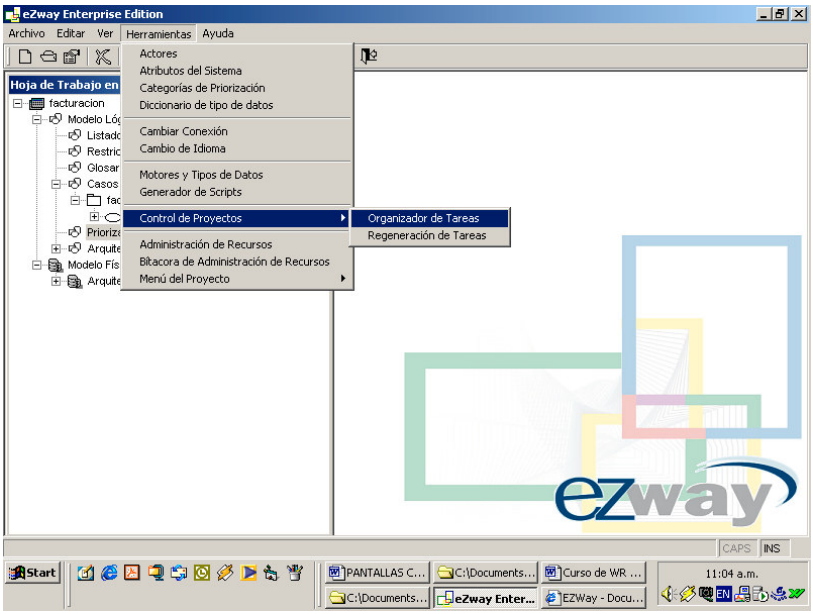


5.13.1 Control de proyectos

a. Organizador de tareas (simple)

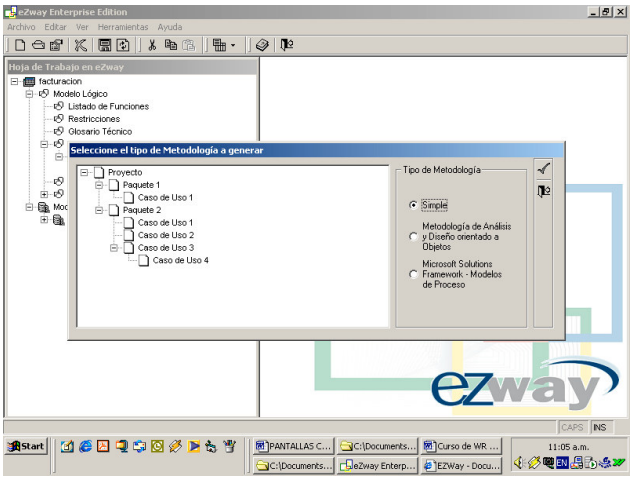
Permite organizar casos de uso priorizados.

Figura 54. Muestra antalla de búsqueda del organizador de tareas



Fuente. **eZway Enterprise Edition**

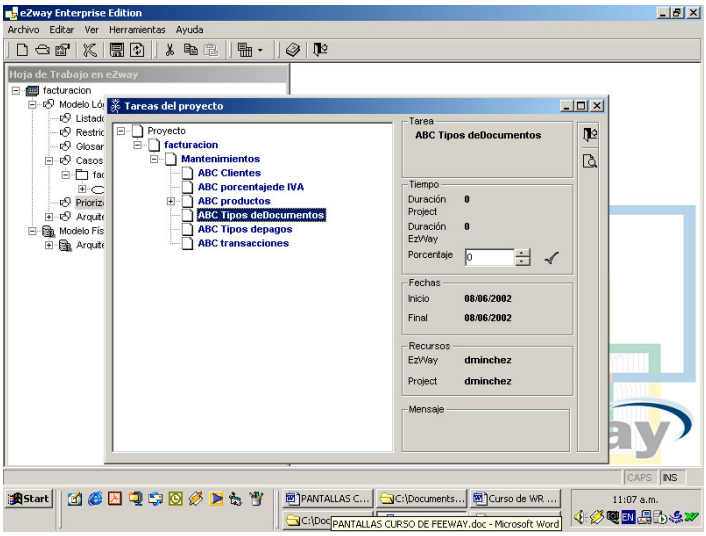
Figura 55. Muestra selección del tipo de organización a generar



Fuente. **eZway Enterprise Edition**

b. Regenerar tareas

Figura 56. Muestra vista del proyecto generado

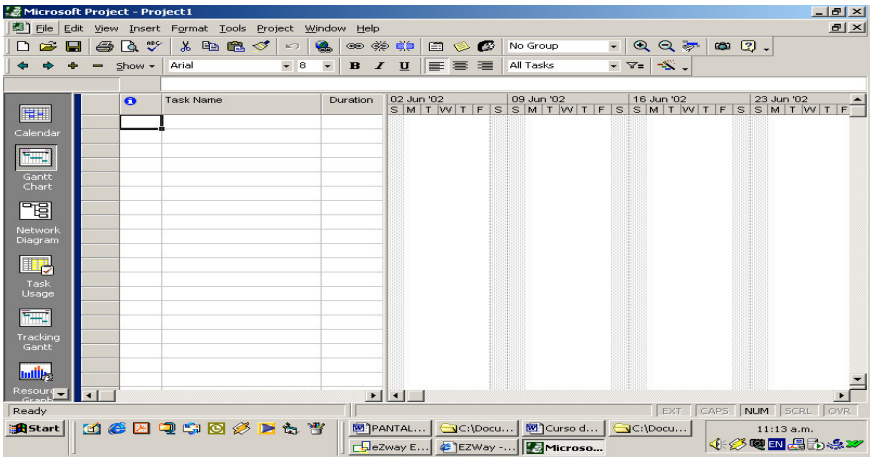


Fuente. **eZway Enterprise Edition**

5.14 MS *Proyect* Professional

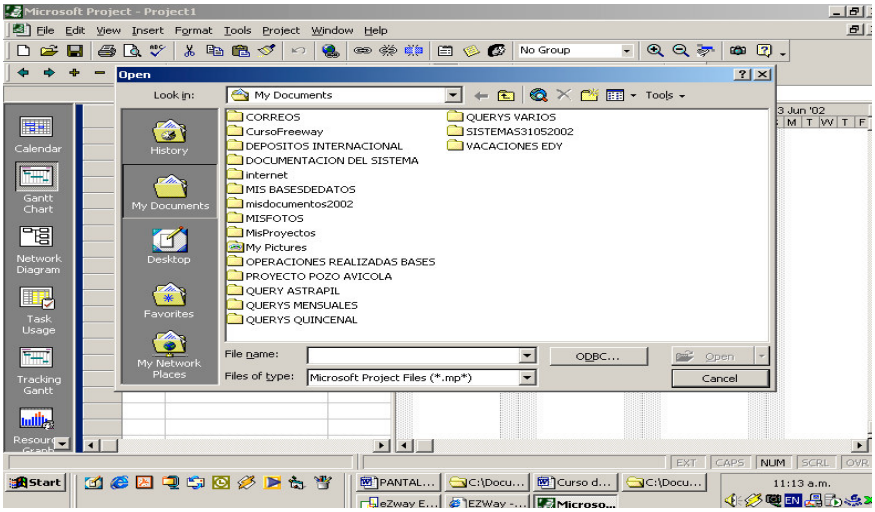
5.14.1 Abrir el proyecto (ODBC-> Freeway)

Figura 57. Muestra pantalla principal de project



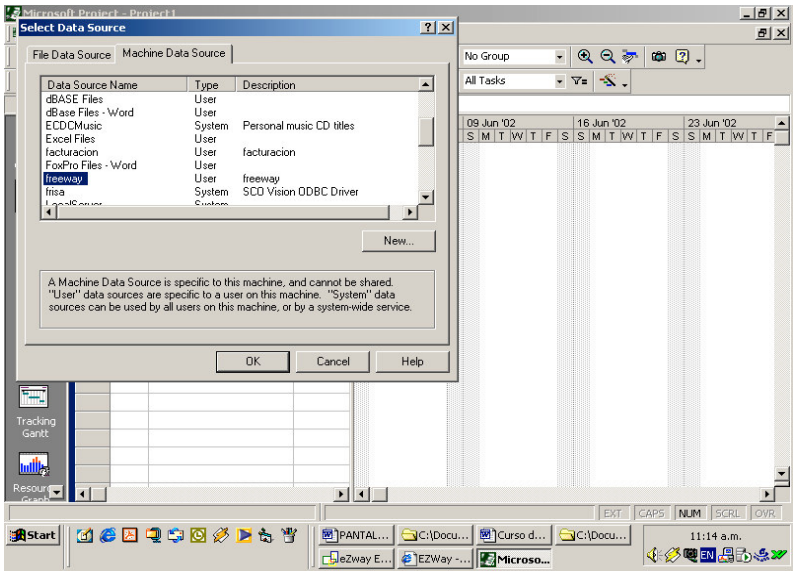
Fuente. **Microsoft Project Manager**

Figura 58. Muestra pantalla de búsqueda de la apertura del proyecto



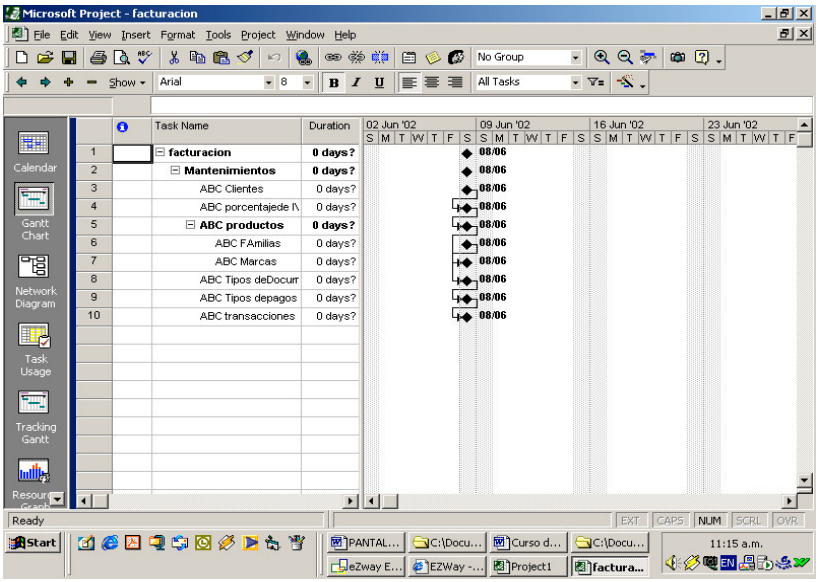
Fuente. **Microsoft Project Manager**

Figura 59. Muestra pertura del proyecto ODBC FREEWAY



Fuente. **Microsoft Project Manager**

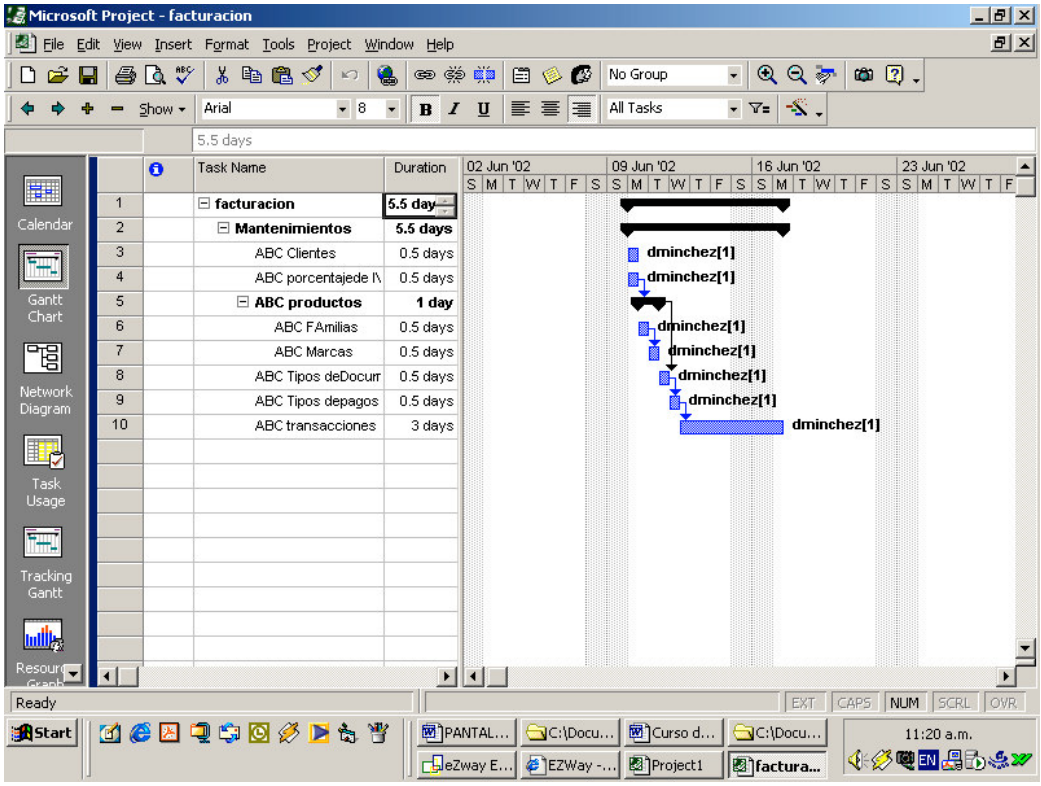
Figura 60. Muestra pantalla del proyecto ya abierto en Project



Fuente. **Microsoft Project Manager**

5.14.2 Modificar Días (Ver resultado en eZway->Herramientas->Control de proyectos->Org. Tareas)

Figura 61. Muestra pantalla de modificación de días.



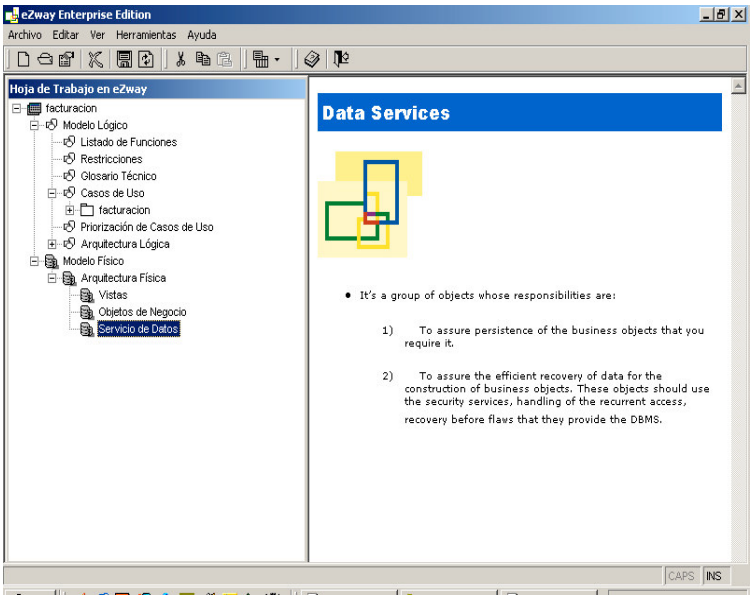
Fuente. **Microsoft Project Manager**

5.15 Modelo lógico

5.15.1 Arquitectura lógica

En el servicio de datos es donde reside la lógica de acceso a la base de datos.

Figura 62. Muestra pantalla de búsqueda de la opción servicio de datos

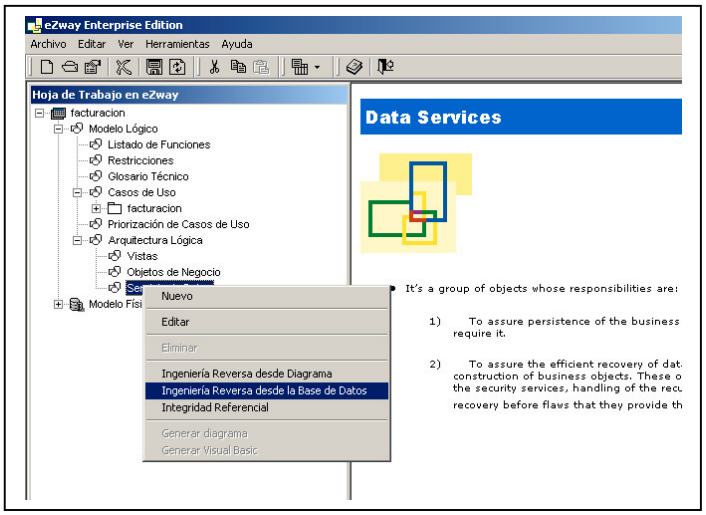


Fuente. **eZway Enterprise Edition**

5.15.2 Ingeniería en reversa

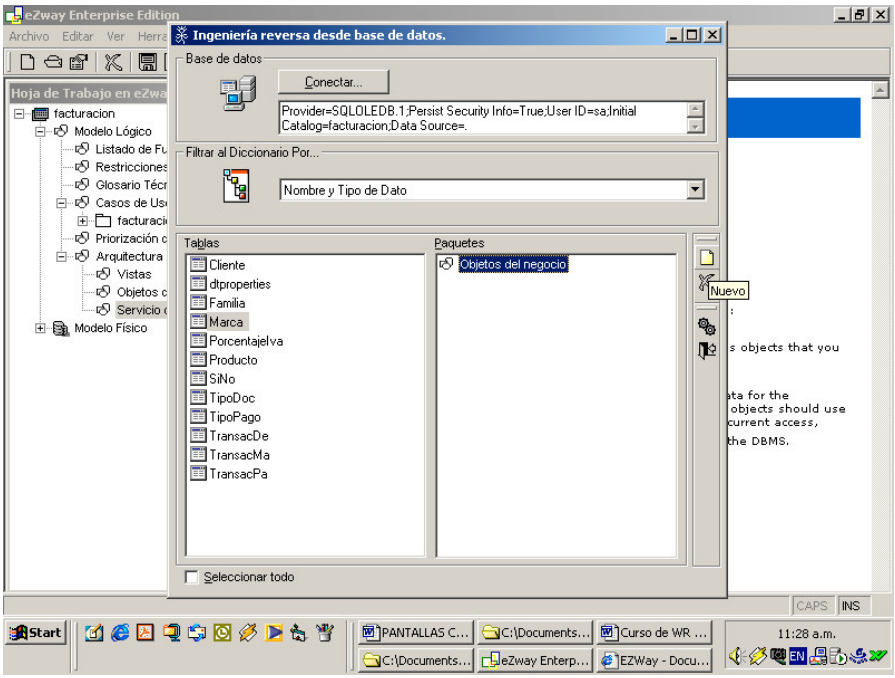
Realiza la ingeniería en reversa del ODBC seleccionado en este caso facturación.

Figura 63. Muestra pantalla de la opción de ingeniería en reversa



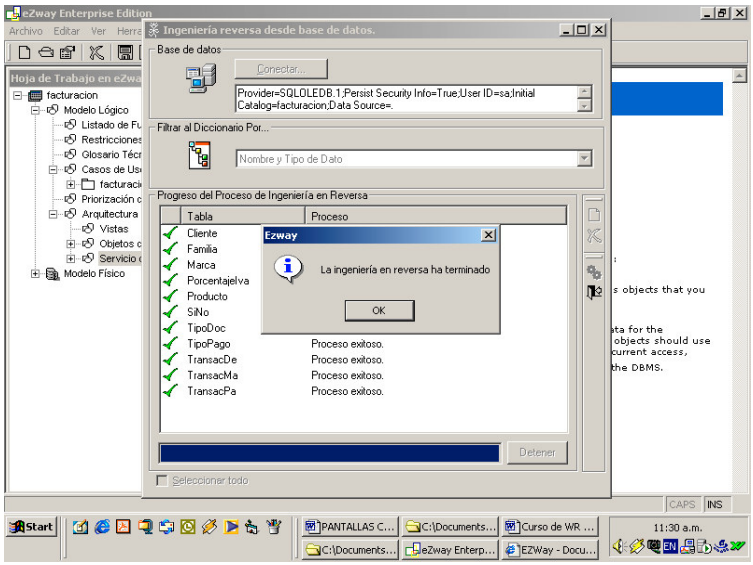
Fuente. **eZway Enterprise Edition**

Figura 64. Muestra pantalla de conexión y obtención de las tablas del proyecto realizado.



Fuente. **eZway Enterprise Edition**

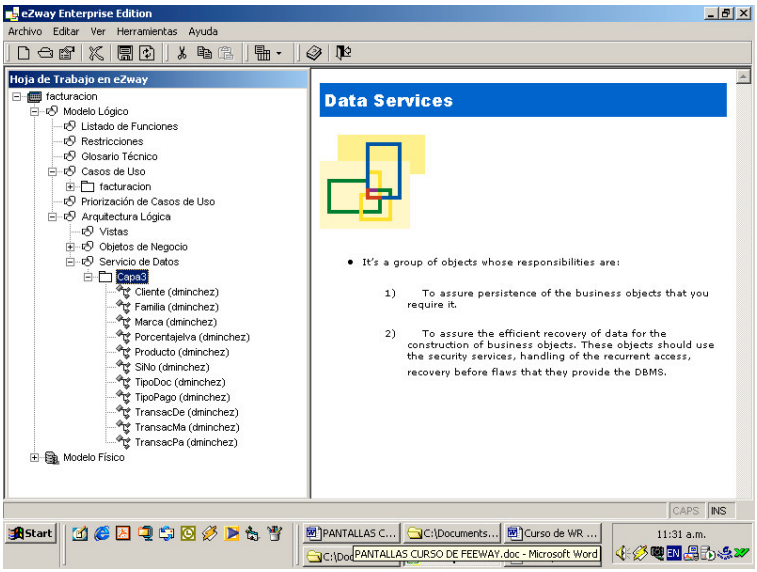
Figura 65. Muestra pantalla donde se indica que la ingeniería en reversa fue generada



Fuente. **eZway Enterprise Edition**

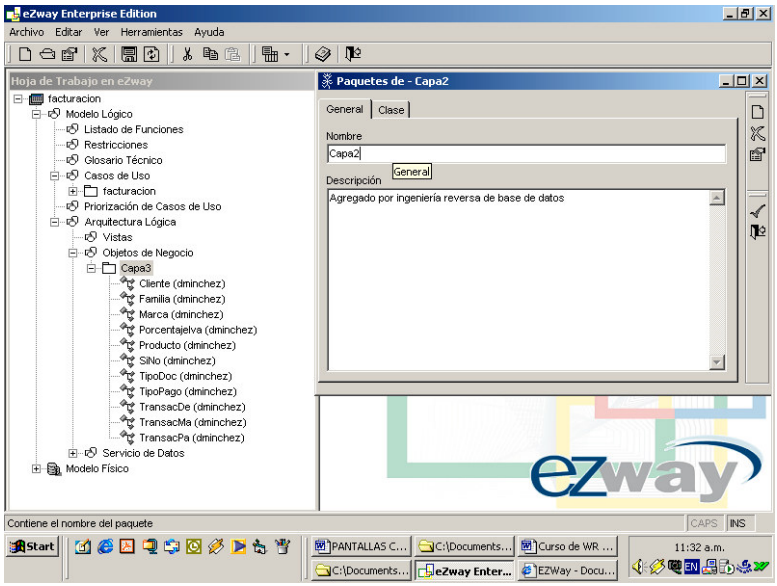
5.15.3 En Objetos de Negocios renombrar capa3 por capa2

Figura 66. Muestra pantalla donde se muestra el proceso de renombrar



Fuente. **eZway Enterprise Edition**

Figura 67. Muestra pantalla donde se realiza el renombramiento de capa3 por capa2



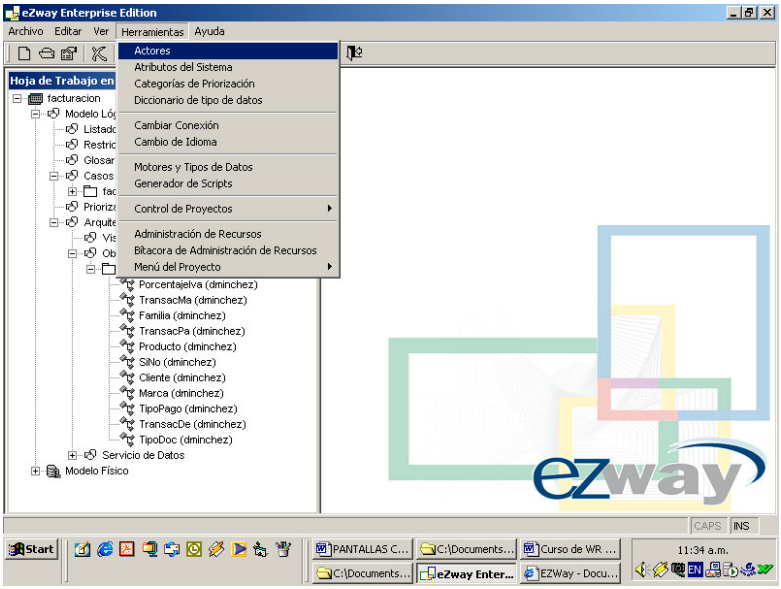
Fuente. **eZway Enterprise Edition**

5.16 Herramientas

5.16.1 Actores

Para poder visualizar a todos los actores involucrados en la aplicación, para efectos de mantenimiento, debe realizar lo siguiente.

Figura 68. Muestra pantalla donde se localiza la opción actores

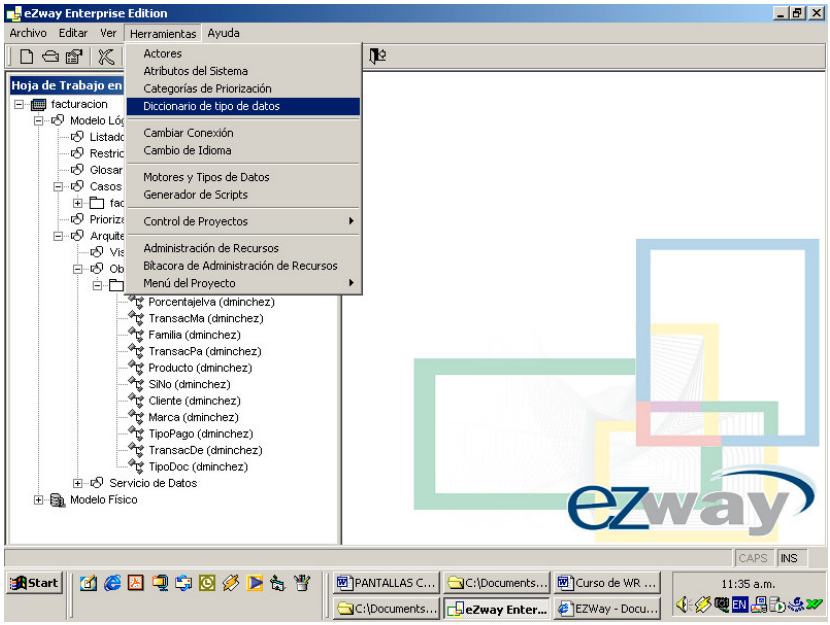


Fuente. **eZway Enterprise Edition**

5.17 Diccionario de tipos de datos

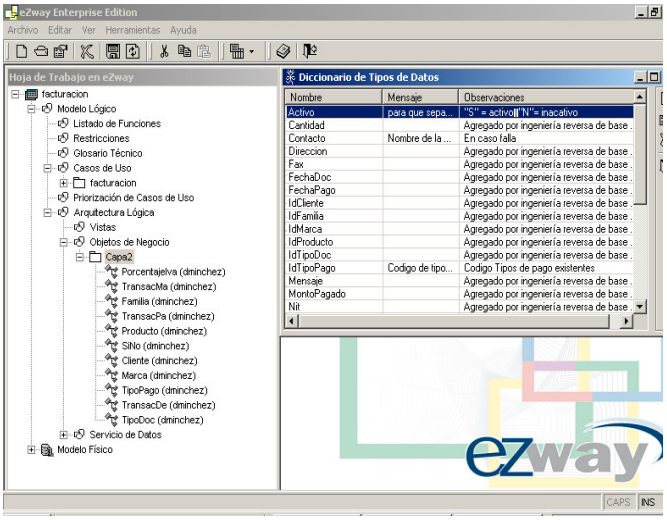
En esta parte se ingresan los mensajes y/o observaciones para el pie de las formas.

Figura 69. Muestra pantalla donde se selecciona la opción diccionario de tipo de datos



Fuente. **eZway Enterprise Edition**

Figura 70. Muestra pantalla donde realizamos el ingreso y actualización del diccionario de datos

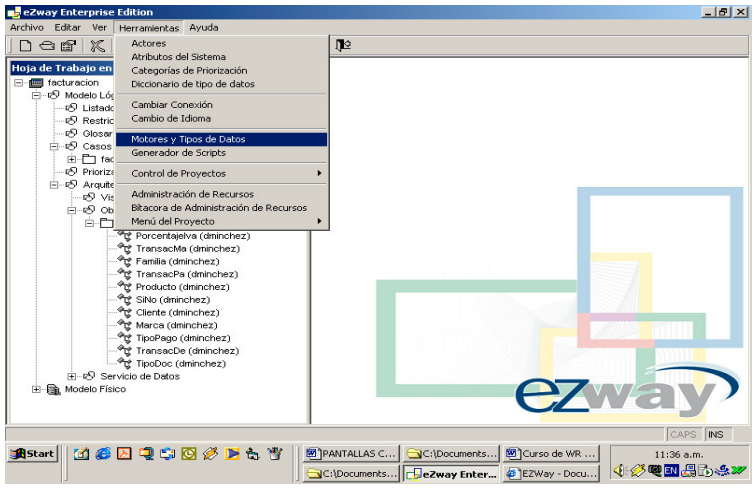


Fuente. **eZway Enterprise Edition**

5.18 Motores y tipos de datos

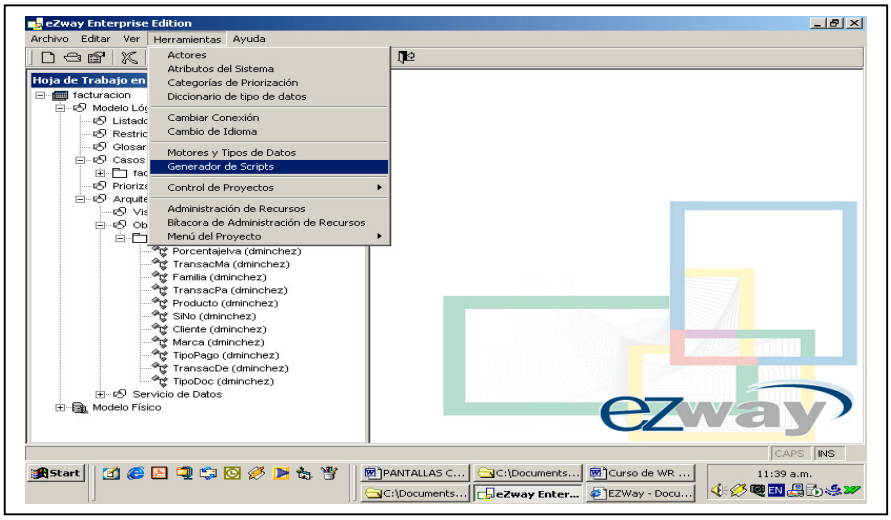
La finalidad es poder extender eZway en el manejo de motores de base de datos para generar los scripts para la creación de tablas, índices y relaciones a la base de datos y poder realizar ingeniería reversa desde cualquiera de los nuevos motores añadidos por el usuario de eZway.

Figura 71. Muestra motores y tipos de datos



Fuente. **eZway Enterprise Edition**

Figura 72. Muestra pantalla de opción generación de Scripts



Fuente. **eZway Enterprise Edition**

5.19 Creación de formularios manualmente

5.19.1 Crea formulario de TIPODOC

Me.Name: frmTipoDoc
Me.BorderStyle: 1-Fixed Single
Me.Caption: Mantenimiento de Tipos de Documentos
Font: Courrier New de 8
Maxlength: (Longitud de cada campo, según el Diagrama de E-R)
Width: (MaxLength + 1) * 105
Nombrar cada Control en la Forma: p.e. Campo=IdTipoDoc Name=txtIdTipoDoc Text=txtIdTipoDoc
DataViewerShow: Muestra el Grid;1-On
DataViewerWidth: Ancho del Grid; 3500 – 5000
DataViewerFields: Campos que mostrará el Grid; "*" o "IdTipoDoc,NomDoc"

5.19.2 Controles para VB

5.19.2.1 FORMBASE

DataViewer
DataViewerShow (1-vbOn)
DataViewerWidth (3500)
DataViewerCom_II (.TipoDoc)
DataViewerFields (IdTipoDoc, NomDocto) o (*)
DataViewerFieldsWidth (500,3000)
DataViewerOrder (NomDocto)

5.19.2.2 Otros controles

TxtCharacter

ValueRequired

TxtNumeric

NotNegative

TypeFormat

EditBox

TxtDate

MaxDate

MinDate

ValidDayDown (formato de ingreso: Normal, Calculadora)

ValidDayUp (requerido)

TxtTime

5.19.2.3 Funcionamiento del FORMBASE

- Clic derecho en del DataViewer
- Filtrado de información
- Doble clic en algún control
- Icono en esquina inferior derecha del FormBase

5.19.3 Crear formularios híbridos

- TipoPago
- TipoDoc
- PorcentajeIva
- Marca
- Familia
- Cliente (Incluir código personalizado para el Combo de cmbActivo)
- Definir Funcionalidad de Controles
  - Origen de datos: Row Source
  - Campo Oculto: BoundColumn-Valor
  - Campo Visible: ListField-Mensaje
- Código personalizado
  - FrmBase_NewClick

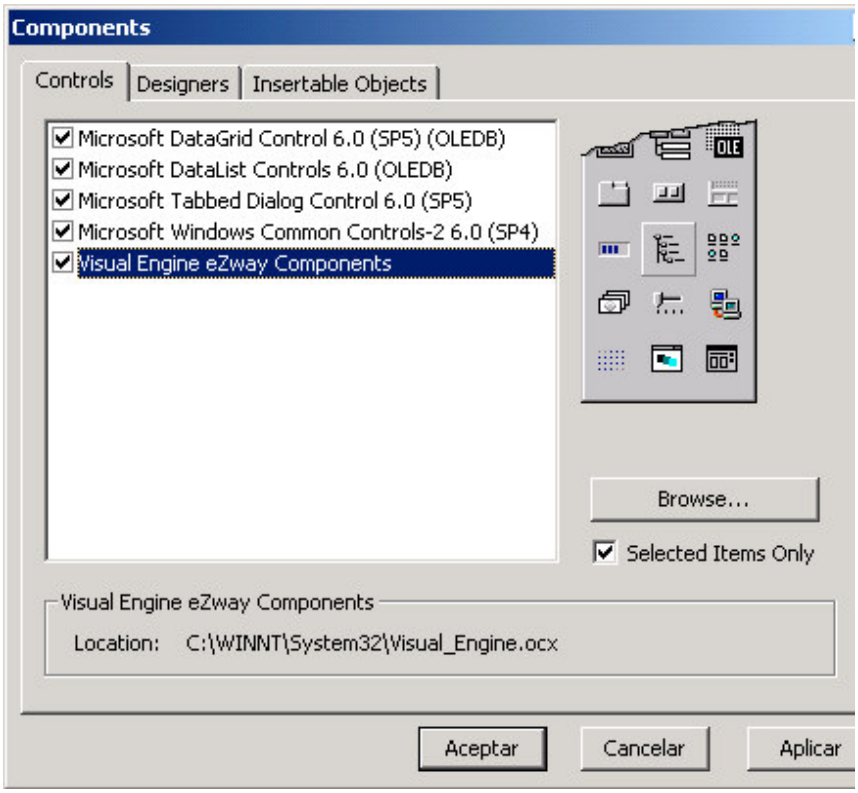
    txtActivo.BoundText = "S"
- Producto (Incluir código personalizado para el Combo de cmbActivo).

5.19.4 Crear formulario manual de transacciones

5.19.4.1 Referencias

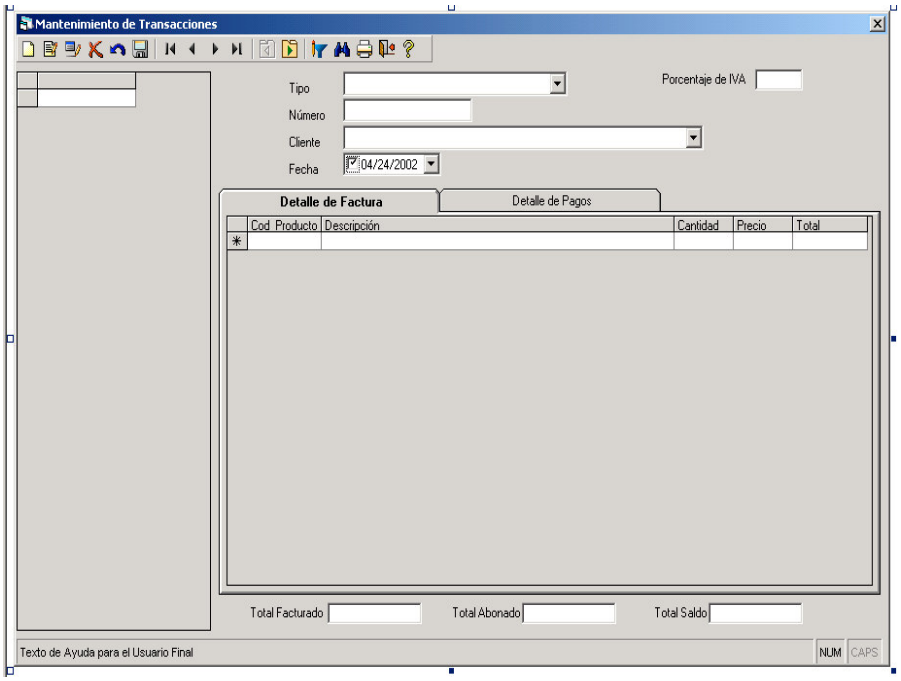
A continuación se define las referencias a utilizar en *visual basic*.

Figura 73. Muestra pantalla donde seleccionamos los componentes



Fuente. **Microsoft Visual Basic**

Figura 74. Muestra vista de la forma Mantenimiento de transacciones realizada por eZway



Fuente. **eZway Enterprise Edition**

Propiedad de FormBase: DateFormat(mm/dd/yyyy)
DataViewerFields(IdTipoDoc, NumDoc)
DataViewerFieldsAs(Tipo, Número)

Propiedad de DtPicker: CheckBox(True)

5.19.5 Detalle de factura

- GrdIdTipoDoc_ oculto
- GrdNumDoc_ oculto
- Cod_Producto_
- Descripción_
- Cantidad_
- GrdTotalSinIva_ oculto
- Precio_
- Total_

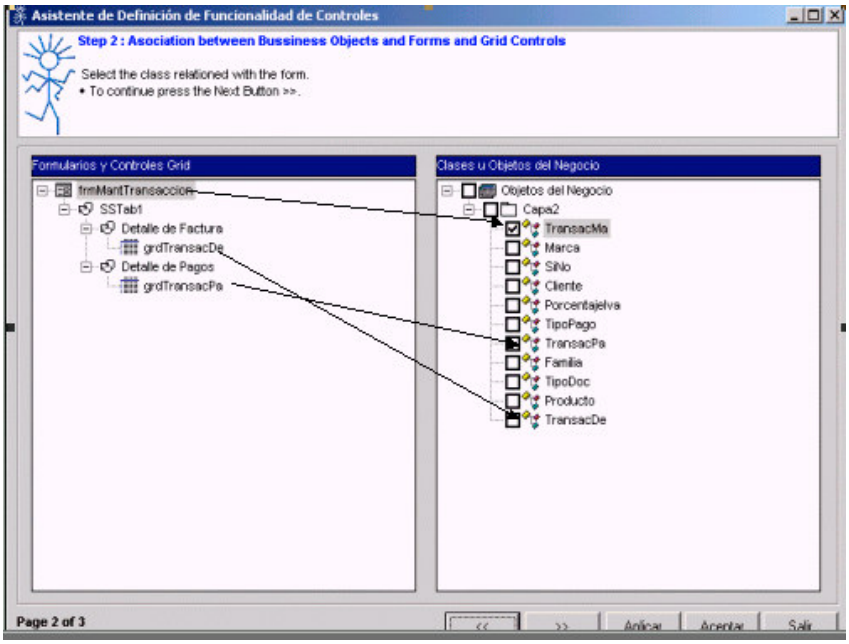
5.19.6 Detalle de pago

- GrdIdTipoDoc oculto
- GrdNumDoc oculto
- Tipo_Pago
- Descripción del Pago
- No.Documento
- Fecha
- Monto

5.19.7 Definir funcionalidad de controles

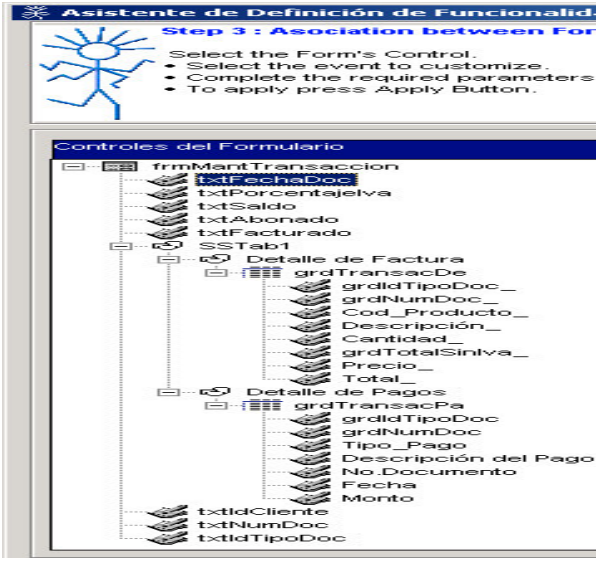
El poblado de controles de tipo Grid se lleva a cabo mediante el siguiente proceso:

Figura 75. Muestra pantalla del asistente de definición de funcionalidad de controles



Fuente. **eZway Enterprise Edition**

Figura 76. Valores que debe ingresar



FechaDoc,O,Value
None,X
None,X
None,X
None,X

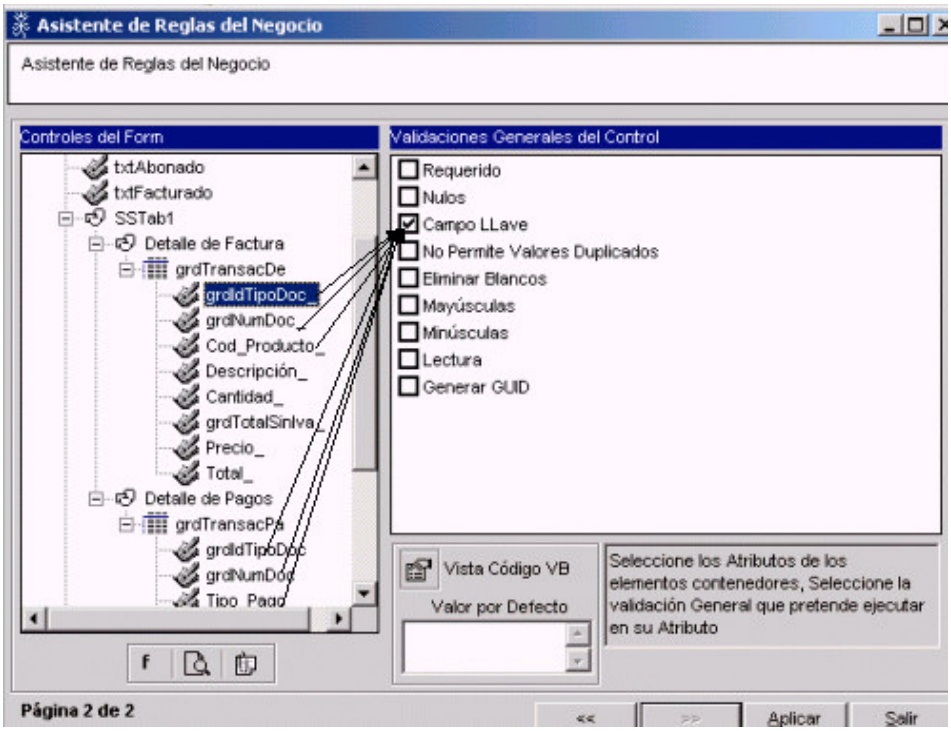
IdTipoDoc,O
NumDoc,O
IdProducto,O
None,X, Definir Búsquedas para Desplegar Información
Cantidad,O
TotalSinIva,O
None,X, Definir Búsquedas para Desplegar Información
TotalConIva,O

Fuente. **eZway Enterprise Edition**

5.20 Reglas del negocio

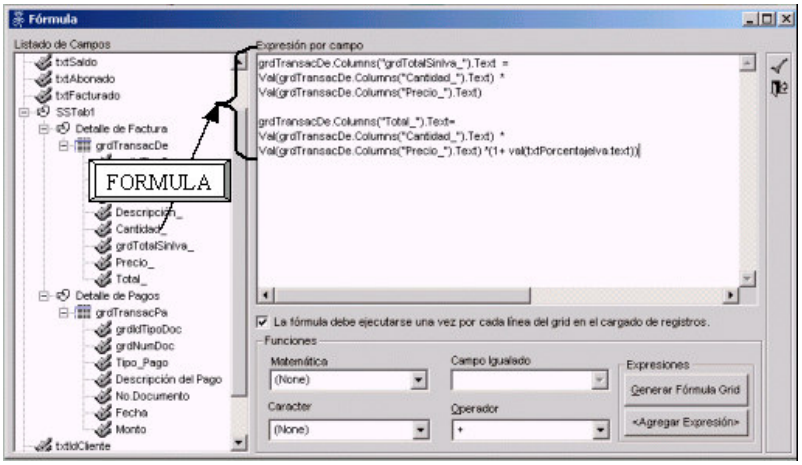
eZway facilita al programador la configuración del comportamiento que debe ejecutar cada control en la pantalla final del usuario. Estas validaciones se asignan al nivel de cada control del formulario y se ejecutan en la capa II Objetos del negocio, aunque también se pueden configurar para ejecutarse en la vista o formulario.

Figura 77. Muestra pantalla asistente de reglas de negocio



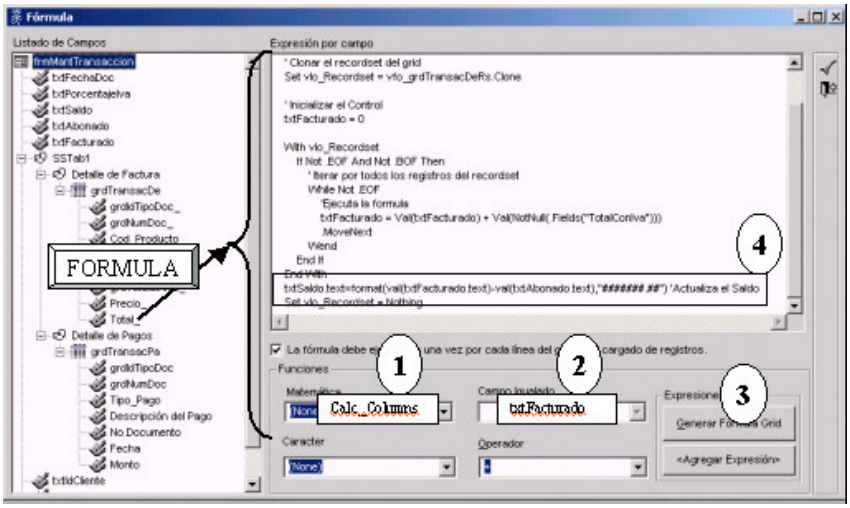
Fuente. **eZway Enterprise Edition**

Figura 78. Muestra ingreso de fórmula al campo cantidad de la forma detalle de factura



Fuente. **eZway Enterprise Edition**

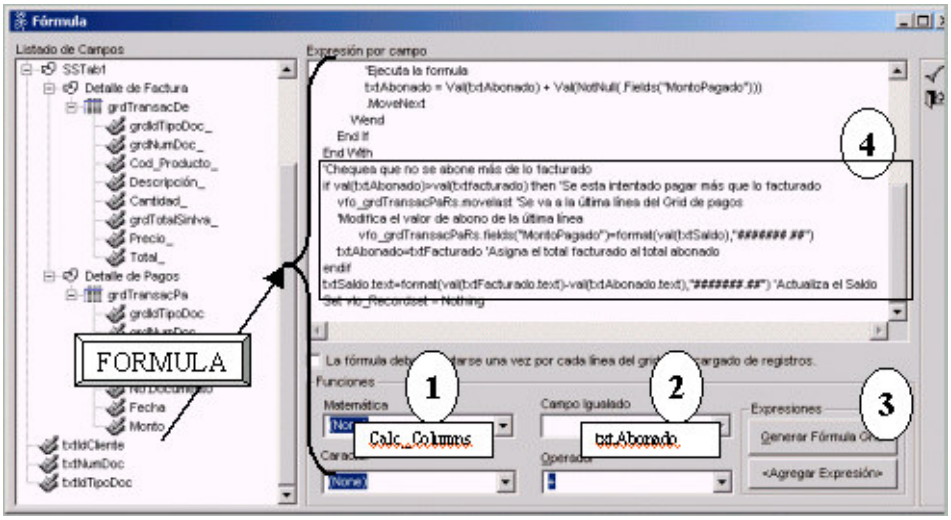
Figura 79. Muestra ingreso de la fórmula al campo



Fuente. **eZway Enterprise Edition**

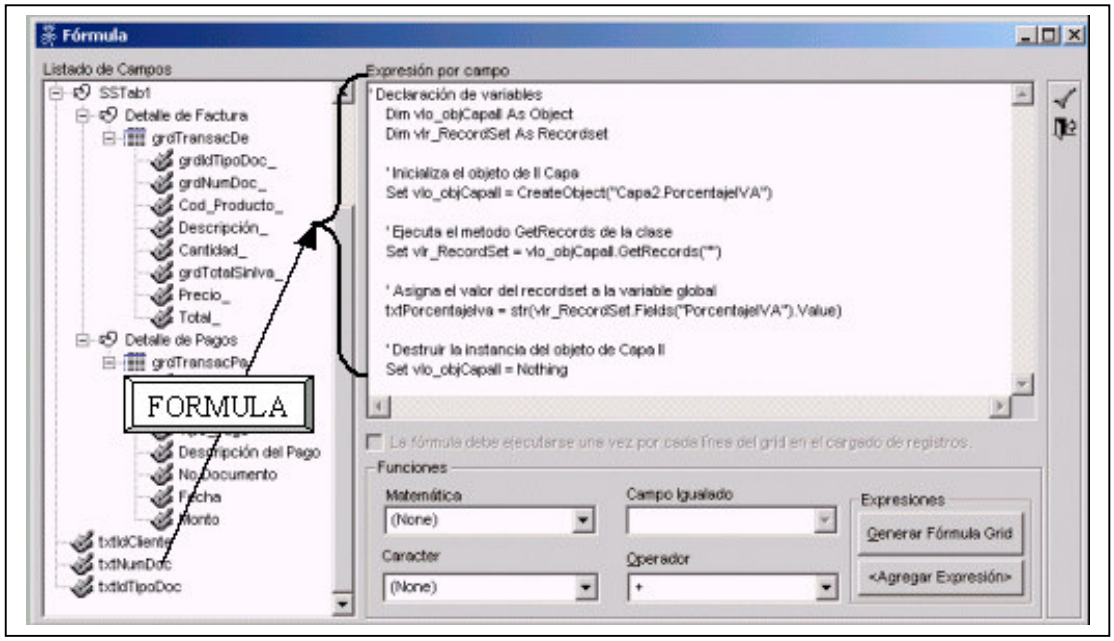
En esta pantalla donde ingresamos la formula al campo saldo de la forma detalle de factura. txtSaldo.text = format(Val(txtFacturado.text)-Val(txtAbonado.text), “#######.##”) ‘Actualiza el Saldo.

Figura 80. Muestra pantalla donde se ingresa la fórmula al campo monto en la forma detalle de factura



‘Chequea que no se Abone mas de lo Facturado
if Val(txtAbonado)>Val(txtFacturado) then
vfo_grdTransacPaRs.movelast ‘Se va a la última línea del Grid de Pagos
‘Modifica el Valor de Abono de la última Línea
vfo_grdTransacPaRs.fields(“MontoPagado”)=Format(Val(txtSaldo), “#######.##”)
txtAbonado=txtFacturado
endif
txtSaldo.text = Format(Val(txtFacturado.text)-Val(txtAbonado.text), “#######.##”) ‘Actualiza el Saldo

Figura 81. Muestra pantalla donde se ingresa la fórmula para el cálculo del IVA.

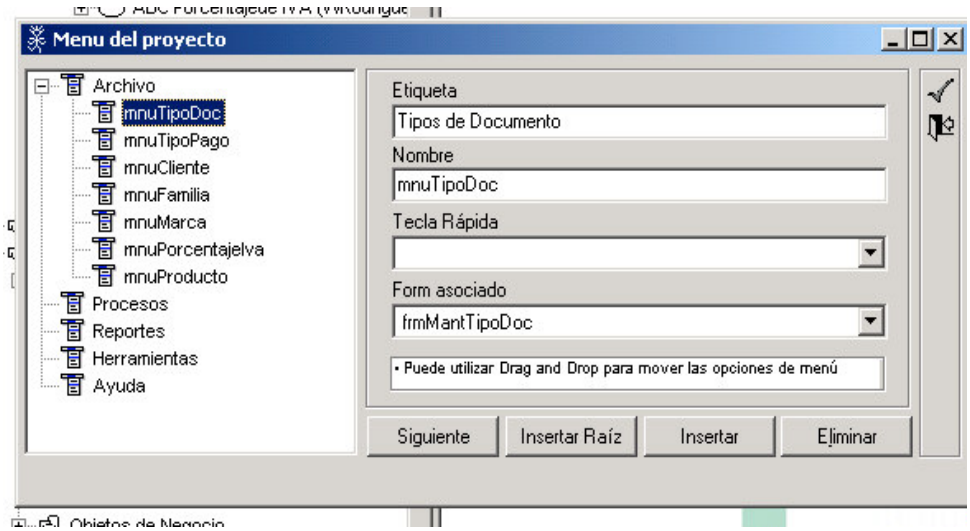


Fuente. **eZway Enterprise Edition**

```
'Declaración de Variables
Dim vfo_objCapaII as Object
Dim vfr_RecordSet as adodb.RecordSet
'Inicializa el Objeto de Capa II
Set vfo_objcapaII = createobject ("Capa2.PorcentajeIva")
'Ejecuta el método GetRecords de la Clase PorcentajeIva
Set vfr_recordset =vfo_objcapaII.getrecords("")
'Asigna el Valor del RecordSet a la variable Global
txtporcentajeiva = str(vfr_recordset.fields("PorcentajeIva").Value)
'Destruye la Instancia del Objeto de Capa 2
Set vfo_objcapaII = nothing
```

5.21 Menú del sistema para Visual Basic

Figura 82. Muestra pantalla principal donde se realiza el menú inicial para Visual Basic



Fuente. **eZway Enterprise Edition**

5.22 Código personalizado

Ingreso de código personalizado a las distintas capas que se encuentran en eZway.

5.22.1 CAPA I

5.22.1.1 INSERT – CAPA I

' EFECTO: Para ejecutar una transacción '

' REQUIERE : Nada.

' MODIFICA : Las tablas a las cuales se les está dando mantenimiento.

' EXCEPCIÓN: En cualquier error se pasa el control al procedimiento "ErrorControl"

```
Private Sub cmdInsert_Click()
' Para instanciar el componente de capa II utilizado en la transacción
Dim vlo_Object As Object
' Inicializar manejo de excepciones
Err.Clear
On Error GoTo CatchError

' Inicializar objeto que maneja el XML.
Set vfo_DataManager = New DataManager

 ' Agregar el maestro para armar el XML que viaja a las capas para los
borrados.
With vfo_DataManager
      ' Agregar el maestro
       ' Los parámetros del Addmaster son
       '1 y 2 El nombre del Componente de Capa 2
       '3 El tipo de Transaccion

       .AddMaster "Capa2.Producto", "Capa2.Producto", ezn_INSERT

       'Agrega los campos al XML
       ' Los parámetros del AddField son
       '1 El nombre del Componente de Capa 2
       '2 Tipo de Mantenimiento en este caso un Maestro
       '3 El nombre del campo como llave para el XML
       '4 El nombre del campo
       '5 El valor inicial del campo antes de insertarlo
       '6 Indicador booleano si el valor requiere comillas (string)
       ‘7 Indica si el Campo A Insertar es Llave Primaria.
```

```
        '8 El nuevo valor a insertar

.AddField "Capa2.Producto", ezn_MASTER, "IdProducto", "IdProducto", "",
True, True, "00009"
.AddField "Capa2.Producto", ezn_MASTER, "IdFamilia", "IdFamilia", "", True,
False, "abar"
 .AddField "Capa2.Producto", ezn_MASTER, "IdMarca", "IdMarca", "", True,
False, "arec"
 .AddField "Capa2.Producto", ezn_MASTER, "NomProducto", "NomProducto",
"", True, False, "Concentrado Varios"
 .AddField "Capa2.Producto", ezn_MASTER, "PrecioBaseUnidadSinIva",
"PrecioBaseUnidadSinIva", "", False, False, "50.85"
 .AddField "Capa2.Producto", ezn_MASTER, "Activo", "Activo", "", True, False,
"S"

End With

' Instanciar el objeto de capa II para el Insert
Set vlo_Object = CreateObject("Facade.BusinessFacade")

' Mandar a Insertar el registro
vlo_Object.Execute vfo_DataManager.GetData

' Destruir la instancia del objeto
Set vlo_Object = Nothing

' Destruir la instancia del objeto que maneja el XML
Set vfo_DataManager = Nothing
```

```
' Control de excepciones
    Exit Sub
CatchError:
    ' Destruir la instancia del objeto que maneja el XML
    Set vfo_DataManager = Nothing
    ' Destruir la instancia del objeto
    Set vlo_Object = Nothing

   ' Llamar al controlador de excepciones.
    ErrorControl Err.Number, Me.Name & "InsertMySql-> " & Err.Source, _
                 Err.Description, Err.Description
End Sub
```

UPDATE (UPdate producto set producto.nomproducto='Concentrado Engordin' from producto inner join marca on (producto.idmarca=marca.idmarca) where marca.nommarca='Arecan' and producto.nomproducto like '%Concentrado varios%')

5.22.1.2 UPDATE – CAPA I

```
' EFECTO   : Para ejecutar una transacción '
' REQUIERE : Nada.
' MODIFICA : Las tablas a las cuales se les está dando mantenimiento.
' EXCEPCIÓN: En cualquier error se pasa el control al procedimiento
"ErrorControl"
Private Sub cmdUpdate_Click()
    ' Para instanciar el componente de capa II utilizado en la transacción
    Dim vlo_Object As Object
```

```
    ' Para almacenar la sentencia sql
    Dim pvc_SQL As String

    ' Instanciar el objeto de capa II para el borrado
    Set vlo_Object = CreateObject("Capa2.Producto")
' Se asigna el string con la sentencia SQL
    pvc_SQL = "Update producto set producto.nomproducto='Concentrado
Engordin' " & _
            "from producto " & _
            "inner join marca on (producto.idmarca=marca.idmarca) " & _
            "where marca.nommarca='Arecan' and producto.nomproducto like
'%Concentrado varios%' "

    ' Mandar a ejecutar el proceso con el parámetro del SQL
    vlo_Object.Proceso pvc_SQL
    ' Destruir la instancia del objeto
    Set vlo_Object = Nothing

    ' Control de excepciones
    Exit Sub
CatchError:

    ' Destruir la instancia del objeto
    Set vlo_Object = Nothing
' Llamar al controlador de excepciones.
    ErrorControl Err.Number, Me.Name & "UpdateMySql-> " & Err.Source, _
                 Err.Description, Err.Description
End Sub
```

5.22.1.3 DELETE – CAPA I

```
' EFECTO   : Para ejecutar una transacción '
' REQUIERE : Nada.
' MODIFICA : Las tablas a las cuales se les está dando mantenimiento.
' EXCEPCIÓN: En cualquier error se pasa el control al procedimiento
"ErrorControl"
Private Sub cmdDelete_Click()
    ' Para instanciar el componente de capa II utilizado en la transacción
    Dim vlo_Object As Object

    ' Para almacenar la sentencia sql
    Dim pvc_SQL As String

    ' Instanciar el objeto de capa II para el borrado
    Set vlo_Object = CreateObject("Capa2.Producto")

    ' Se asigna el string con la sentencia SQL
    pvc_SQL = "Delete producto " & _
              "from producto " & _
              "inner join marca on (producto.idmarca=marca.idmarca) " & _
              "where marca.nommarca='Arecan' and producto.nomproducto like
'%Engordin%' "

    ' Mandar a ejecutar el proceso con el parámetro del SQL
    vlo_Object.Proceso pvc_SQL
```

```
    ' Destruir la instancia del objeto
    Set vlo_Object = Nothing

       ' Control de excepciones
    Exit Sub
CatchError:

    ' Destruir la instancia del objeto
    Set vlo_Object = Nothing

    ' Llamar al controlador de excepciones.
    ErrorControl Err.Number, Me.Name & "UpdateMySql-> " & Err.Source, _
                 Err.Description, Err.Description
End Sub
```

5.22.1.4 SELECT – CAPA I

```
' EFECTO   : Carga la pantalla del mantenimiento con los datos traídos de la
base de datos. Se posiciona en el primer registro.
' REQUIERE : Que los componentes de capa II utilizados para el cargado estén
disponibles.
' MODIFICA : Los controles del formulario, pues se les establecen los valores
del primer registro.
' EXCEPCIÓN: En cualquier error se pasa el control al procedimiento
"ErrorControl"
Private Sub cmdSelect_Click()
    ' Para instanciar los componentes de capa II utilizados en la carga de datos.
    Dim vlo_Object As Object
    ' Para instanciar el recordset
    Dim vlr_RecordSet As Recordset
    ' Inicializar manejo de excepciones
```

```
    Err.Clear
    On Error GoTo CatchError

    ' Inicializar el objeto para cargar los datos al frmBase.
    Set vlo_Object = CreateObject("Capa2.Producto")

    ' Establecer el recordset principal del frmBase
    Set vlr_RecordSet = vlo_Object.CargarRegistros()

' NOTA : En esta parte del código puede recorrer el recordset para obtener los
' valores y mostrarlos al usuario
vlr_RecordSet.MoveLast
MsgBox Str(vlr_RecordSet.RecordCount)

    ' Destruir la instancia del objeto
    Set vlo_Object = Nothing

    ' Control de excepciones
    Exit Sub
CatchError:
    ' Eliminar la instancia del componente
    Set vlo_Object = Nothing

    ' Llamar al controlador de excepciones.
    ErrorControl Err.Number, Me.Name & ".CapaI_CargarRegistros()-> " &
Err.Source, _
                Err.Description, Err.Description
End Sub
```

5.22.2 CAPA II

5.22.2.1 UPDATE/DELETE – CAPA II

```
' EFECTO   : Ejecuta una sentencia SQL
' REQUIERE : Un criterio, opcional el campo por que ordenar.
' MODIFICA : Nada.
' EXCEPCIÓN: Si se produce un error este se pasa al controlador de errores
Public Sub Proceso(pvc_SQL As String)
' Para crear una instancia de la tercera capa
    Dim vlo_DataService As Object

    ' Inicializar el manejo de excepciones.

    On Error GoTo CatchError

    ' Crear una instancia de el componente de tercera capa.
    Set vlo_DataService = CtxCreateObject(GetDataService)

    ' Ejecutar el método de III Capa de ejecución del proceso
    vlo_DataService.Proceso pvc_SQL

    ' Terminar la transacción
    CtxSetComplete

    ' Bajar la instancia del componente
    Set vlo_DataService = Nothing

    ' Control de excepciones
    Exit Sub
```

```
CatchError:
    ' Abortar la transacción
    CtxSetAbort
    ' Bajar la instancia del componente
    Set vlo_DataService = Nothing
    ' Pasar el error al controlador de errores
    ControlError ezc_CLASS_NAME, "Proceso"
End Sub
```

5.22.2.2 SELECT – CAPA II

```
' EFECTO   : Solicita a la tercera capa un recordset con la información que
cumpla con el criterio
' REQUIERE : Un criterio, opcional el campo por que ordenar.
' MODIFICA : Nada.
' EXCEPCIÓN: Si se produce un error este se pasa al controlador de errores
Public Function CargarRegistros() As ADODB.Recordset
' Para crear una instancia de la tercera capa
Dim vlo_DataService As Object

    ' Inicializar el manejo de excepciones.

    On Error GoTo CatchError

    ' Crear una instancia de el componente de tercera capa.
    Set vlo_DataService = CtxCreateObject(GetDataService)
    ' Validar el tamaño del bloque
    If vcn_SizeBlock > 0 Then
        ' Pasar el tamaño del bloque
        vlo_DataService.SizeBlock = vcn_SizeBlock
```

```
    End If

    ' Pedir un recordset con los datos,con el criterio y el orden correspondiente
    Set CargarRegistros = vlo_DataService.CargarRegistros()

    ' Terminar la transacción
    CtxSetComplete

    ' Bajar la instancia del componente
    Set vlo_DataService = Nothing

    ' Control de excepciones
    Exit Function
CatchError:
    ' Abortar la transacción
    CtxSetAbort
    ' Bajar la instancia del componente
    Set vlo_DataService = Nothing
    ' Pasar el error al controlador de errores
    ControlError ezc_CLASS_NAME, "CargarRegistros"
End Function
```

5.22.3 CAPA III

5.22.3.1 UPDATE/DELETE – CAPA III

```
' EFECTO   : Ejecuta una sentencia SQL
' REQUIERE : Un criterio, opcional el campo por que ordenar.
' MODIFICA : Nada.
' EXCEPCIÓN: Si se produce un error este se pasa al controlador de errores
Public Sub Proceso(ByVal pvc_SQL As String)
```

```
' Para crear una instancia del broker de datos.
    Dim vlo_DataBroker As Object

    ' Inicializar el manejo de excepciones
    On Error GoTo CatchError
    ' Crear la instancia del broker de datos
    Set vlo_DataBroker = CtxCreateObject("FwDataBroker.TransactionBuilder")

    ' Pedir la conexión para este componente e iniciar una transacción
    vlo_DataBroker.BeginTransactionFromConnecetionString ( _
vlo_DataBroker.GetConnectionStringFromComponentName(App.EXEName))

    ' Armar el SQL y agregarlo a la cola de transacciones
    vlo_DataBroker.AddTransaction pvc_SQL
    ' Pedir que se ejecuten las transacciones en la cola
    vlo_DataBroker.CommitTransactions

    ' Terminar la transacción
    CtxSetComplete

    ' Destruir la instancia del broker
    Set vlo_DataBroker = Nothing
    ' Control de excepciones
    Exit Sub
CatchError:
    ' Abortar la Transacción
    CtxSetAbort
    ' Destruir la instancia del componente
    Set vlo_DataBroker = Nothing
```

```
    ' Pasar el error al controlador de errores
    ControlError ezc_CLASS_NAME, "Proceso"
End Sub
```

DELETE (delete producto from producto inner join marca on producto.idmarca=marca.idmarca where marca.nommarca='Arecan' and producto.nomproducto like '%Engordin%')

5.22.3.2 SELECT – CAPA III

```
' EFECTO   : Crea una instancia del broker de datos y pide un recordset con el SQL construido.
' REQUIERE: Un criterio válido, opcionalmente un ordenamiento y un tamaño de bloque.
' MODIFICA : Nada.
' EXCEPCIÓN: Retorna un recordset con los datos solicitados. En caso de error, este se pasa al controlador de errores.
Public Function CargarRegistros() As ADODB.Recordset
' Para crear una instancia del broker de datos
    Dim vlo_Broker As Object
    ' Utilizada para guardar el SQL a ejecutar
    Dim vlc_SQL As String
        ' Inicializar el manejo de excepciones
        On Error GoTo CatchError
        ' Armar el SQL
        vlc_SQL = " select producto.*,marca.nomMarca" & _
            " from Producto,Marca" & _
            " where producto.idmarca=marca.idmarca and marca.nomMarca='Arecan'"
```

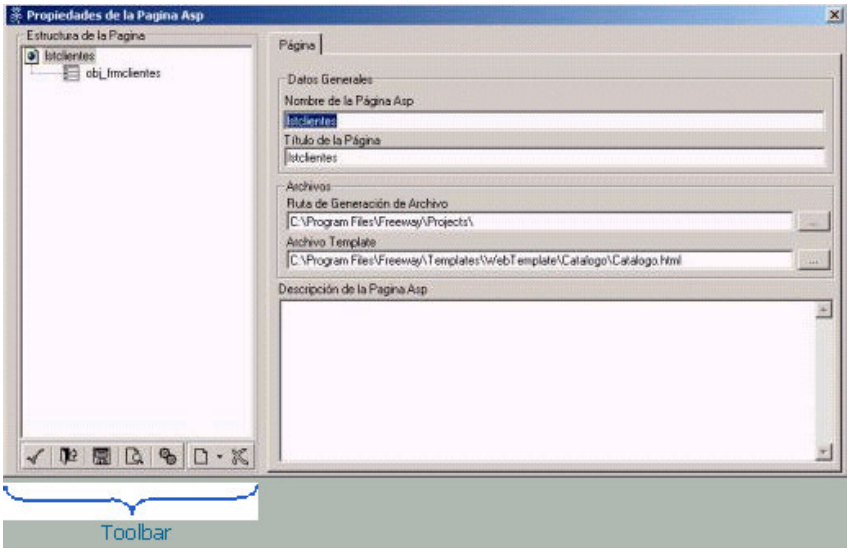
```
' Crear el objeto de servicio de datos  Set vlo_Broker =
CtxCreateObject(ezc_BROKER_NAME)
' Se inicializa el tamaño del bloque en CERO para que el método del broker
cargue todos  'los registros vcn_SizeBlock = 0
 ' Solicitar al Broker el recordset con los datos correspondientes para la consulta
SQL
Set CargarRegistros = vlo_Broker.RunSQLReturnRS_RW(vlc_SQL,
App.EXEName, , vcn_SizeBlock)
      ' Terminar la transacción
   CtxSetComplete
      ' Destruir la instancia del componente  Set vlo_Broker = Nothing
     ' Control de excepciones Exit Function CatchError:
   ' Aborta la transacción
   CtxSetAbort
   ' Destruir la instancia del componente
   Set vlo_Broker = Nothing
   ' Pasa el error al controlador de errores
   ControlError ezc_CLASS_NAME, "CargarRegistros"
End Function
```

El nombre de la forma es frmtutorial. Lo que va dentro es una descripción de lo que va ha realizar cada *command bottom*, es decir que tienen que utilizar 4 *labels* y 4 *command bottom* estos últimos tienen que llamarse tal y como están escritos Ej. INSERT = cmdInsert.

5.23 Crear formularios ASP

Los formularios asp son pantallas orientadas a la publicación en web que le permiten al usuario procesar datos bajo un ambiente Internet / intranet. Para crear un formulario con formato ASP, es necesario posicionarse sobre un caso de uso, hacer click derecho pulsar propiedades y acceder a la cejilla denominada formularios.

Figura 83. Muestra pantalla de propiedades de la página ASP



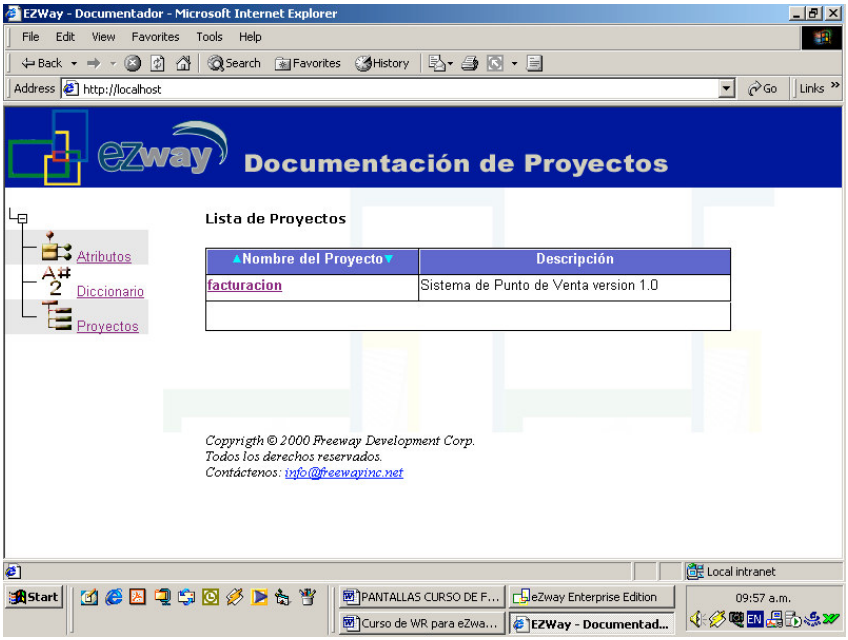
5.24 Configurar el documentador (ezdocumenter)

Una de las grandes características que presenta eZway es que conforme se va desarrollando el proyecto, eZway automáticamente va documentando las diferentes secciones del proyecto.

Para observar la documentación en el proyecto realice los siguientes pasos:

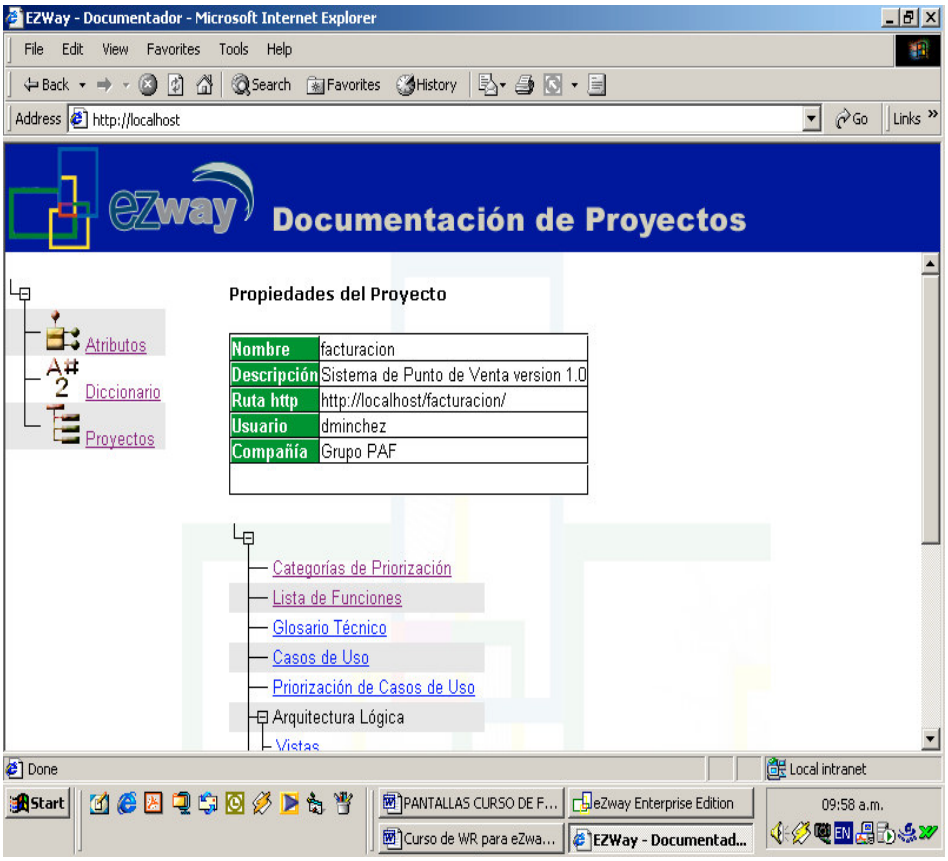
- Inicie una sesión del browser para Internet.
- Digite en la línea de dirección localhost/ezdoc

Figura 84. Muestra pantalla localhost/ezdoc del proyecto realizado



Fuente. **Microsoft Internet Explorer**

Figura 85. Muestra pantalla que visualiza las propiedades del proyecto



Fuente. **Microsoft Internet Explorer**

5.25 Análisis comparativo metodólogia vrs. programación tradicional.

A continuación se presentan las ventajas y desventajas de la utilización de una metodologia asistida por computadora vrs. programación tradicional:

5.25.1 Metodologia asistida por computadora

5.25.1.1 Ventajas

- Permite reutilización de componentes.
- Genera documentación automatica.
- Genera codigo automaticamente.
- Notación gráfica que permite una visión intuitiva y unificada del entorno.
- Facil manejo de proyectos grandes.
- Integración de herramientas para el modelado.
- Modelo preciso y completo
- Lenguaje simple, pero suficientemente expresivo.
- Libera al programador de escribir todo el codigo.
- Compatibilidad con entornos de desarrollo y lenguajes.
- Permite agregar codigo propio.
- Menor tiempo y costo de desarrollo.

5.25.1.2 Desventajas

- Generación de código algunas veces poco entendible.
- Existencia de ambiguedad.

5.25.2 Programacion tradicional

5.25.2.1 Ventajas

- Mayor control en los procesos a crear.

5.25.2.2 Desventajas

- Tiempo extra en generación de documentación.
- No tiene un standar de programación.
- Mayor costo y tiempo de desarrollo.
- No permite el modelado del sistema.
- Dificultad en el manejo proyectos grandes.
- Mayor lineas de codigo
- Detalle de Analisis empoblecido.

5.25.3 Estimacion de tiempos y costos programación tradicional vrs. herramienta asistida por computadora.

Para la estimación de duración del proyecto se utilizó el método de estimación de puntos de función, que consiste en evaluar cada uno de los procesos involucrados en el proyecto y realizar una asignación de un grado de complejidad.

5.25.3.1 Programación tradicional

Para una mejor estimación del proyecto se dividio en dos áreas:

5.25.3.1.1 Estimación del tiempo

Cada uno de los criterios involucrados en la tabla siguiente, dependio de la experiencia en desarrollo de software, capacidad del recurso humano involucrado.

Tabla XII. Muestra criterios involucrados

Componente \ tiempo	Fácil	Media	Difícil
Formas	3 horas	7-8 horas	12-16 horas
Reportes	3 horas	6 horas	16 horas
Procesos	6 horas	12 horas	32 horas

Luego utilice la siguiente tabla:

Tabla XIII. Muestra asignacion de variables

Formas	Facil	Intermedio	Difícil
Cliente			X3
Marca		X2	
Familia		X2	
Porcentajeiva	X1		
Producto			X3
TipoDoc	X1		
TipoPago	X1		

Por lo tanto, el total de tiempo para desarrollar las formas seria de:

Y1 = 50 horas

Ademas se deben tomar los siguientes tiempos:

Tabla XIV. Muestra tiempos involucrados

Actividad	Tiempo (en días)
Investigación Preliminar y Determinación de Requerimientos	4
Análisis	4
Diseño	8
Determinación de estándares a todo nivel, Desarrollo y documentación (Interna y Externa)	8
Pruebas – Modificaciones	5
Instalación - Capacitación	3

Y2 = 32 dias

El tiempo estimado total de este proyecto fue:

T = 39 dias.

5.25.3.2 Estimación del costo

Para la estimación de costos de este proyecto se tomo en cuenta que se incurre en costos de salarios de todo el equipo de desarrollo.

Tabla XV. Muestra salarios del equipo de desarrollo

Puesto	Tiempo a Trabajar (Meses)	Cantidad Recurso Humano	Salario por Persona	Total
Gerente de Proyecto	X	1	Q 16,000.00	Q 16,000.00
Analista-Programador con Experiencia	X	1	Q 10,000.00	Q 10,000.00
Analista - Programador	X	1	Q 7,000.00	Q7,000.00
Programador	X	1	Q 4,000.00	Q 4,000.00
Documentador y pruebas internas	X	1	Q 3,500.00	Q. 3,500.00

Total de Costo = Q 40,000 mensual.

No se toma encuenta el equipo de computo, y Licenciamiento de herramientas a utilizar.

5.25.4 Estimación de tiempos y costos de herramienta asistida por computadora.

Como en la programación tradicional se dividio en dos áreas:

5.25.4.1 Estimación de tiempo

Cada uno de los criterios involucrados en la tabla siguiente, dependio de la experiencia en desarrollo de software con esta metodología, capacidad del recurso humano involucrado.

Tabla XVI. Muestra Criterios involucrados

Componente \ tiempo	Fácil	Media	Difícil
Formas	1 horas	2-4 horas	6-8 horas
Reportes	3 horas	6 horas	16 horas
Procesos	6 horas	12 horas	32 horas

Luego utilice la siguiente tabla:

Tabla XVII. Muestra asignacion de variables

Formas	Facil	Intermedio	Difícil
Cliente			X3
Marca		X2	
Familia		X2	
Porcentajeiva	X1		
Producto			X3
TipoDoc	X1		
TipoPago	X1		

Por lo tanto, el total de tiempo para desarrollar las formas seria de:

Y1 = 23 horas

Ademas se deben tomar los siguientes tiempos:

Tabla XVIII. Muestra tiempos involucrados

Actividad	Tiempo (en días)
Descripción del proyecto y diagrama UML	3
Modelo Lógico	2
Modelo Fisico	4
Desarrollo y documentación (Interna y Externa)	1
Pruebas – Modificaciones	3
Instalación - Capacitación	2

Y2 = 15 dias

El tiempo estimado total de este proyecto utilizando la herramienta asistida por computadora fue:

T = 18 dias.

5.25.4.2 Estimación de costo

Para la estimación de costos de este proyecto utilizando la herramienta asistida por computadora, incurre en costos de salario de dos personas solamente:

Tabla XIX. Muestra salarios del equipo de desarrollo

Puesto	Tiempo a Trabajar (Meses)	Cantidad Recurso Humano	Salario por Persona	Total
Gerente de Proyecto	X	1	Q 16,000.00	Q 16,000.00
Analista-Programador con Experiencia	X	1	Q 10,000.00	Q 10,000.00

Total de Costo = Q 26,000 mensual.

No se toma encuenta el equipo de computo, y licenciamiento de herramientas a utilizar.

CONCLUSIONES

1. El uso de herramientas asistida por computadora, aplicado al análisis, diseño, implementación y documentación, ahorra tiempo significativamente a la hora de realizar proyectos de *software*.

2. El conocimiento del funcionamiento de la herramienta eZway le proporciona al profesional de ingeniería en ciencias y sistemas realizar un proyectos de *software* de mejor calidad.

3. La principal desventaja del uso de sistemas asistidos por computadora es la inversión inicial en licencias, equipo de computación y capacitación del recurso humano.

4. Hay que tener conocimiento previos análisis y diseño de *software*, como también en los programas con los que interactúa para poder tener un *software* de calidad.

5. Como se puede ver en el comparativo de la estimación de tiempos y costos de la programación tradicional vrs. la programacion asistida por computadora, minimizamos un aproximado del 50% en tiempo y costo del proyecto.

RECOMENDACIONES

1. La tecnología avanza con el paso del tiempo, entonces es necesario que el estudiante de ingeniería en ciencias y sistemas, vaya de la mano con dicho avance, por eso se recomienda que se adiestre en el uso de esta y otras herramientas CASE.

2. Es importante que cuando una empresa decida utilizar este tipo de herramienta informática, le proporcione al recurso humano la capacitación necesaria para el uso de dichas herramientas.

3. Otro aspecto de relevancia es que la persona que va a utilizar esta u otra herramienta CASE, debe tener conocimientos sólidos de análisis y diseño de software.

4. Se sugiere a las personas que vayan a utilizar eZway, que antes de hacerlo aprendan a manejar los programas: Microsoft proyect, visio, case estudio , SQL Server , visual basic e internet, ya que dichos programas se complementan mutuamente.

BIBLIOGRAFÍA

1. DATE, C. J. Introducción a los sistemas de bases de datos. Vol. 1, publicada por Addison Wesley Iberoamericana S. A. Quinta edición en español. 1998.

2. PRESSMAN, R. Ingenieria de software un enfoque práctico. Ed. MacGraw-Hill. 1997.

3. SENN, James. Análisis y Diseño de Sistemas de Información. 2o..ed. Editorial MacGraw-Hill. 1992. 11 pp.

4. WILLIAMS, Edwin B. Diccionario Ingles / Español - Español / Ingles. Editorial Bantam Booes: Estados Unidos. 1985. 723 pp.

5. http://www.freewayinc.net/contenido_productos.html. Recopilado de internet el 04 de noviembre 2005

6. http://www.frewayinc.net/contenido_casos.html. Recopilado de internet el 07 de noviembre 2005

7. http://www.artinsoft.com/es/successstories/ss_freeway.asp?page=freeway. Recopilado de internet el 07 de enero 2006

8. http://ceds.nauta.es/Program/vacorp.PDF. Recopilado de internet el 15 de marzo 2006