UNIVERSIDAD DEL VALLE DE GUATEMALA

Facultad de Ingeniería

"ANÁLISIS Y DISEÑO DE UNA SOLUCIÓN PARA EL FÁCIL ACCESO A LA INFORMACIÓN SISTEMATIZADA DE LA EMPRESA PRECON Y OPTIMIZAR EL CONTROL DE LA EMPRESA EN EL ÁREA DE VENTAS"

MIRNA ROCÍO GARCÍA MORALES

Guatemala

2005

""“ANÁLISIS Y DISEÑO DE UNA SOLUCIÓN PARA EL FÁCIL ACCESO A LA INFORMACIÓN SISTEMATIZADA DE LA EMPRESA PRECON Y OPTIMIZAR EL CONTROL DE LA EMPRESA EN EL ÁREA DE VENTAS”

UNIVERSIDAD DEL VALLE DE GUATEMALA

Facultad de Ingeniería

"ANÁLISIS Y DISEÑO DE UNA SOLUCIÓN PARA EL FÁCIL ACCESO A LA INFORMACIÓN SISTEMATIZADA DE LA EMPRESA PRECON Y OPTIMIZAR EL CONTROL DE LA EMPRESA EN EL ÁREA DE VENTAS"

Trabajo de investigación presentado por
MIRNA ROCÍO GARCÍA MORALES
previo a optar al título de
Ingeniera en Ciencias de la Comutación
en el grado académico de
Licenciado

Guatemala

2005

Vo.Bo.:

(f) _____
Ing. Ligia Pimentel

Tribunal:

(f) _____
Ing. Ligia Pimentel

(f) _____
Ing. Roger Sandoval

(f) _____
Ing. Carmelo Durán

Guatemala, 07 de noviembre de 2005

PREFACIO

El tema de trabajo de graduación no es algo fácil de determinar, existe una amplia variedad de temas y ramas para investigación en el área de informática. Sin embargo, mi objetivo es presentar un trabajo útil dentro de la empresa para la que laboro.

A pesar del corto tiempo que he trabajado en la empresa Precon, he notado que el manejo de recurso de información es un área que ha causado problemas y que ha sido descuidada. Me inquieta encontrarle solución a los problemas relacionados con este recurso y pienso que el departamento de ventas es apropiado para iniciar la investigación.

Por lo anterior he propuesto en mi trabajo de graduación el análisis y diseño de un conjunto de aplicaciones que mejoren el acceso a la información en el área de ventas, complementando la información capturada y almacenada en la base de datos de la empresa y capturando nuevos datos de procesos que no incluye el sistema de operaciones utilizado actualmente.

CONTENIDO

LISTA DE CUADROS

Página

LISTA DE GRÁFICOS

Página

RESUMEN

«Un sistema de información puede definirse técnicamente como un conjunto de componentes interrelacionados que permiten capturar, procesar, almacenar y facilitar el acceso a la información para apoyar la toma de decisiones y el control en una institución». Precon es una empresa que provee al mercado de Centroamérica productos masivos como block, adoquín, losas prefabricadas y cercas de concreto y en otros casos, según las necesidades del proyecto, diseña y fabrica productos especiales de construcción. Dicha empresa cuenta con un sistema para automatización de procesos de operación, éste no puede considerarse un sistema de información debido a que presenta dos puntos débiles: no captura toda la información relevante para el control de la institución y aún cuando la información es capturada, procesada y almacenada los miembros de la empresa no tienen acceso a ella. A pesar de las deficiencias del sistema, la empresa no desea reemplazarlo debido al costo y al tiempo invertido para implementarlo. Por lo anterior, este trabajo pretende identificar las necesidades del recurso de información en el departamento de ventas y con base a ellas, proveer el análisis y diseño de una solución en línea que utilice los datos capturados por el sistema, los complemente capturando nuevos datos, los procese y facilite el acceso a la información resultante, con el fin de mejorar el control del departamento.

I. INTRODUCCIÓN

En la actualidad toda organización exitosa se ha concientizado de la importancia del manejo del recurso de información como un elemento clave para la toma de decisiones y el control de la institución.

Para administrar el recurso de información las organizaciones utilizan los sistemas de información que deben capturar los datos, procesarlos y almacenarlos, facilitando el acceso a la información resultante del proceso.

La empresa Precon utiliza actualmente un sistema de información para automatizar los procesos operativos. Este sistema captura datos, los procesa y almacena, pero la información no siempre es accesible a las personas involucradas en todas las etapas operativas. Además, no todos los procesos operativos son tomados en cuenta por el sistema, por lo que no toda la información relevante es capturada.

El sistema utilizado tomó más de 5 años en desarrollarse e implementarse y su costo asciende aproximadamente a $50,000.00. Es por ello que la empresa no desea reemplazarlo, sino utilizar los datos que este sistema captura y procesa, para complementarlos y mejorar el acceso y distribución de la información resultante. Por otra parte la empresa es propietaria únicamente de los archivos ejecutables del sistema, más no del código fuente, por lo que las aplicaciones nuevas deben ser complemento del sistema actual.

Este trabajo es un análisis de la forma en que se maneja el recurso de información en el Departamento de Ventas de Precon. Se describe primero los puestos dentro del departamento para entender quienes están involucrados en el proceso, luego se describe el proceso de ventas y por último se presenta una descripción del sistema actual y las tablas de la base de datos que almacenan la información relevante para el departamento de ventas.

Con base a la información obtenida sobre el funcionamiento actual del departamento y a las solicitudes del Gerente de Ventas, se determinaron las necesidades y se proponen en el trabajo cinco aplicaciones complemento. De estas aplicaciones dos facilitan el acceso a los datos operativos almacenados en la base de datos actual, dos sistematizan procesos que no fueron tomados en cuenta en el sistema y una alimenta una base de datos en Excel a partir de la cual generan los reportes de ventas mensuales.

En la parte final del trabajo se presenta el análisis y diseño de las aplicaciones propuestas, para lo que se utilizan diagramas y documentos de caso de uso, diagramas de secuencia y se especifican las clases que se utilizarán. Por último, se presenta el diseño de una base de datos paralela que utilizarán las aplicaciones.

Espero el trabajo exponga la necesidad de mejorar el control del departamento a través de las aplicaciones propuestas y sea utilizado como una base para iniciar el proceso de mejora del manejo de datos e información dentro de la empresa Precon.

II. OBJETIVOS

A. Generales

- Realizar una investigación sobre sistemas de información y la importancia de la información como un recurso de las empresas.

- Detectar los problemas y necesidades de recurso de información en el área de ventas de la empresa Precon.

- Entender el funcionamiento del sistema de operaciones utilizado actualmente en Precon, para determinar la información que captura y la forma en la que la almacena.

- Proponer un conjunto de aplicaciones (solución) en línea cuya función sea mejorar la comunicación, distribución y el acceso a información con significado y valor para el área de ventas de la empresa Precon, en cualquier momento.

- Delimitar la funcionalidad de las aplicaciones que componen la solución.

B. Específicos

- Comprender el esquema de trabajo del Departamento de Ventas de Precon para detectar y documentar las necesidades de recurso de información en el mismo.

- Identificar, en la base de datos actual, la información capturada relevante para el departamento de ventas, e identificar los datos que aún no son capturados que también presentan valor para el departamento.

- Basado en las necesidades detectadas y la información identificada en la base de datos del sistema actual, proponer una solución que utilice los datos de dicho sistema, los complemente, procese y ponga la información resultante a disposición de los interesados y autorizados en aplicaciones en línea.

- Analizar y diseñar las aplicaciones propuestas en la solución para que sean implementadas posteriormente en la empresa.

III. INFORMACIÓN COMO RECURSO DE UNA EMPRESA

Las empresas han reconocido, desde hace mucho, la importancia de administrar recursos principales tales como la mano de obra y las materias primas. En momentos de cambios acelerados y continuos, como el actual, la información se ha colocado en un lugar adecuado como recurso principal. Se ha desencadenado una rápida evolución en el campo de la información de tal forma que su manejo y utilización ha pasado a ser una disciplina activa y dinámica, así como de trascendental importancia.

El cambio de concepción con respecto a la información ha pasado por tres fases: en la primera (1950-1970), se asociaba la información con una montaña de papeles que formaban parte de las burocracias y que podía llegar a salirse de control: se tomaba como un mal necesario. En la década de 1970, la información se asociaba con la expedición de cientos de informes concernientes con todas las áreas funcionales de la empresa, como un soporte administrativo de las organizaciones.

Desde 1980, la información se ha utilizado como una herramienta vital para el apoyo a las decisiones administrativas y que no sólo sirve para expedir informes sino también para hacer análisis que conduzcan a diagnosticar el estado actual de las empresas.

Al darse cuenta que la información constituye un recurso básico para cualquier actividad humana, que es un bien necesario para la toma de decisiones, el avance de los conocimientos, el control de actividades y el desarrollo económico. Las organizaciones han debido modificar o readecuar sus propias modalidades de gestión. En este contexto la información se ha transformado en un recurso cada vez más indispensable para el éxito de cualquier organización.

Los líderes de las compañías han comprendido que la información no es sólo un subproducto de la conducción, sino que ésta alimenta a los negocios y puede ser el factor crítico para la determinación del éxito o fracaso de éstos. De ahí la necesidad que ésta sea oportuna, precisa, relevante, bien gestionada y orientada hacia los actores de los diferentes procesos organizacionales para la toma de decisiones.

A. Gestión de la información

Una definición concisa sobre gestión de la información es:

> *«Todo lo que se refiere a la obtención de la información adecuada, para la persona adecuada, a su precio adecuado, en el tiempo y lugar adecuado para tomar la decisión adecuada»*

De manera más concreta y aplicándolo a una organización, la gestión de la información es el conjunto de procesos que intervienen en la producción, almacenamiento, recuperación y distribución de los recursos

de información, para que interactúe con los demás recursos de la organización de forma eficiente, efectiva y económica buscando siempre mejorar el funcionamiento de la organización.

La gestión de la información debe coordinar los siguientes procesos:

- cómo la información es capturada, registrada, validada y almacenada.
- cómo la información es utilizada y distribuida.
- cómo las personas que manejan la información aplican sus habilidades y cooperan entre ellas.
- de qué forma las actividades relacionadas con la información contribuyen al logro de los objetivos de la organización.
- cómo se usan las tecnologías de la información en las actividades anteriores.
- qué costos y beneficios conllevan las actividades de información.

Como primer paso en un plan de gestión de información se debe hacer un análisis de la cantidad de información que se recibe y el grado de demanda y la frecuencia de utilización de la información. Con esta base se deben identificar las entidades internas y externas que participan en el intercambio de información. Y por último, se elige la tecnología adecuada que permita realizar la gestión apropiada en cuanto al tipo de información, agilidad y facilidad de acceso.

Respecto al manejo de información generada por computadora es importante mencionar que, por lo general, hay mayor cantidad de información a administrar y el costo de organizarla y mantenerla puede crecer a tazas alarmantes.

Lo anterior conduce al desarrollo de sistemas de información para el manejo y tratamiento de la información en las organizaciones, donde una vez soportados sobre una infraestructura tecnológica, permitirán la agilidad y facilidad de acceso necesaria para la toma de decisiones.

IV. SISTEMAS DE INFORMACIÓN

«Un sistema de información puede definirse técnicamente como un conjunto de componentes interrelacionados que permiten capturar, procesar, almacenar y facilitar el acceso a la información para apoyar la toma de decisiones y el control en una institución»

Como se mencionó en el capítulo anterior. La información constituye un recurso principal en las empresas, útil para controlar sus actividades, tomar decisiones, crear nuevos productos y servicios, evaluar posibilidades de expansión, entre otros. Para producir la información requerida, los sistemas de información llevan a cabo cuatro procesos: alimentación del sistema o entrada de datos, almacenamiento de los datos, procesamiento de los datos y salida de la información propiamente dicha. A continuación se describen cada uno de los procesos mencionados.

- Alimentación del sistema o entrada de datos: Es el proceso por el cual el sistema recibe datos de fuentes internas o externas a la empresa como elementos de entrada. Las entradas pueden ser manuales o automáticas. Las manuales son a aquellas que se proporcionan en forma directa por el usuario, mientras que las automáticas provienen de otros sistemas o módulos.

- Almacenamiento de datos: A través de la capacidad de las computadoras de almacenar datos, un sistema de información convierte los datos de entrada en estructuras que el sistema de la computadora maneja, con el objetivo de utilizar los datos de etapas o procesos anteriores en etapas posteriores o de utilizar los datos para fines de análisis de datos históricos.

- Procesamiento de los datos: Es el proceso a través del cual el sistema actúa sobre los datos para producir información en base a una secuencia de operaciones preestablecida. Estos cálculos pueden efectuarse con datos introducidos recientemente en el sistema o bien con datos históricos, según sea el fin. Esta característica de los sistemas permite la transformación de datos fuente en información que puede ser utilizada para la toma de decisiones y para el control de la organización.

- Salida de información: El sistema produce la información con valor y sentido para el futuro usuario, y la salida es la capacidad de un sistema de información para presentar la información procesada al exterior. En muchos casos la salida de un sistema de información puede constituir la entrada a otro sistema de información o módulo.

A. Clasificación de los sistemas de información

Los sistemas de información son desarrollados con propósitos diferentes dependiendo de las necesidades del negocio. A continuación presento los que a mi parecer son las tres principales categorías de sistemas de información.

1. Sistemas de procesamiento de transacciones. Son sistemas de información computarizados desarrollados para procesar gran cantidad de datos de transacciones rutinarias de los negocios, tales como nóminas e inventarios. Este tipo de sistemas eliminan el tedio de las transacciones operacionales necesarias y reducen el tiempo que alguna vez se requirió para ejecutarlas manualmente, la gente únicamente debe alimentar datos a los sistemas computarizados.

2. Sistemas de información gerencial y de apoyo a decisiones. Suelen introducirse después de haber implantado los sistemas transaccionales más relevantes de la empresa, ya que estos últimos constituyen su plataforma de información. Dan apoyo a un espectro más amplio de tareas organizacionales que los sistemas de procesamiento de transacciones, con el objeto de apoyar la planeación, control y las operaciones de una organización.

Un sistema de información gerencial y apoyo a decisiones utiliza la información del pasado, presente y de proyección para apoyar la solución de problemas y la toma de decisiones. Lo importante es tener disponible la información de manera oportuna y que sea uniforme entre todos los usuarios, aunque la decisión actual todavía es del dominio de la persona que toma la decisión.

Estos sistemas deben definir la forma en que se capturarán y validan los datos, para mantener una integridad y precisión, debido a que datos inexactos o incompletos pueden traer como consecuencia tomas de decisión incorrectas y, por tanto, obtener malos resultados para la organización. Además deben controlar el almacenamiento y la conservación de los datos por medio de procedimientos y normas para garantizar el acceso desde varias aplicaciones, evitando pérdidas, y controlando el acceso sólo a personas autorizadas.

3. Sistemas expertos e inteligencia artificial. La inteligencia artificial puede ser considerada la meta de los sistemas expertos. Los sistemas expertos usan los enfoques del razonamiento de la IA para resolver los problemas que les plantean los usuarios de negocios. Son un caso muy especial de un sistema de información, capturan en forma efectiva y usan el conocimiento de un experto para resolver un problema particular experimentado en una organización. El sistema selecciona la mejor solución a un problema o a una clase específica de problemas.

B. Beneficios de los sistemas de información computarizada

Entre los beneficios que se pueden obtener usando sistemas de información se encuentran los siguientes:

- Organización en el manejo de la información.
- Acceso rápido a la información.
- Generación de informes e indicadores, que permiten corregir fallas difíciles de detectar y controlar con un sistema manual.
- Evitar pérdida de tiempo recopilando información que ya está almacenada en bases de datos que se pueden compartir o replicar.
- Motivación a los empleados para buscar soluciones en base al análisis de datos históricos y para anticipar requerimientos.
- Solución al problema de falta de comunicación entre las diferentes instancias.
- Aumento de la productividad gracias a la liberación de tiempos en búsqueda y generación de información repetida.

En general, los sistemas de información tienen como objetivo:

- Respaldar las operaciones empresariales, contribuyendo a la automatización de actividades y procesos.
- Respaldar la toma de decisiones gerenciales.
- Proporcionar información de valor a las instancias de la empresa que así lo requieran.
- Proporcionar un diagnóstico de la empresa en un momento dado.
- Brindar elementos de juicio para realizar pronósticos empresariales.

Es a través de los sistemas de información, que se mantiene y mejora la eficacia de las instituciones y políticas existentes en una organización, se responde de manera eficaz a los cambios para un mejor desarrollo y se hace frente al impacto de las tecnologías de información para mejorar el rendimiento, la productividad y la responsabilidad en una organización.

V. CICLO DE VIDA DE UN SISTEMA

El desarrollo de un sistema va unido a un ciclo de vida compuesto por una serie de etapas que comprenden todas las actividades, desde el momento en que surge la idea de crear un nuevo sistema, hasta que el sistema deja de ser utilizado.

El ciclo de vida provee una organización para administrar las actividades en el desarrollo del proyecto, conduciendo a que los problemas correctos sean resueltos en el momento correcto.

Existen tres objetivos principales por los cuales seguir un ciclo de vida para el desarrollo de un sistema de información computarizada, estos son:

- Definir las actividades que deben llevarse a cabo en el desarrollo del sistema.
- Introducir consistencia entre varios sistemas de información dentro de la misma organización.
- Proveer puntos de control para la administración de decisiones dentro del desarrollo del sistema.

A. El ciclo de vida clásico de desarrollo de sistemas

Todo proyecto debe pasar por algún tipo de análisis, diseño e implementación. El ciclo de vida clásico incluye las ocho etapas que se describen a continuación.

1. Identificación de problemas, oportunidades y objetivos. Esta fase requiere que se observe lo que está sucediendo en la organización para encontrar problemas y situaciones que pueden ser mejoradas con el uso de sistemas de información computarizados. Esta etapa es crítica para el éxito del resto del proyecto debido a que nadie quiere desperdiciar el tiempo subsecuente resolviendo el problema equivocado.

Las personas involucradas en esta fase son los usuarios, analistas y administradores de sistemas que coordinan el proyecto. Las actividades de esta fase consisten en entrevistas a los futuros usuarios, estimación del alcance del proyecto y documentación de los resultados.

2. Determinación de los requerimientos de información. En esta fase el analista debe comprender qué información necesitan los usuarios para realizar su trabajo, sirve para formar la imagen que el analista tiene de la organización y sus objetivos. Se deben entender los detalles de las funciones actuales del sistema: quién (las personas que están involucradas), qué (la actividad del negocio), dónde (el ambiente donde se lleva a cabo el trabajo), cuándo (en qué momento) y cómo

manera se desarrollan los procedimientos actuales). Al término de esta fase el analista debe comprender el por qué de las operaciones del negocio y tener información completa sobre las personas, objetivos, datos y procedimientos involucrados.

3. Análisis del sistema. Durante el análisis del sistema se determinan los objetivos y límites del sistema objeto de análisis, se caracteriza su estructura y funcionamiento y se marcan las directrices que permitan alcanzar los objetivos propuestos y evaluar sus consecuencias.

El análisis inicia con la conceptualización, que consiste en obtener una visión de muy alto nivel del sistema, identificando sus elementos básicos y las relaciones de éstos entre sí y con el entorno. Luego de la conceptualización se crean un conjunto de modelos y diagramas sobre el funcionamiento del sistema y la forma en la que el usuario interactúa con él.

Un componente esencial del análisis lo constituyen los casos de uso. Éstos describen bajo la forma de acciones y reacciones el comportamiento de un sistema desde el punto de vista del usuario. Permiten definir los límites del sistema y las relaciones entre el sistema y el entorno. Los casos de uso son descripciones de la funcionalidad del sistema independientes de la implementación. El analista debe descubrir a los actores que inician cada caso de uso y a los actores que se beneficiarán de ellos (un actor puede ser tanto un sistema como una persona). El objetivo es analizar la secuencia de pasos a ejecutar por cada caso de uso.

Con base a la información obtenida de los casos de uso se realizan los diagramas secuencia y colaboración que también forman parte del análisis. Los diagramas de caso de uso secuencia y colaboración se describen más a detalle en el siguiente capítulo.

El análisis debe reflejar todas aquellas limitaciones impuestas al sistema que restringen el margen de las soluciones posibles. Estas se derivan a veces de los propios objetivos del sistema.

A fin de comprobar que el análisis efectuado es correcto y evitar la posible propagación de errores a la fase de diseño, es imprescindible proceder a la validación del mismo. Para ello hay que comprobar los extremos siguientes:

- El análisis debe ser consistente y completo.
- Si el análisis se plantea como un paso previo para realizar un diseño, habrá que comprobar, además que los objetivos propuestos son correctos y realizables.

4. Diseño del sistema. El diseño de sistemas se ocupa de desarrollar las directrices propuestas durante el análisis en términos de aquella configuración que tenga más posibilidades de satisfacer los objetivos planteados. Éste incluye decisiones acerca de la organización del sistema en subsistemas, la asigna-

de subsistemas a componentes hardware y software, y decisiones fundamentales conceptuales respecto a la organización global del sistema, que se denomina la arquitectura del sistema.

El proceso de diseño de un sistema complejo se suele realizar de forma descendente:

- Diseño de alto nivel o descomposición del sistema a diseñar en subsistemas menos complejos.
- Diseño e implementación de cada uno de los subsistemas
- Integración de todos los subsistemas.
- Validación del diseño.

La fase de análisis determina lo que debe hacer la implementación y la fase de diseño del sistema determina el plan de ataque. En esta fase se determinan las definiciones completas de las clases y asociaciones que se utilizarán en la implementación, así como los algoritmos de los métodos utilizados para implementar las operaciones. La fase de diseño añadirá objetos internos para la implementación y optimizará las estructuras de datos y los algoritmos. Los objetos descubiertos durante el análisis sirven como esqueleto del diseño, pero el diseñador debe escoger distintas formas de implementarlos con el objetivo de minimizar el tiempo de ejecución, la memoria y el costo. En particular, las operaciones identificadas durante el análisis deben expresarse en forma de algoritmos, descomponiendo las operaciones complejas en operaciones internas más sencillas. Las clases, atributos y asociaciones del análisis deben de implementarse en forma de estructuras de datos específicas.

Conforme prolifera la información, es esencial un enfoque planeado y sistemático para la introducción, modificación y mantenimiento de los sistemas de información, el análisis y diseño de sistemas proporciona esto.

El análisis y el diseño es un enfoque sistemático para la identificación de la entrada de datos, el proceso o transformación de los datos, el almacenamiento de datos y la salida de la información dentro del contexto de una empresa en particular. Además, el diseño y el análisis de sistemas es usado para analizar, diseñar e implementar mejoras en el funcionamiento de los negocios que pueden ser logradas por medio del uso de los sistemas de información.

Este trabajo abarcará hasta esta etapa del ciclo de vida de cada uno de los sistemas que conforman la solución propuesta, sin embargo, a continuación se describen brevemente las etapas subsiguientes.

5. Desarrollo y documentación. En esta fase del ciclo de vida del desarrollo de sistemas, se traduce el diseño de la fase anterior al lenguaje de computadora que se haya seleccionado. También se desarrolla la documentación efectiva para el software, incluyendo manuales de procedimientos. La documenta-

ción le dice al usuario la manera de usar el software y también qué hacer si suceden problemas con el software.

6. Pruebas sobre el sistema. Antes que pueda ser usado el sistema de información debe ser probado. Es mucho menos costoso encontrar problemas antes que el sistema sea entregado a los usuarios. Primero se ejecuta una serie de pruebas para que destaquen los problemas con datos de ejemplo y eventualmente con datos reales del sistema actual.

El mantenimiento del sistema y su documentación comienza en esta fase y es efectuado rutinariamente a lo largo de la vida del sistema de información.

7. Implementación y evaluación del sistema. En esta fase se implementa el sistema para que sea utilizado dentro de la organización. Esto incluye el entrenamiento de los usuarios para que manejen el sistema. Se debe implementar un plan para una conversión suave del sistema antiguo al nuevo, evaluando como los usuarios se van adaptando al nuevo sistema.

8. Mantenimiento del sistema. Después de que el sistema está instalado se le debe dar mantenimiento. El mantenimiento se realiza por dos razones. La primera de éstas es para corregir errores de software (mantenimiento correctivo). Y la segunda es para mejorar las capacidades del software en respuesta a las necesidades organizacionales cambiantes, ya sea por solicitud de características adicionales por parte de los usuarios o por cambios en el negocio luego de un tiempo.

En resumen, el mantenimiento es un proceso continuo a lo largo del ciclo de vida de un sistema de información, conforme pasa el tiempo y cambia el negocio y la tecnología los esfuerzos de mantenimiento se incrementan dramáticamente. Muchos de los procedimientos sistemáticos que emplea el analista a lo largo del ciclo de vida del desarrollo del sistema pueden ayudar a asegurar que el mantenimiento minimice.

Debe hacerse notar, que a veces, los sistemas trabajan en forma cíclica, al terminar una fase y pasar a la siguiente, el descubrimiento de un problema puede obligar a que se regrese a la fase anterior y se modifique el trabajo de la misma. De forma que, varias actividades del ciclo de vida pueden suceder simultáneamente y las actividades pueden ser repetidas.

B. Problemas en el desarrollo de sistemas

La instalación de un sistema sin el previo análisis y diseño lleva a grandes frustraciones, y frecuentemente causa que el sistema deje de ser usado.

Existen algunas características poco deseables en el software y éstas se describen a continuación.

1. Rigidez. Es la tendencia del software a ser difícil de cambiar, incluso en las cosas más sencillas. Cada cambio produce una cascada de cambios en módulos dependientes. Lo que parecía un cambio de dos días en un módulo resulta en varias semanas de cambios en módulos a través de la aplicación.

Cuando el software toma este camino, los gestores temen decir a los programadores que arreglen pequeños problemas que no son críticos. De manera que, lo que empezó siendo un diseño ineficiente acaba siendo una mala política de gestión.

2. Fragilidad. Es la tendencia que tiene el software a romperse por muchos sitios cada vez que se cambia algo. Muchas de las roturas ocurren en sitios que no están relacionados conceptualmente con el área que se está cambiando. Cada vez que los gestores autorizan un cambio temen que el programa se rompa por algún lugar inesperado.

3. Inmovilidad. Es la resistencia del software a ser reutilizado en otros proyectos. El módulo en cuestión depende demasiado de la aplicación en la que está integrado.

4. Viscosidad. Es la dificultad de mantener la filosofía del diseño original. Cuando se afronta un cambio, los programadores encuentran normalmente más de una manera de realizarlo. Algunas de estas formas preservan la filosofía del diseño y otras no. Cuando las formas de implementar el cambio preservando el diseño son más difíciles de realizar que las que no lo preservan, entonces la viscosidad del diseño es alta. Es fácil hacerlo mal y difícil hacerlo bien.

VI. ANÁLISIS Y DISEÑO UML

UML (*Unified Modeling Language/Lenguaje Unificado de Modelado*) es un lenguaje para especificar, construir, visualizar y documentar los componentes de un sistema de software orientado a objetos. UML se enfoca en un lenguaje estándar para modelar no en procesos, debido a que diferentes organizaciones y dominios de problemas requieren diferentes procesos.

UML no es un método de desarrollo, no es una serie de pasos a seguir. Al no ser un método de desarrollo UML es independiente del ciclo de vida que se siga para un sistema.

UML representa un conjunto rico en elementos de notación gráficos. Describe la notación para clases, componentes, nodos, actividades, flujos de trabajo, casos de uso, objetos y estados, y cómo modelar la relación entre esos elementos.

A. Cómo nace UML

UML fue impulsado por los autores de los tres métodos más usados de análisis y diseño orientado a objetos: Grady Booch, Ivar Jacobson y Jim Rumbaugh. Booch había escrito *"Object-Oriented Analysis and Design with Applications"*. James Rumbaugh había desarrollado su propia notación de diseño orientado a objetos llamada OMT (Object Modeling Technique/Técnica de Modelado de Objetos) en su libro *"Object-Oriented Modeling and Design"*. Jacobson, padre de los casos de uso, sorprende a todo el mundo en su libro *"Object-Oriented Software Engineering: A Use Case Driven Approach"*.

En 1994 Rational Software Co. contrató a Rumbaugh en donde ya trabajaba Booch, un año después Jacobson se unía a ellos en Rational. Juntos crearon una notación unificada en la cual basar sus herramientas CASE, en 1997 salió a la luz la versión 1.0 de UML, producto de este trabajo. En el proceso de creación de UML han participado, no obstante, otras empresas de gran peso en la industria como Microsoft, Hewlett-Packard, Oracle o IBM, así como grupos de analistas y desarrolladores.

B. Diagramas de caso de uso

El modelado de casos de uso es una técnica para modelar los requisitos del sistema desde la perspectiva del usuario. Un caso de uso representa una unidad discreta de interacción entre un usuario (humano o máquina) y el sistema, es una unidad de trabajo significativo, por ejemplo agregar un pedido, modificar un pedido o eliminar un pedido son casos de uso. Los casos de uso típicamente se relacionan con actores, un actor es un humano o una máquina que interactúa con el sistema para realizar un trabajo significativo.

Los casos de uso no son parte del diseño (el cómo), sino parte del análisis (el qué), generalmente son el punto de partida del análisis orientado a objetos con UML. Al ser parte del análisis ayudan a describir qué es lo que el sistema debe hacer.

Cada caso de uso tiene una descripción que especifica la funcionalidad que se incorporará al sistema propuesto, puede 'incluir' la funcionalidad de otro caso de uso o puede 'extender' otro caso de uso con su propio comportamiento.

Un caso de uso incluye un diagrama de interacción y lo realmente importante es el documento que describe el caso de uso, en este documento se explica la forma de interactuar entre el sistema y el usuario.

A continuación presento un cuadro con la información mínima que debe incluir el documento de caso de uso y una explicación de cada apartado. El formato de la tabla fue tomado de la página *"Desarrollo de Software Orientado a Objetos"* de Joaquín Gracia.

Cuadro 1: Documento de caso de uso

Nombre:	Nombre del caso de uso
Autor:	Autor del caso de uso
Fecha:	Fecha de creación (puede incluir versión)
Descripción: Descripción de la funcionalidad	
Actores: Usuarios que interactúan con el sistema en este caso de uso	
Precondiciones: Hechos que se han de cumplir para que el flujo de evento se pueda llevar a cabo.	
Flujo normal: Conjunto de pasos que corresponde a la ejecución normal y exitosa del caso de uso.	
Flujo alternativo: Indica cuál es el procedimiento que sigue el sistema en los casos menos frecuentes e inesperados.	
Poscondiciones: Hechos que se ha de cumplir si el flujo de eventos normal se ha ejecutado correctamente.	

(Gracia, 2003)

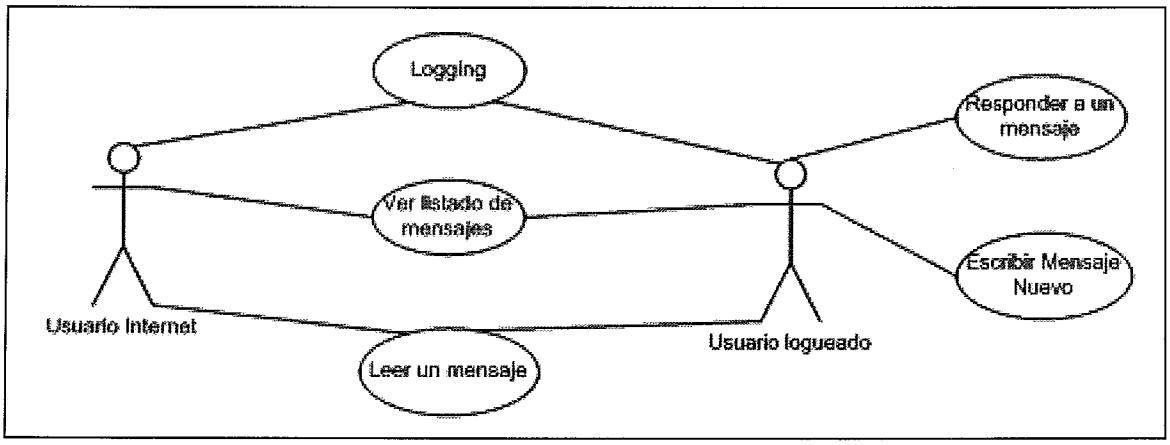
Algunas preguntas claves que deben responderse en el documento son:

- ¿cuáles son las tareas del actor?
- ¿qué información crea, guarda, modifica, destruye o lee el actor?

- ¿debe el actor notificar al sistema los cambios externos?
- ¿debe el sistema informar al actor de los cambios internos?

El diagrama de interacción utiliza, tres elementos básicos: muñecos, pelotas y líneas. Los muñecos son los actores, las pelotas encierran un caso de uso (el cual es descrito en su documento) y las líneas indican la relación entre el actor y el caso de uso. Estos diagramas ayudan a tener una idea global y una visión más clara de la interacción de los usuarios con el sistema.

Figura 1: Ejemplo de diagrama de caso de uso



(Gracia, 2003)

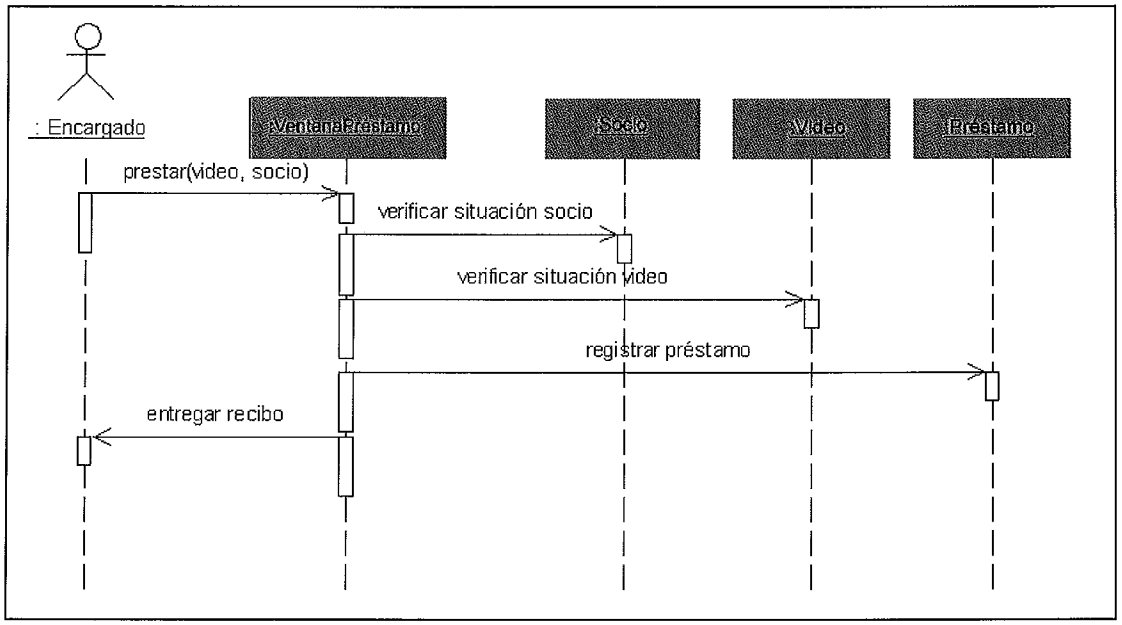
C. Diagramas de secuencia

Un diagrama de secuencia representa la interacción entre los objetos a lo largo del tiempo, muestran a un actor y los objetos y componentes con los que interactúa durante el flujo de un caso de uso. Estos diagramas muestran los objetos de una secuencia mediante líneas verticales y los mensajes son flechas que conectan los objetos. Los mensajes son dibujados cronológicamente de arriba hacía abajo y los rectángulos en las líneas verticales representan los períodos de actividad de los objetos.

D. Diagramas de colaboración

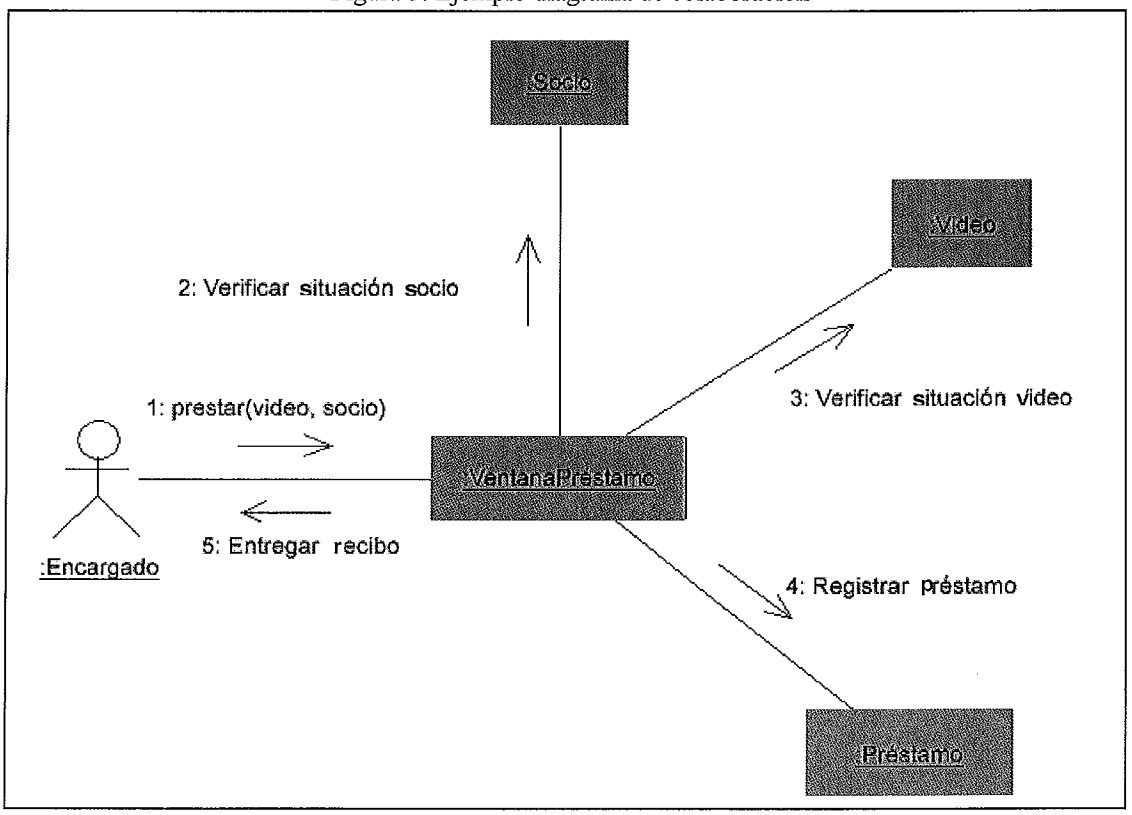
El diagrama de colaboración es otra forma de modelar la interacción entre los objetos de un caso de uso. Lo que cambia es la notación, en estos diagramas los objetos están conectados por enlaces (*links*) en los cuales se representan los mensajes enviados acompañados de una flecha que indica su dirección. A continuación presento el diagrama de colaboración para el mismo ejemplo de la figura 2.

Figura 2: Ejemplo diagrama de secuencia



(Gracia, 2003)

Figura 3: Ejemplo diagrama de colaboración



(Gracia, 2003)

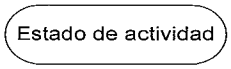
E. Diagramas de actividad

Los diagramas de actividad se utilizan para especificar un método, un caso de uso o un proceso de negocio. Son útiles para entender el comportamiento de alto nivel de la ejecución de un proceso, sin profundizar en los detalles internos de los mensajes. El componente principal es un estado de actividad que representa la ejecución de una secuencia en un procedimiento, o el funcionamiento de una actividad en un flujo de trabajo. Un estado de actividad espera la terminación de su cómputo. Cuando la actividad termina, entonces la ejecución procede al siguiente estado de actividad dentro del diagrama.

1. Notación

- El estado inicial se representa con un círculo relleno.

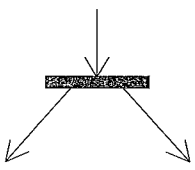
- Un estado de actividad se representa como una caja con los extremos redondeados.



- Las transacciones simples de terminación se muestran como flechas.

- Las condiciones como diamantes con múltiples flechas de salida etiquetadas.

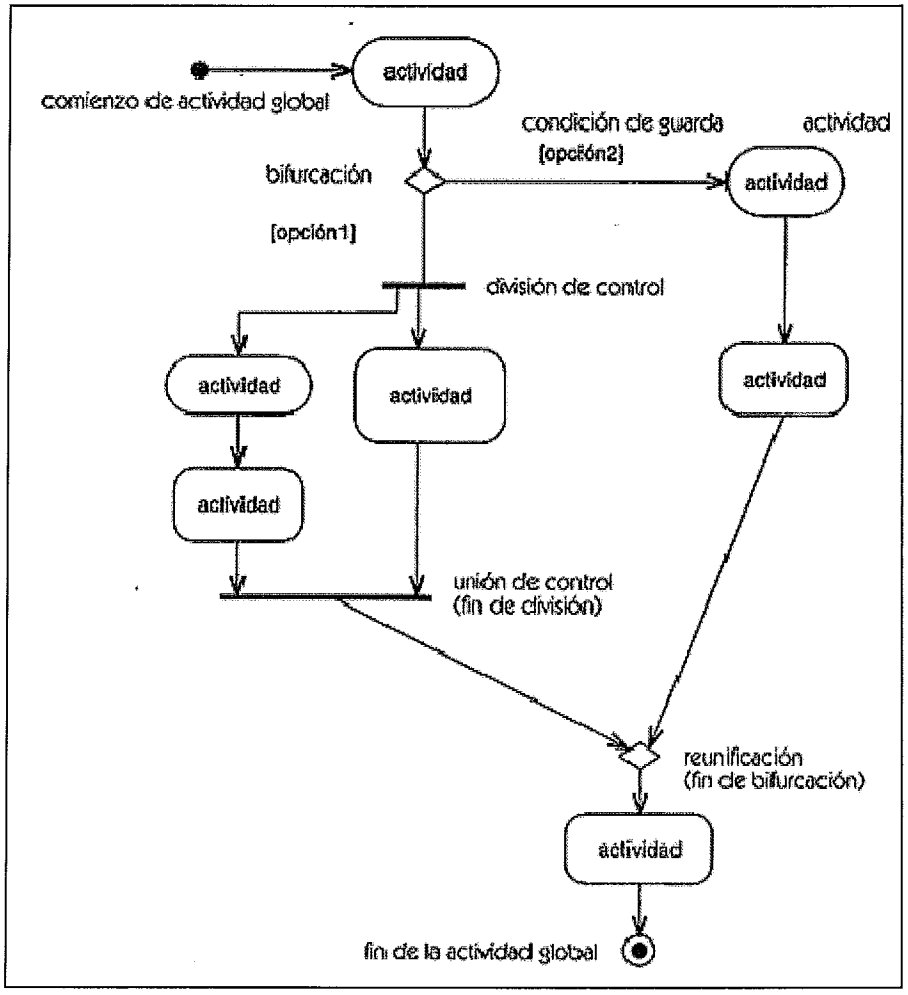
- Una división o una unión de control se representa con múltiples flechas que entran o salen de la barra gruesa de sincronización.



- El estado final como un círculo relleno doble.

En la figura que sigue se muestra la interacción entre los objetos.

Figura 4: Diagrama de actividad



(Jiménez)

F. Diagramas de clases

La clasificación es uno de los mecanismos de abstracción más utilizados. Una clase define un conjunto de objetos; Cada objeto del sistema pertenece a una clase. La definición de una clase incluye definicio-

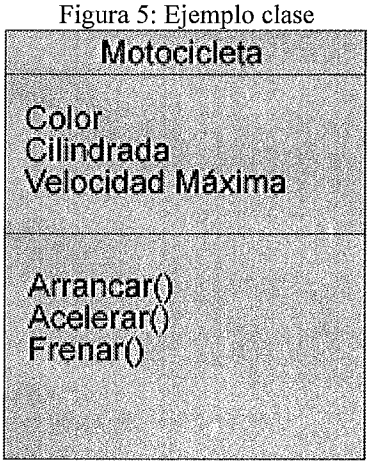
nes para atributos y para operaciones o métodos de los objetos que clasifica. Un diagrama de clases presenta las clases del sistema con sus relaciones estructurales y de herencia.

Los diagramas de clases se desarrollan con base a la información obtenida en los casos de uso, los diagramas de secuencia y de colaboración. Los objetos encontrados durante el análisis son modelados en términos de la clase a la que instancian y las interacciones entre objetos representan las relaciones entre las clases. Estos diagramas son la principal herramienta para el análisis y el diseño del sistema.

Cada clase se representa en un rectángulo con tres compartimientos:

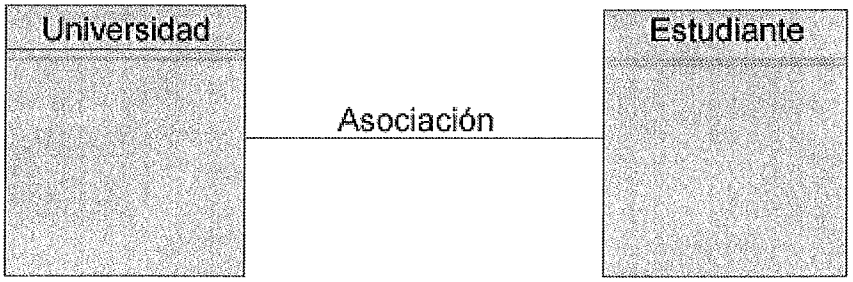
- nombre de la clase
- atributos de la clase
- operaciones de la clase

Figura 5: Ejemplo clase



1. Asociación. Los diagramas de secuencia y colaboración modelan la interacción entre los objetos, en los diagramas de clases una asociación es una abstracción de la relación existente en los enlaces entre los objetos.

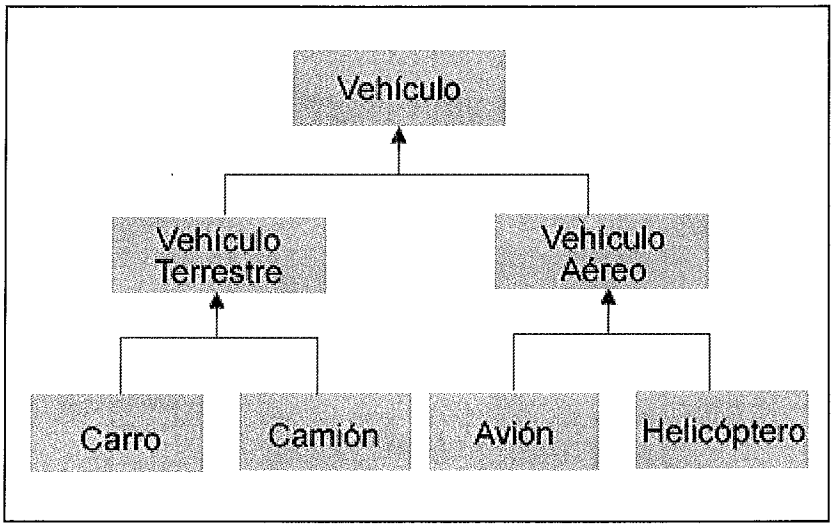
Figura 6: Ejemplo asociación



2. Generalización y agregación. Para formar jerarquías de clases se utilizan las relaciones de agregación y generalización. La agregación representa una relación *parte_de* entre objetos. La Generalización consiste en factorizar las propiedades comunes de un conjunto de clases en una clase más general, de forma que existirá una clase padre o superclase y clases hijas o subclases que heredarán las propiedades generales de la clase padre y sus propiedades le darán su especialización.

En el diagrama de la figura 6, *vehículo* representa la clase más general, *vehículo terrestre* o *vehículo aéreo* son clases más especializadas que poseen los atributos generales de la clase vehículo y sus atributos especiales, de la misma forma carro y camión heredan también los atributos generales de la clase vehículo y de la clase vehículo terrestre y sus atributos les dan la especialización necesaria para representar una clase única de vehículos terrestres.

Figura 7: Ejemplo generalización-especialización



(Gracia, 2003)

VII. PRECON: PRESENTACIÓN Y PRODUCTOS

A. Presentación de la empresa

Los líderes de Precon definen a la empresa como:

> *«Una empresa que ha llegado a constituirse como la mayor productora y proveedora de estructuras de concreto prefabricado en Guatemala y El Salvador.»*

Fundada en 1970, Precon es una empresa que, con sus dos plantas de producción, provee al mercado de Centroamérica productos masivos como block y adoquín, losas prefabricadas y cercas de concreto. También bajo las necesidades de cada proyecto se diseñan y fabrican productos especiales.

Los productos que la empresa ofrece a sus clientes son:

- Losa Vigueta y Bovedilla.
- Tilt Up.
- Block.
- Blocón.
- Formavigas.
- Vigas intermedias.
- Columnas prefabricadas.
- Planchas de cerramiento.
- Vigas tipo "T".
- Vigas tipo "T" con bulbo.
- Adoquín estándar y decorativo.
- Bardas decorativas (Ladricreto y Madecreto).

Para los propósitos de este trabajo no es relevante la descripción de cada producto, esta información es brindada en la página de la empresa, www.precon.com.gt. Sin embargo, sí es importante exponer la forma en la que manejan estos productos internamente y esa información se presenta en el siguiente apartado.

B. Manejo interno de productos

En la empresa se manejan dos grandes categorías de productos:

1. Producto estándar o de línea. Este rubro incluye block o similares, barda y blocón. Es producto prefabricado para el cual existen medidas estándar y se ofrece y vende en alguna de sus presentaciones estándar al cliente. De este tipo de producto se tiene existencias en la planta y puede ser cotizado por metro cuadrado a un cliente por cualquier asesor de ventas.

2. Producto a la medida / Proyectos. Como su nombre lo indica es producto a la medida, en este caso las medidas no son estándar sino de acuerdo a los requerimientos de un proyecto específico. No se tienen existencias de este tipo de producto y para cotizarlo se necesita de la intervención de un asesor técnico y en algunos casos de un asesor arquitectónico, según sea el proyecto.

VIII. DEPARTAMENTO DE VENTAS DE LA EMPRESA

A. Puestos y responsabilidades

El departamento de ventas se encuentra a cargo del gerente de ventas. Él es el encargado de que se cumplan los objetivos del departamento manteniendo siempre la integridad del proceso de ventas.

Las demás responsabilidades del departamento se encuentran distribuidas en los siguientes puestos:

- Secretaria de ventas: Es la responsable del ingreso de pedidos y órdenes de producción de producto estándar al sistema de información, además de sus responsabilidades secretariales con los asesores y el gerente.

- Asistente administrativo de ventas: Es responsable de crear la ficha de un cliente en el sistema, generar órdenes de despacho, lleva control de cobros de pedidos así como de las comisiones de los asesores de ventas y mantiene una base de datos de ventas en Excel actualizada.

- Asesores de ventas: Son el contacto directo con el cliente, deben identificar la necesidad del cliente conduciendo una negociación respecto a precios, condiciones de pago y tiempos de entrega. Inician el trámite administrativo de pedidos y llevan control junto con el asistente administrativo de ventas de los cobros de un pedido.

- Asesores técnicos: Su participación es en pedidos de producto a la medida. Realizan el cálculo estructural de losas y elaboran un presupuesto. Cuantifican el producto y el transporte y elaboran planos de montaje o definitivos de losas vendidas.

- Asesor arquitectónico: Su trabajo es en proyectos que incluyan arquitectura. Realiza la definición arquitectónica de los proyectos vendidos.

- Asesor geotectónico: Su papel es acompañar a los asesores de ventas en las soluciones que involucren geosintéticos.

Un departamento que se encuentra bastante ligado al departamento de ventas es el de despachos, responsable de que el producto llegue al lugar que el cliente ha solicitado. Si el departamento de ventas no genera la orden despacho y libera los pedidos, el departamento de despachos puede tener problemas en entregar un pedido dentro del tiempo ofrecido al cliente

B. Proceso de venta

El procedimiento de ventas se puede dividir en varias etapas básicas, éstas son:

1. Identificación de la necesidad del cliente, cotización y negociación con el asesor de ventas. El asesor de ventas inicia el contacto con el cliente identificando sus necesidades para comercializar los productos de la empresa y según sea producto estándar o producto especial, él realiza la cotización o solicita que la cotización sea hecha por algún asesor técnico respectivamente. Si el cliente acepta el precio de la cotización el asesor lleva a cabo una negociación, condiciones de pago y tiempos de entrega. Si el cliente acepta las condiciones, entonces nace un pedido que el asesor debe reportar a la secretaria de ventas para iniciar el proceso interno de la venta.

2. Generación y autorización del pedido y orden de producción. Para un pedido nuevo existen dos posibilidades, una es que el cliente ya sea conocido por la empresa y otra que el cliente sea nuevo. En el segundo caso antes de ingresar el pedido el asistente administrativo de ventas debe crear la ficha del nuevo cliente y del proyecto para este pedido. Una vez el cliente ya existe en el sistema la secretaria de ventas ingresa el pedido y genera la orden de producción.

Para cada pedido, si se cumplen las condiciones de anticipo o se ha autorizado el crédito, el director financiero administrativo libera el pedido y autoriza la orden de despacho.

3. Traslado de la orden de producción a planta. Un pedido puede sufrir revisiones o cambios luego de ingresado al sistema. Estos cambios son ingresados por la secretaria de ventas y autorizados por el gerente de ventas o el director administrativo antes de que la orden de producción sea llevada a la planta. Al final de cada día, las órdenes de producción autorizadas son trasladadas a la planta, para la fabricación y despacho del producto.

4. Emisión de orden de despacho. Una vez fabricado el producto y listo para despacho, el cliente debe cancelarlo y luego de esto el asistente administrativo emite la orden de despacho. Las órdenes de despacho también son enviadas a la planta al final de cada día.

5. Despacho del pedido. El departamento de despachos de planta emite tres copias de una nota de envío para cada pedido liberado, las cuales son firmadas por el cliente cuando recibe el producto y en donde se anota si algún producto llegó defectuoso para una devolución.

6. Facturación del pedido. Con la copia de la nota de envío, se inicia el proceso de facturación. Todas las facturas emitidas son entregadas a la secretaria para que verifique que los datos coincidan con los del pedido registrado y las entregue a los asesores quienes nuevamente verifican los datos del cliente y finalmente la entregan al cliente y así acaba el proceso de venta.

La figura 8 muestra el flujo de las actividades, del proceso de ventas. Resumiendo, los documentos involucrados en el proceso de venta son:

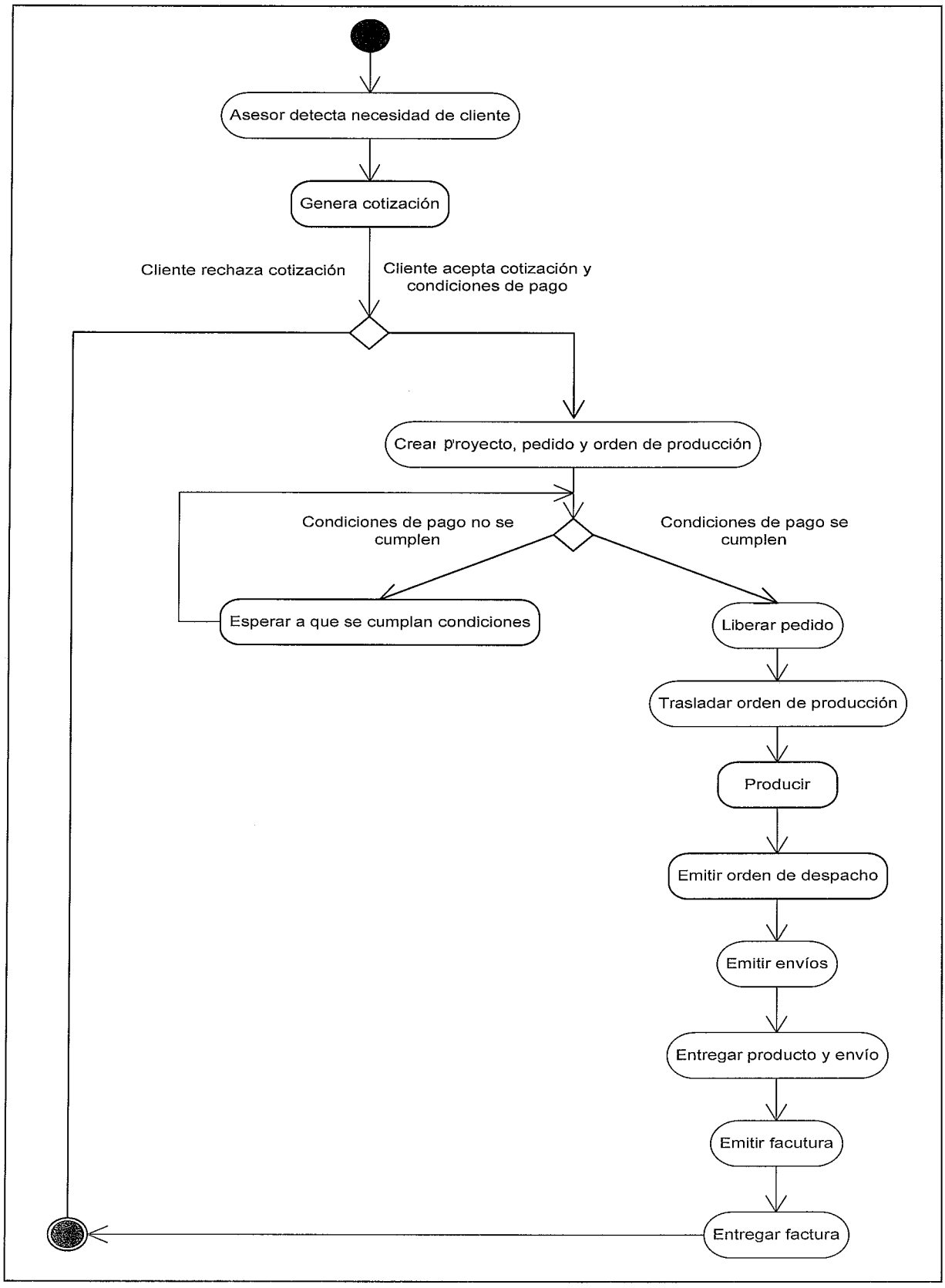
- Cotización,
- Orden de producción,
- Orden de despacho,
- Nota de envío y
- Factura

El primer estado es una cotización que puede ser aceptada o rechazada. Si ésta es aceptada entonces se genera un pedido y amarrado a él, una orden de producción. Un pedido puede estar o no liberado, si el pedido no es liberado no se puede fabricar nada del mismo en planta ni generar la orden de despacho. Una vez el pedido es liberado es llevado a planta en donde puede tener tres estados: en espera para iniciar producción; en producción; y en patio listo para ser despachado. Es aquí en donde se genera una orden de despacho luego de que el cliente ha cancelado el monto total, con la orden de despacho se genera la nota de envío para posteriormente despachar el pedido en la fecha sugerida por la orden de despacho. Luego de despachado el pedido se encuentra en estado pendiente de facturación y con la facturación se termina el proceso.

En el apéndice A, presento ejemplos de los documentos usados en el proceso de ventas, generados por el sistema.

Un segundo proceso involucra una licitación, elaborada por el departamento técnico de la empresa. Si la licitación es adjudicada a Precon, existirán varios pedidos para el mismo proyecto de un cliente, y por cada pedido el proceso que se sigue es el mismo.

Figura 8: Diagrama de actividad del proceso de venta



IX. BASE DE DATOS Y SISTEMA ACTUAL

A. Plataforma y funcionalidad

El sistema actual fue desarrollado sobre la plataforma de Delphi con servidor de base de datos Internase/Firebird. Es un sistema de ambiente de ventanas, con base de datos centralizada en el servidor de la empresa, está compuesto por un conjunto de módulos, cada uno con su archivo ejecutable. Los módulos son instalados en las máquinas clientes que los necesiten, según las funciones que el usuario de la máquina desempeñe. El administrador de la red de la empresa es el encargado de seleccionar los módulos que una máquina tiene instalados.

Los módulos utilizados en el departamento de ventas son:

1. Módulo producc. Es utilizado por la secretaria de ventas para crear cotizaciones que generan un proyecto asociado a un cliente del sistema, dicho proyecto puede tener uno o más pedidos. Por cada pedido que la secretaria ingresa se genera la orden de producción, con el detalle de los productos pedidos, y el pedido queda pendiente de liberar. Se pueden modificar las cotizaciones, proyectos, pedidos y órdenes de producción y únicamente se pueden eliminar los pedidos y órdenes de producción si no han sido despachados. En la figura 9, presento el diagrama de casos de uso del módulo.

2. Módulo ventas. En este módulo el director financiero administrativo puede ver los pedidos pendientes de liberar y liberarlos en el caso en que se cumplan las condiciones de pago. Además, el módulo es utilizado por el asistente administrativo de ventas para crear la ficha de un cliente nuevo, modificar los datos o eliminar un cliente existente. El asistente también lleva en este módulo el control de cobros de un pedido y de las comisiones de los asesores de ventas. El único reporte que se utiliza del módulo es el de comisiones de los asesores de ventas. En la figura 10 se muestra el diagrama de casos de uso del módulo.

3. Módulo despachos. Únicamente es utilizado para generar órdenes de despacho de los pedidos liberados. Se ingresa el cliente y proyecto, se selecciona número de pedido y se especifica la fecha y lugar de despacho. El diagrama de casos de uso para este módulo se presenta en figura 11.

Figura 9: Casos de uso del módulo Producc

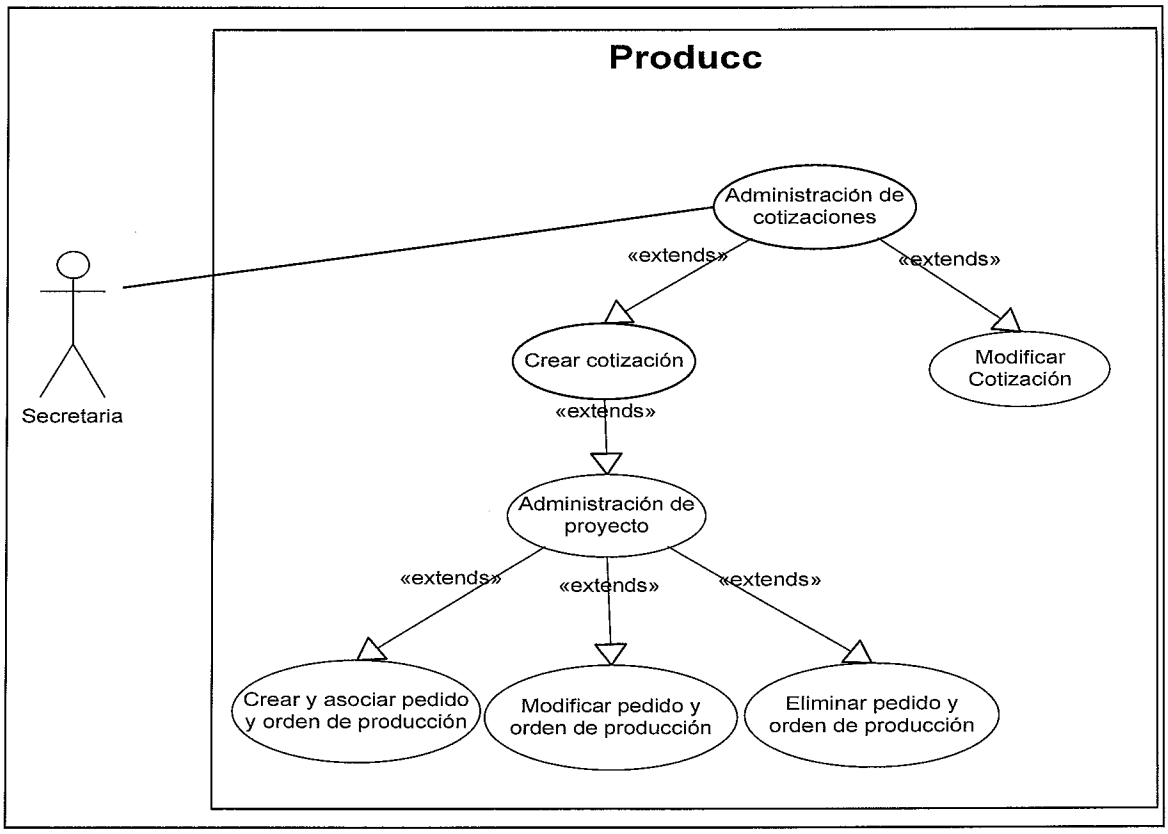


Figura 10: Casos de uso del módulo de ventas

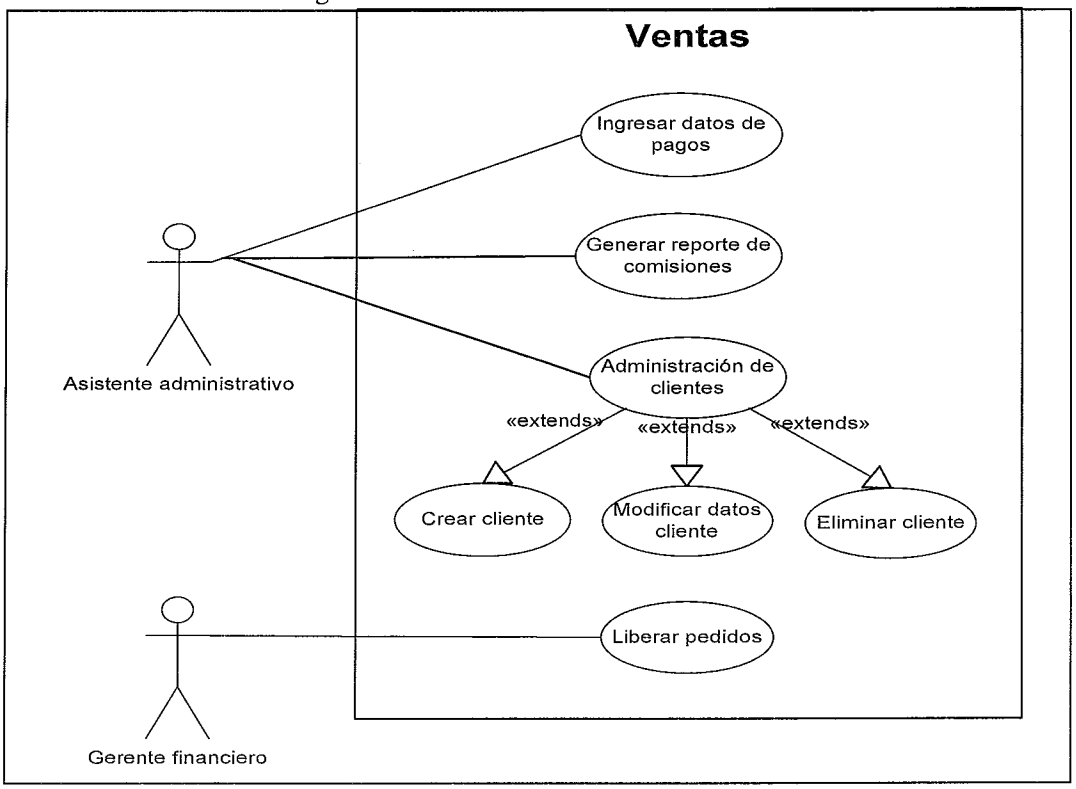
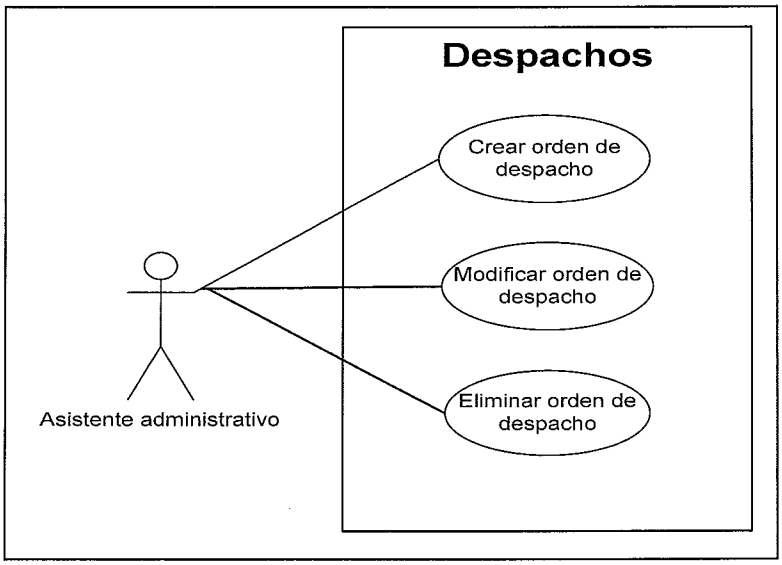


Figura 11: Casos de uso del módulo de despachos



En todos los módulos el ambiente es de ventanas de ingreso de información, en donde el número de pedido es siempre el identificador principal y en base al cual se ingresa la demás información. Hay que resaltar que únicamente se usan los módulos de ingreso de información, la cual sí es almacenada en la base de datos, pero no se utiliza ningún tipo de reportes de este sistema. Se lleva en Excel otra base de datos, que se utiliza para reportes de ventas. Por lo que la información muchas veces es ingresada dos veces en bases de datos diferentes.

D. Base de datos: tablas relevantes para el proceso de ventas

El sistema utiliza una base de datos relacional Interbase / Firebird. Esta base cuenta con 295 tablas; a continuación, presento la descripción de las tablas que almacenan información relacionada con el departamento de ventas. La descripción incluye el nombre de cada tabla, los campos que la conforman, y una breve explicación de los campos más relevantes. Las filas en las que presento los campos de las tablas pueden tener tres colores, según pertenezcan a uno de los siguientes casos.

- Los campos de una fila roja son aquellos que fueron definidos en la estructura de la tabla, pero que en más del 95% de registros son nulos.

- Los campos de una fila amarilla son campos que existen en la estructura de la tabla y en más del 80% de los registros tienen un valor distinto de nulo, pero no almacenan información relevante para este estudio.

- Los campos de una fila verde son campos que existen en la estructura de la tabla; en más del 80% de los registros almacenan un valor distinto de nulo y además representan información relevante para el departamento de ventas.

1. Tabla de clientes. El nombre de la tabla en la base de datos es *Clientes*, los campos que la componen los presento en el siguiente cuadro:

Cuadro 2: Campos de la tabla *CLIENTES*

Nombre del campo	Tipo de dato
ID_EMPRESA	INTEGER
ID_CIUDAD	INTEGER
ID_TIPO_COBRO	INTEGER
ID_MONEDA	INTEGER
RAZON_SOCIAL	VARCHAR(100)
TELEFONO	VARCHAR(20)
REPRESENTANTE	VARCHAR(100)
LIMITE_CREDITO	NUMERIC(15,2)
NIT	VARCHAR(20)
CEDULA	VARCHAR(10)
OBSERVACIONES	VARCHAR(200)
ID_GRUPO	INTEGER
CENTROCOSTO	INTEGER
DEPARTAMENTO	INTEGER
CUENTA	VARCHAR(50)
VENDEDOR	INTEGER
ID_TIPO	INTEGER
CLIENTEID	INTEGER
NOMBRE	VARCHAR(50)
DIRECCION	TEXT
CONTACTO	VARCHAR(30)
ZONA	INTEGER
TELEFONO1	VARCHAR(20)
TELEFONO2	VARCHAR(20)
TELEFONO3	VARCHAR(20)
FAX	VARCHAR(20)
E_MAIL	VARCHAR(20)
CONTACTO1	VARCHAR(50)
CONTACTO2	VARCHAR(50)
CONTACTO3	VARCHAR(50)
TELEFONO_CONTACTO1	VARCHAR(20)
TELEFONO_CONTACTO2	VARCHAR(20)
TELEFONO_CONTACTO3	VARCHAR(20)
ID_CLASECLIENTE1	INTEGER
ID_CLASECLIENTE2	INTEGER

Esta tabla almacena datos personales de los clientes de la empresa. La llave primaria es el campo *CLIENTEID,* a pesar de que existe el campo para el vendedor ningún registro almacena algún valor en este campo, por lo que se deduce que el sistema no lleva control sobre la cartera de un vendedor. Entre las llaves foráneas de la tabla se encuentra la del campo *ID_TIPO_COBRO* que hace referencia a la llave primaria de la tabla de tipos de cobro que se describe a continuación.

2. Tabla de tipos de cobro. El nombre de la tabla en la base de datos es: *CACC0TCC*, los campos que la componen los presento en el siguiente cuadro:

Cuadro 3: Campos de la tabla de tipos de cobro *CACC0TCC*

Nombre del campo	Tipo de dato
ID_TIPO_COBRO_CLI	INTEGER
DIAS_PLAZO	INTEGER
DESCRIPCION	VARCHAR(50)

La llave primaria de la tabla es *ID_TIPO_COBRO_CLI*, y los nombres de los campos son suficientes para explicar que información que guardan.

3. Tabla de empleados. El nombre de la tabla en la base de datos es: *EMPLMAS*, ésta incluye los datos personales y contables de los empleados de la empresa, entre los que se incluyen los asesores de ventas. La estructura de la tabla tiene 65 campos, de los cuales 26 no se usan. Por la cantidad de campos, no presento el cuadro con los nombres, sin embargo la importancia de la tabla radica en que la llave primaria *EMPID* es llave foránea en la tabla *PEDIDOS* para asociar un pedido a un asesor de ventas.

4. Tabla de proyectos. El nombre de la tabla en la base de datos es *PROYECTO*, los campos que la componen los presento en el siguiente cuadro:

Cuadro 4: Campos de la tabla *PROYECTO*

Nombre del campo	Tipo de dato
PROYECTOID	INTEGER
CODIGO	VARCHAR(20)
NOMBRE	VARCHAR(50)
LOCALIZACION	VARCHAR(100)
USOPRODUCTO	VARCHAR(20)
COTIZACIONID	INTEGER
FECHAAPROBADO	DATE
FECHAINI	DATE
FECHAFIN	DATE

Continuación cuadro 4:

ESTADOID	SMALLINT
CLIENTEID	INTEGER
COSTOESTIMADO	NUMERIC(15,2)
COSTO	NUMERIC(15,2)
PCOMPLETO	NUMERIC(15,2)
PRECIO	NUMERIC(15,2)
COBRADO	NUMERIC(15,2)
OBSERVACIONES	TEXT

Es importante notar que solamente cuatro campos almacenan información. La llave primaria de la tabla es *PROYECTOID,* un identificador correlativo. Existe una llave foránea a la tabla de clientes: *CLIENTEID*, un cliente puede tener asociados varios proyectos.

5. Tabla de productos. El nombre de la tabla en la base de datos es *PRODUCTO*, los campos que la componen los presento en el siguiente cuadro:

Cuadro 5: Campos de la tabla *PRODUCTO*

Nombre del campo	Tipo de dato
PRODID	INTEGER
NOMBRE	VARCHAR(50)
EMPRESAID	INTEGER
SUBGRUID	INTEGER
ALTERNO	VARCHAR(20)
EXISTENCIA	NUMERIC(15,2)
COSTO	NUMERIC(15,2)
COSTULT	NUMERIC(15,2)
PRECIO	NUMERIC(15,2)
MINIMO	NUMERIC(15,2)
MAXIMO	NUMERIC(15,2)
SEVENDE	VARCHAR(1)
VOLUMENU	NUMERIC(15,4)
UNIDAD	VARCHAR(10)
LMINIMA	NUMERIC(15,2)
LMAXIMA	NUMERIC(15,2)
CENTROID	INTEGER
COSTOVENTA	INTEGER
PRECIOCALC	INTEGER
PGANANCIA	INTEGER
ANCHO	NUMERIC(15,4)
ALTO	NUMERIC(15,4)
NOMBRE2	VARCHAR(50)
FECHACPR	DATE
ACTIVO	CHAR(1)

La tabla almacena información sobre los productos que se venden o compran en la empresa, esto lo determina el campo *SEVENDE* que puede ser 1 ó 0 respectivamente. Guarda la unidad en la que se despacha el producto, el nombre de venta y los costos y precios asociados. Tiene llave foránea a la tabla de subgrupos de productos (campo *SUBGRUPID*) para clasificar cada producto dentro de una subcategoría de una categoría mayor. La llave primaria es el campo *PRODID.*

6. Tabla de grupos de productos. El nombre de la tabla en la base de datos es *GRUPO,* contiene las categorías principales de los productos que se manejan dentro de la empresa. La llave primaria es el campo *GRUPID* y el único campo relevante, además de la llave, es el *NOMBRE* del grupo.

7. Tabla de subgrupos de productos. El nombre de la tabla en la base de datos es *SUBGRUP*, contiene los subgrupos de los grupos de productos de la tabla anterior. La llave primaria es el campo *SUBGRUPID*, al que se hace referencia en la tabla productos (cada producto pertenece a un subgrupo). Tiene un campo con el *NOMBRE* del subgrupo y para asociar el subgrupo a un grupo el campo *GRUPOID* es una llave foránea a la tabla GRUPO, un grupo puede tener muchos subgrupos.

8. Tabla de pedidos. El nombre de la tabla en la base de datos es *PEDIDO*, los campos que la componen los presento en el siguiente cuadro:

Cuadro 6: Campos de la tabla *PEDIDO*

Nombre del campo	Tipo de dato
PEDIDOID	INTEGER
PROYECTOID	INTEGER
REFERENCIA	VARCHAR(10)
FECHAINI	DATE
FECHAFIN	DATE
COSTO	NUMERIC(15,2)
COSTOESTIMADO	NUMERIC(15,2)
ESTADOID	SMALLINT
OBSERVACIONES	TEXT
VENDEDOR	INTEGER
DESPACHADO	BOOL
TOTPRODUCTO	NUMERIC(15,2)
TOTOTROS	NUMERIC(15,4)
TOTMOLDELK	NUMERIC(15,2)
TRANSPORTE	NUMERIC(15,2)
MONTAJE	NUMERIC(15,2)
DESCUENTO	NUMERIC(15,2)
DESCUENTOTRANSPORTE	NUMERIC(15,2)
DESCUENTOMONTAJE	NUMERIC(15,4)

Continuación cuadro 6:

DESCUENTOOTROS	NUMERIC(15,4)
IDRUBRO	INTEGER
TRANSPORTEIDRUBRO	INTEGER
MONTAJEIDRUBRO	INTEGER
PUEDEDESPACHAR	CHAR(1)
OTROSIDRUBRO	INTEGER

La tabla de *PEDIDO,* como su nombre lo indica, guarda información de los pedidos, la llave primaria de la tabla es *PEDIDOID,* este campo es al que se hace referencia en las demás tablas de la base de datos a través de llaves foráneas. Sin embargo, para manejo en la empresa el campo que identifica un pedido es *REFERENCIA*, que consta de nueve dígitos: los primeros cuatro identifican el año en el que se creó el pedido, luego un guión seguido de cuatro dígitos para un número correlativo, por ejemplo una referencia es 2005-0017 que representa el pedido 17 del año 2005. La referencia es la que se utiliza para que el asesor de ventas identifique el pedido específico de un cliente. El pedido se encuentra asociado a un proyecto y un vendedor (Existen llaves foráneas a las tablas *EMPLMAS y PROYECTOID*), además la tabla contiene información sobre descuentos y transporte. El campo *PUEDEDESPACHAR* puede tomar valor de 1 ó 0 e indica si el pedido ha sido liberado o no respectivamente.

9. Tabla de órdenes de producción. El nombre de la tabla en la base de datos es *OP*, los campos que la componen los presento en el siguiente cuadro:

Cuadro 7: Campos de la tabla *OP*

Nombre del campo	Tipo de dato
OPID	INTEGER
PEDIDOID	INTEGER
REFERENCIA	VARCHAR(10)
FECHAINI	DATE
FECHAFIN	DATE
COSTO	NUMERIC(15,2)
COSTOESTIMADO	NUMERIC(15,2)
ESTADOID	SMALLINT
OBSERVACIONES	TEXT
FECHAENTREGA	DATE
CANTIDADCAMIONES	SMALLINT
CANTIDADPLATAFORMAS	SMALLINT

La tabla almacena los datos de las órdenes de producción. Al igual que la tabla de pedidos tiene una llave primaria para manejo dentro de la base de datos y es el campo *OPID,* y para manejo dentro de la empresa se utiliza el campo *REFERENCIA* con el mismo formato que el de la tabla *PEDIDO.* Una orden de producción está asociada a un pedido (el campo *PEDIDOID* es una llave foránea a la tabla de pedidos). El campo *FECHAENTREGA* almacena la fecha que el cliente solicitó le fuera entregado el pedido.

10. Tabla de detalle de orden de producción. El nombre de la tabla en la base de datos es *OPDET*, los campos que la componen los presento en el siguiente cuadro:

Cuadro 8: Campos de la tabla *OPDET*

Nombre del campo	Tipo de dato
LOPID	INTEGER
OPID	INTEGER
PRODID	INTEGER
APRODUCIR	NUMERIC(15,2)
PRODUCIDO	NUMERIC(15,2)
DESPACHADO	VARCHAR(1)
FECHAINI	DATE
FECHAFIN	DATE
TAMANO	NUMERIC(15,4)
COSTODIRECTO	NUMERIC(15,2)
ESTATUSID	SMALLINT
CANTORDENADA	FLOAT
DESPACHADOQTY	NUMERIC(15,2)
CANTDESPACHADO	NUMERIC(15,2)
PRECIO	NUMERIC(15,2)

La llave primaria de la tabla es el campo *LOPID*. La tabla guarda el detalle de las órdenes de producción, cada registro incluye el identificador de la orden de producción a la que se asocia el detalle (*OPID*), existen varios registros de la tabla que hacen referencia a una misma orden de producción, el identificador del producto (*PRODID*), el campo *APRODUCIR* que indica la cantidad pedida del producto, *PRECIO* indica el precio al que se despachará el producto y *DESPACHADOQTY* indica la cantidad que se ha despachado hasta el momento de ese producto respecto a la cantidad pedida. La tabla tiene llaves foráneas a la tabla de órdenes de producción y a la tabla de productos.

11. Tabla de cotizaciones. El nombre de la tabla de cotizaciones es *COTIZA*, tiene como llave primaria el campo *COTIZACIONID*, almacena los datos de un proyecto al que la empresa puede proveerle productos de construcción. La tabla cuenta con cuarenta y cinco campos, por lo que no presento la descripción de los mismos, la información importante es la referente a la ubicación del proyecto, el cliente, el área y la altura, la cantidad de camiones, los días que la cotización es válida, el anticipo, el vendedor, el descuento y el contacto.

12. Tabla de detalle de cotización. El nombre de la tabla en la base de datos es *COTIZADET*, los campos que la componen los presento en el siguiente cuadro:

Cuadro 9: Campos de la tabla *COTIZADET*

Nombre del campo	Tipo de dato
COTIZDETID	INTEGER
COTIZACIONID	INTEGER
PRODID	INTEGER
APRODUCIR	NUMERIC(15,2)
TAMANO	NUMERIC(15,4)
DESPACHADOQTY	NUMERIC(15,2)
PRECIOU	NUMERIC(15,4)
IDRUBRO	INTEGER

Esta tabla guarda el detalle del producto para una cotización, existe un registro en la tabla por cada producto de la cotización, el campo *PRODID* es el identificador del producto, *COTIZACIONID* es el identificador de la cotización, ambos campos son llaves foráneas a las tablas correspondientes. La cantidad cotizada del producto se guarda en el campo *APRODUCIR* y el campo *TAMANO* se usa para describir el tamaño del producto en caso de que no sea producto estándar, por último *PRECIOU* es el precio al que se está ofreciendo la unidad del producto.

13. Tabla de órdenes de despacho. El nombre de la tabla en la base de datos es *CAOD0DES*, los campos que la componen los presento en el siguiente cuadro:

Cuadro 10: campos de la tabla de órdenes de despacho *CAOD0DES*

Nombre del campo	Tipo de dato
ID_DESPACHO	INTEGER
DESPACHOREF	VARCHAR(15)
ID_CLIENTE	INTEGER
ID_PROYECTO	INTEGER
ID_EMPLEADO	INTEGER
FECHA_EMITE	DATE
FECHA_ENTREGA	DATE
DIRECCION	VARCHAR(75)
ID_EMPRESA	INTEGER
OBSERVACIONES	TEXT
TRANSPORTE	VARCHAR(20)

La llave primaria es el campo *ID_DESPACHO.* Al igual que en la tabla de pedidos, se maneja también el campo *REFERENCIA*, con el mismo formato. Son muy importantes cuatro campos: la fecha en la que se emita la orden de despacho: *FECHA_EMITE*; El campo *FECHA_ENTREGA* que guarda la fecha en la que tiene que despacharse el pedido; *DIRECCION* que guarda la dirección en donde se realiza el despacho; y el campo *TRANSPORTE* que indica si se usará transporte de Precon o transporte del cliente. Los demás campos no deberían existir en la tabla ya que la siguiente tabla que se describe relaciona el despacho

a un pedido y el pedido ya incluye el identificador del cliente, vendedor y proyecto, campos que también se incluyen en ésta tabla.

14. Tabla de órdenes de despacho por pedido. El nombre de la tabla en la base de datos es *CAOD0DESDET*, relaciona un despacho a un pedido, un pedido puede tener varios despachos es por ello que se utiliza una tabla sólo para especificar esta relación. La tabla únicamente tiene tres campos: la llave primaria que es un número correlativo, el campo *ID_DESPACHO* que es una llave foránea a la tabla de despachos y *ID_PEDIDO* una llave foránea a la tabla de pedidos.

15. Tabla de movimientos. El nombre de la tabla en la base de datos es *MASMOV*, los campos que la componen los presento en el siguiente cuadro:

Cuadro 11: Campos de la tabla de movimientos *MASMOV*

Nombre del campo	Tipo de dato
MASMOVID	INTEGER
EMPRESAID	INTEGER
TIPMOVID	INTEGER
BODEGID1	INTEGER
BODEGID2	INTEGER
MOVID	INTEGER
FECHA	DATE
OBSERV	TEXT
FINCAID	INTEGER
PROVEEDORID	INTEGER
NUMDOC	VARCHAR(20)
FACTURA	VARCHAR(20)
OCOMPRA	VARCHAR(20)
REQUISICION	VARCHAR(20)
GUIA	VARCHAR(20)
CLIENTEID	INTEGER
ID_DESPACHO	INTEGER
ID_TRANSPORTE	INTEGER
PEDIDOID	INTEGER
PARCIAL	VARCHAR(1)
ENTREGANUMERO	SMALLINT
TOTALNUMEROENTREGAS	SMALLINT
HORADESPACHO	DATE
TRANSPORTEEX	NUMERIC(15,2)
FECHA_ACTUALIZADO	DATE

Esta tabla guarda los movimientos del producto en las diferentes bodegas de Precon, la forma de saber si el movimiento del producto es una venta es por el tipo de movimiento especificado en el campo *TIP-MOV*. La llave primaria de la tabla es *MASMOVID*, se guarda también la fecha del movimiento, el número

de documento al que este asociado, el identificador del despacho, el pedido, el número de entrega y la hora del despacho.

16. Tabla de tipos de movimiento. El nombre de la tabla en la base de datos es *TIPMOV*, los campos que la componen los presento en el siguiente cuadro:

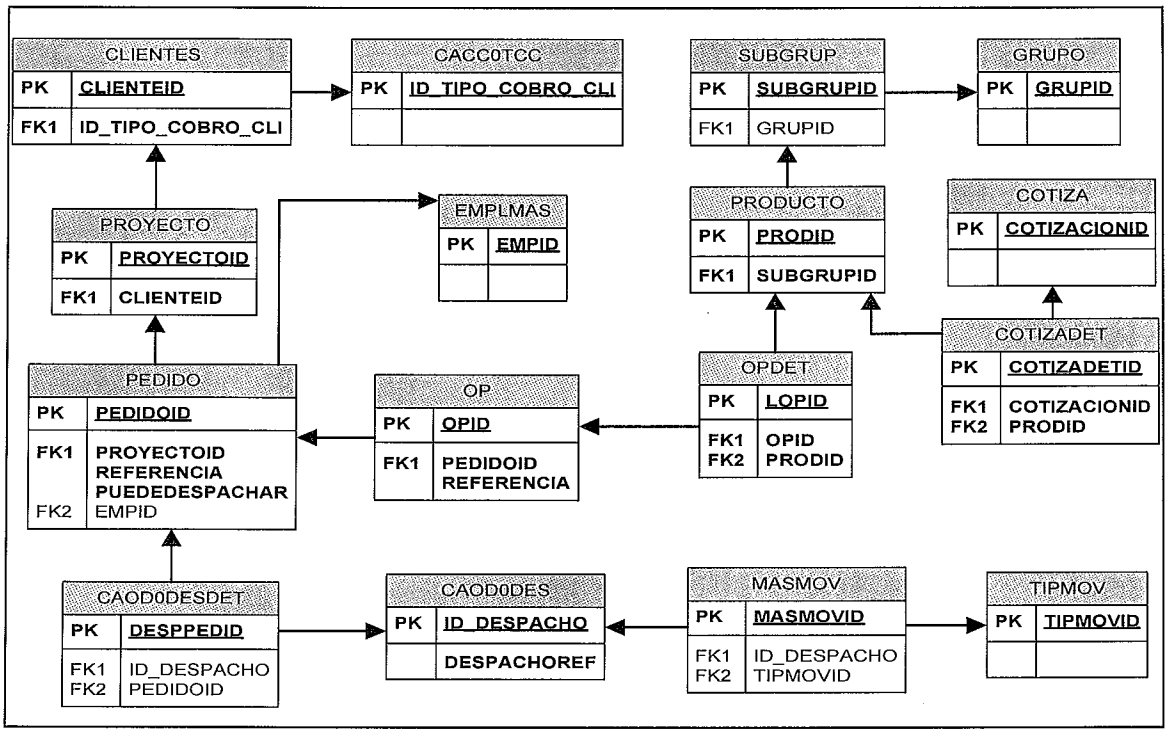
Cuadro 12: Campos de la tabla de tipos de movimiento *TIPMOV*

Nombre del campo	Tipo de dato
TIPMOVID	INTEGER
ESTRASLA	VARCHAR(1)
TRASLADARA	INTEGER
SIGNO	SMALLINT
NOMBRE	VARCHAR(50)
CCONTA	VARCHAR(20)
CONFINCA	VARCHAR(1)
CONPROVE	VARCHAR(1)
CONFACTURA	VARCHAR(1)
CONENVIO	VARCHAR(1)
CONOCOMPRA	VARCHAR(1)
CONREQUISICION	VARCHAR(1)
CONGUIA	VARCHAR(1)
DEPRODUCCION	VARCHAR(1)
APRODUCCION	VARCHAR(1)
ORDEN	SMALLINT
CONPLANPROD	VARCHAR(1)
CONCENTROCOS	VARCHAR(1)
CONCLIENT	VARCHAR(1)
TAG	INTEGER
VISIBLE	VARCHAR(1)
CONDESPACHO	VARCHAR(1)
CONTRANSPORTE	VARCHAR(1)
NEXTNODOC	INTEGER

Esta tabla especifica los tipos de movimiento que pueden tener los productos en las bodegas de Precon. *TIPMOVID* es la llave primaria; *SIGNO* puede ser 1 ó 0, según sume o reste al inventario respectivamente; *NOMBRE* es el campo con la descripción del movimiento; en el caso de que sea venta los campos *CONFACTURA, CONENVIO, CONCLIENT, CONDESPACHO y CONTRANSPORTE* deben ser 1.

La figura 12 presenta la relación entre las tablas relevantes en el proceso de ventas dentro de la base de datos actual.

Figura 12: Relación entre tablas de la base de datos actual



X. PROBLEMÁTICA DEL RECURSO DE INFORMACIÓN EN EL DEPARTAMENTO DE VENTAS

Existen dos casos que generan problemas que involucran al recurso de información en el departamento de ventas, estos casos son los siguientes:

1. En el primer caso se utilizan los módulos del sistema de operaciones para ingresar cotizaciones y pedidos y generar órdenes de producción y órdenes de despacho con sus respectivos clientes y vendedores. Sin embargo, la información que es almacenada no es utilizada para otros propósitos. En este caso, ya se cuenta con una parte del sistema de información; la de capturar, procesar y almacenar los datos, pero no se presenta la información resultante a los involucrados. Entre los problemas que se generan de este caso se encuentran los siguientes:

a. Un vendedor no puede conocer el estado de su pedido de forma inmediata, la planta y las oficinas no se encuentran en el mismo lugar, para que el vendedor conozca el estado de su pedido debe llamar a la planta y según la información que se le brinde debe contactar después al jefe de despachos. Por otra parte el gerente no tiene forma de conocer la cantidad de pedidos pendientes de despachar, la cantidad de pedidos entregados a tiempo y el tiempo que tarda un pedido en planta y en ser despachado.

b. El sistema de operaciones únicamente es utilizado para emitir cotizaciones de un producto (LOSAS), esto es debido a que el formato del documento de cotización que genera el sistema no es útil para los demás productos estándar que se venden (en el análisis y diseño presentaré el formato emitido por el sistema y los formatos requeridos para los demás productos). Por esta razón cada vendedor utiliza actualmente un formato de cotización y lleva control sobre cotizaciones aceptadas y rechazadas propios, el gerente no es informado sobre las cotizaciones que no se lograron, únicamente lleva control sobre las ventas cerradas, de esta forma no puede medir la cantidad de ventas que la empresa perdió, la cantidad de cotizaciones realizadas por un empleado y la relación entre ventas cerradas y cotizaciones rechazadas. A esto hay que agregarle que luego de que una venta es cerrada el vendedor entrega su cotización a la secretaria y ella debe ingresarla en el sistema, de forma que el proceso de crear la cotización es realizado dos veces: la primera por el vendedor y la segunda por la secretaria.

c. Toda la información ingresada por la secretaria o por el asistente administrativo del departamento, es ingresada una segunda vez por el asistente administrativo en una base de datos de Excel utilizada para generar los reportes y resúmenes de ventas del mes. No existe un método para alimentar esta base de datos desde la base de datos del sistema de operaciones, de forma que no se introduzcan los datos manualmente por segunda vez.

2. En el segundo caso el sistema actual no incluye algún módulo que tome en cuenta un proceso específico. En este caso no se cuenta con ninguna parte del sistema de información, en el análisis y diseño deben tomarse en cuenta todas las etapas desde capturar los datos hasta permitir el acceso a la información resultante. Entre los procesos que no se toman en cuenta se encuentran los siguientes:

a. No se cuenta con una aplicación para la programación de visitas a clientes de asesores o de clientes a asesores. El gerente no es informado de las próximas visitas de un vendedor, de los resultados de una visita o de una secuencia de visitas a un mismo cliente.

b. Tampoco se cuenta con una aplicación para coordinar las tareas y tiempos involucrados en una licitación.

XI. SOLUCIÓN PROPUESTA

Se propone una solución que incluye cinco aplicaciones que utilizan la base de datos actual y una base de datos paralela para presentar información a los usuarios.

El motivo por el que se utiliza una base de datos paralela y no simplemente se agreguen campos o tablas a la existente, es que la empresa tiene autorización para escribir y leer la base de datos actual, más no de cambiar la estructura de la misma. Por ello se propone que la información nueva sea almacenada en una base de datos paralela Internase/Firebird, en la cual la información necesaria sea replicada desde la base de datos original para que ambas bases de datos estén sincronizadas y coordinadas.

Una aplicación puede escribir en una de las bases de datos, leer información de la otra o leer información de ambas bases de datos y combinarla para presentar resultados útiles. Esto dependerá de la función que la aplicación realice. De esta forma existirán aplicaciones que utilicen ambas bases de datos o aplicaciones que únicamente utilicen una de las dos.

Las cinco aplicaciones que incluye la solución propuesta son:

A. Control de pedidos

Constituye una aplicación en línea en la que un asesor de ventas puede consultar el estado de uno o más pedidos, el gerente de ventas puede ver el estado de todos los pedidos pendientes de despachar y resultados de los pedidos ya despachados: la cantidad de pedidos despachados a tiempo, pedidos despachados fuera de tiempo y tiempo que tarda un pedido en ser despachado. En planta pueden ver detalle de pedidos pendientes de despachar y pedidos pendientes de liberar. La aplicación proporciona el medio para actualizar el estado de un pedido y la programación del despacho, función que será realizada por un usuario de planta y un usuario del departamento de despachos.

Esta aplicación utilizará la base de datos actual para leer la información que se ingresa de un pedido y la base de datos paralela para almacenar y leer las fechas relacionadas al estado del pedido.

B. Control de cotizaciones

Lo primero que debe cambiarse en cuanto al manejo de las cotizaciones es que el asesor de ventas las ingrese directamente en el sistema sin imprimirla, ya que el formato del documento impreso es lo que no se utiliza. El estado default de una cotización siempre es rechazado, pero el sistema no ofrece la capacidad de

cambiar este estado cuando una cotización fue aceptada. El control de cotizaciones únicamente presentará al asesor el listado de cotizaciones ingresadas en el sistema y permitirá imprimirlas en el formato correcto o cambiarles el estado si han sido aceptadas, ingresando como garantía el número de recibo provisional del anticipo del pedido. El gerente podrá ver las cotizaciones rechazadas de un asesor específico, la relación entre ventas cerradas y cotizaciones perdidas por vendedor y de forma general. También será el único que puede cambiar el estado de una cotización sin número de recibo provisional en caso que se autorice el despacho de un pedido sin anticipo.

C. Alimentador de base de datos secundaria

La función de esta aplicación será alimentar la base de datos de Excel que se utiliza para generar los reportes y resúmenes mensuales de ventas. La base de datos se actualizará una vez al día. De acuerdo a una consulta de varias tablas de la base del sistema de operaciones se generarán las columnas de la base de datos en Excel.

D. Control de visitas

El control de visitas es un proceso que no fue tomado en cuenta en el sistema actual. La aplicación debe proveer a un vendedor la capacidad de programar sus visitas en una fecha y hora, con un cliente y un propósito específico, así como de guardar los resultados de una visita para que sean analizados por el gerente de ventas. El asesor podrá ver únicamente sus visitas programadas, recalendarizarlas o cancelarlas e ingresar el resultado de alguna visita. El gerente podrá ver las visitas programadas de todos los asesores o clientes y la secuencia de visitas realizadas de un cliente específico, con el propósito de que analice el proceso que se siguió con el cliente.

E. Control de licitaciones

Otro proceso que no se toma en cuenta en el sistema actual son las licitaciones, la empresa lícita para ser la constructora o proveedora de materiales de un proyecto de construcción. El coordinador de licitaciones recibe un documento en donde se detalla lo que el proyecto requiere, la aplicación para control de licitaciones deberá permitir la creación, modificación y eliminación de licitaciones asociadas a un documento inicial. Para cada licitación se podrán crear un conjunto de tareas a las que se les asignará una fecha límite de entrega, un porcentaje de avance y un responsable; por cada tarea puede existir además un conjunto de documentos asociados. Al ser creada la licitación se enviará un correo a las personas responsables de las diferentes tareas indicando que tienen una tarea en una nueva licitación, el personal involucrado podrá ver únicamente las tareas que le corresponden de una licitación y subir los documentos en caso que su tarea

tenga asociados, y el único que puede ingresar el porcentaje de avance de una tarea es el coordinador de la licitación, de acuerdo a lo que ha recibido de cada persona. El coordinador será el responsable de la creación, modificación y eliminación de una licitación y sus respectivas tareas, éste además ingresará los resultados de la licitación y si el proyecto fue adjudicado a la empresa o no. Esta también será una aplicación en línea.

XII. ANÁLISIS Y DISEÑO DE LAS APLICACIONES

A. Plataforma y conexión a las bases de datos

Como mencioné con anterioridad, la base de datos actual es tipo Interbase/Firebird, debido a que ya se cuenta con el servidor de Firebird y a la compatibilidad de tipos para combinar la información de las dos bases de datos, se decidió de que la base de datos paralela también fuera del tipo Interbase/Firebird.

Cuatro de las aplicaciones propuestas son aplicaciones en línea; la empresa cuenta actualmente con dos licencias de Visual Studio .Net Enterprise y un Servidor de Windows 2000 para publicar las aplicaciones, por ello las aplicaciones serán de tipo ASP desarrolladas sobre la plataforma de Visual Studio .Net 2003.

Para la conexión a las bases de datos Firebird desde aplicaciones de .Net se utilizará el *Firebird ADO.NET Data Provider* que provee una implementación de alto rendimiento para acceder a bases de datos Firebird sin necesidad de instalar el cliente de Firebird en las máquinas cliente de la aplicación.

La información devuelta por el proveedor de *Firebird ADO.NET* será manejada en la aplicación a través de *DataTables* y *DataSets*.

Un objeto *DataTable* representa una tabla de datos relacionales en memoria, utiliza un conjunto de objetos *DataColumn* permitiendo definir *Constraints* y llaves primarias. Las tablas son agrupadas en colecciones denominadas *DataSet* que permiten además definir relaciones entre sus tablas y vistas.

En el portal msdn se define un dataset como:

> *«Una representación residente en memoria de datos que proporciona un modelo de programación relacional coherente independientemente del origen de datos.»*

Para ampliar la información sobre los objetos *DataTable y DataSet* ver apéndice D.

Las clases que provee *Firebird ADO.NET Data Provider* para el manejo de la conexión, lectura y escritura a la base de datos son:

1. FbConnection. Un objeto de esta clase representa la conexión a un servidor de Firebird. La propiedad fundamental de esta clase es *ConnectionString* que representa la cadena utilizada para abrir

conexión a una base de datos, la cadena específica la base de datos, el servidor en el que se encuentra la base de datos, el tamaño del paquete, la versión del servidor, el usuario y contraseña.

Los métodos de mayor relevancia de la clase son:

- Open: este método abre una nueva conexión a una base de datos con los parámetros especificados en la cadena de conexión (*ConnectionString*).
- Close: cierra la conexión a la base de datos.
- CreateCommand: crea un objeto del tipo fbCommand asociado al objeto fbConnection que llama al método.

2. FbComand. Un objeto de esta clase representa una sentencia de SQL o un *Stored Procedure* que se ejecuta sobre una base de datos Firebird, especificada por un objeto *fbConnection* asociado al objeto *fbCommand.*

Provee los siguientes métodos para ejecutar la sentencia:

- ExecuteReader: ejecuta una sentencia que regresa un conjunto de filas, como resultado de una consulta en la base de datos.
- ExecuteNonQuery: ejecuta una sentencia de tipo INSERT, DELETE Y UPDATE, que no regresan filas.
- ExecuteScalar: devuelve un solo valor de una base de datos.

Las propiedades más importantes de la clase son:

- CommandText: representa el texto de la sentencia a ejecutarse.
- CommandType: indica de que forma debe interpretarse el texto de la sentencia, como un stored procedure o como un texto plano que se ejecuta.
- Parameters: es una colección de objetos del tipo *FbParameter*, cada uno de los cuales representa un parámetro que la sentencia recibe de la aplicación antes de ser ejecutada, cada parámetro tiene un tipo y tamaño específico (propiedades *DbType y Size*) y la propiedad *Value* guarda el valor que se recibe de la aplicación.

3. FbDataAdapter. Cada objeto de esta clase representa un conjunto de instrucciones y una conexión a una base de datos. Un FbDataAdapter Representa un puente entre un DataSet y FirebirdSQL para leer y guardar datos.

Tiene cuatro tipos de instrucciones:

- SelectCommand: para realizar consultas en una fuente.
- UpdateCommand: para actualizar los datos de una fuente.
- DeleteCommand: para eliminar datos de una fuente.
- InsertCommand: para insertar datos en una fuente.

Los dos métodos principales son:

- Fill: ejecuta el *SelectCommand* y llena un dataset o una tabla con las filas resultantes de la ejecución.
- Update: este método ejecuta la sentencia de inserción, actualización o eliminación para cada fila insertada, eliminada o actualizada en la tabla o el DataSet.

El método *Fill* actualiza los datos en el DataSet o la tabla para que coincidan con los de la fuente, por el contrario, el método *Update* actualiza los datos en la fuente para que coincidan con los datos en el DataSet o la tabla.

4. FbError. Un objeto de esta clase agrupa la información relevante de un error regresado por Firebird. Entre sus propiedades se encuentran el mensaje de error, el número de línea y el número de error (propiedades *Message, LineNumber y Number* respectivamente).

B. División de la arquitectura de las aplicaciones en tres capas funcionales

Se realizará la división de la arquitectura de las aplicaciones en línea en tres capas funcionales:

- Capa de datos.
- Capa de aplicación.
- Capa de presentación.

1. Capa de datos. El acceso a los datos que se capturan, procesan, almacenan y presentan en las aplicaciones es a través de esta capa.

En las aplicaciones propuestas la capa de datos está conformada por:

- La base de datos del sistema actual en el servidor de Firebird.
- La base de datos paralela en el servidor de Firebird.

- Una clase llamada *dbObjects,* compuesta por objetos de las clases de *Firebird ADO.NET Data Provider.* DbOjects permitirá conectarse a bases de datos de tipo Interbase/Firebird y ejecutar instrucciones *select, insert, delete y update* sobre las tablas de la base de datos. La especificación de la clase se presenta a continuación:

Figura 13: Clase dbObjects

dbObjects
-FbConnection conexion -FbDataAdapter adapter
+dbObjects() +ConectarseADB(in servidor: string, in basedatos: string, in usuario: string, in contrasena: string):int +EjecutarSelect(in sentencia: string): dataset +EjecutarNonQuery(in setencia: string): int +EjecutarScalar(in sentencia: string):int +GuardarCambios(in sentenciaInsert: string, in sentenciaDelete:string, in sentenciaUpdate: string, in dataset: DataSet) +CerrarConexión():int

- Los objetos privados *conexion* y *adapter* son utilizados para almacenar la información utilizada en los métodos de la clase.

- El método ConectarseADB utiliza el objeto *conexion* al cual le asigna en la propiedad *ConnectionString* los valores que recibe como parámetros y luego llama al método *open* de *conexion.* La función devuelve '1' si la conexión a la base de datos se realizó, de lo contrario devuelve '0'.

- El método EjecutarSelect recibe como parámetro una cadena con la sentencia que debe ejecutar, crea un objeto FbCommand al que le asigna en la propiedad *Connection* el objeto *conexion* de la clase y en la propiedad *CommandText* la cadena de sentencia recibida. Al objeto *adapter* de la clase le asigna en la propiedad *SelectCommand* el objeto FbCommand creado y llena un dataset llamando al método *Fill* de *adapter.* Devuelve el dataset generado.

- El método EjecutarNonQuery crea un objeto de tipo FbCommand, le asigna en la propiedad *Connection* el objeto *conexion* de la clase y en *CommandText* asigna la cadena de sentencia que recibe como parámetro; Llama al método *ExecuteNonQuery* del objeto de tipo FbCommand y devuelve el número de filas afectadas por la ejecución de la sentencia.

- El método EjecutarScalar crea un objeto de tipo FbCommand, le asigna en la propiedad *Connection* el objeto *conexion* de la clase y en *CommandText* asigna la cadena de sentencia que recibe

como parámetro; Llama al método *ExecuteScalar* del objeto de tipo FbCommand y devuelve el resultado.

- El método GuardarCambios crea tres objetos de tipo FbCommand y en cada uno asigna en la propiedad *Connection* el objeto *conexion* de la clase; en uno de los objetos guarda en la propiedad *CommandText* la cadena *sentenciaInsert,* en el otro *sentenciaDelete* y *sentenciaUpdate* en el tercero. Asigna en las propiedades *InsertCommand, DeleteCommand* y *UpdateCommand* del objeto adapter de la clase los objetos FbCommand creados, según las sentencia que almacenen. Llama al método *Update* de adapter y le envía como parámetro el dataset que recibió; de esta forma realiza las inserciones, eliminaciones y actualizaciones del dataset en la base de datos.

- El método CerrarConexion únicamente llama al método *close* del objeto conexión para cerrar la conexión a la base de datos que el objeto dbObject representa.

2. Capa de aplicación. La capa de aplicación es donde se localiza la lógica del negocio. Ésta recibe las solicitudes del usuario a través de la capa de presentación y se encarga de cumplirlas recurriendo si es necesario a la capa de datos.

En las aplicaciones propuestas la capa está conformada por un conjunto de clases que en base a los parámetros que reciben y los datos que procesan sus métodos, preparan las sentencias de tipo *select, update, delete e insert* para una trabajar sobre una tabla de una de las bases de datos. Cada clase de esta capa posee un objeto de la clase *dbObjects* para ejecutar las sentencias que sus métodos preparan.

3. Capa de presentación. Esta capa es la encargada de la interacción con el usuario y tradicionalmente se conoce como interfaz de usuario. Se realizan las validaciones de los datos ingresados por el usuario tanto sintáctica como lógicamente.

En las aplicaciones propuestas se utilizarán para el desarrollo de las páginas que componen las aplicaciones, los controles de ASP.NET para aplicaciones web.

Las tres capas deben ser independientes, los cambios en una de las capas no deben afectar el funcionamiento ni implicar modificaciones en alguno de los elementos de las otras capas. Por ejemplo, si por cualquier razón se migrase el repositorio de información y se alojase en una base de datos diferente, únicamente sería necesario modificar la capa de datos.

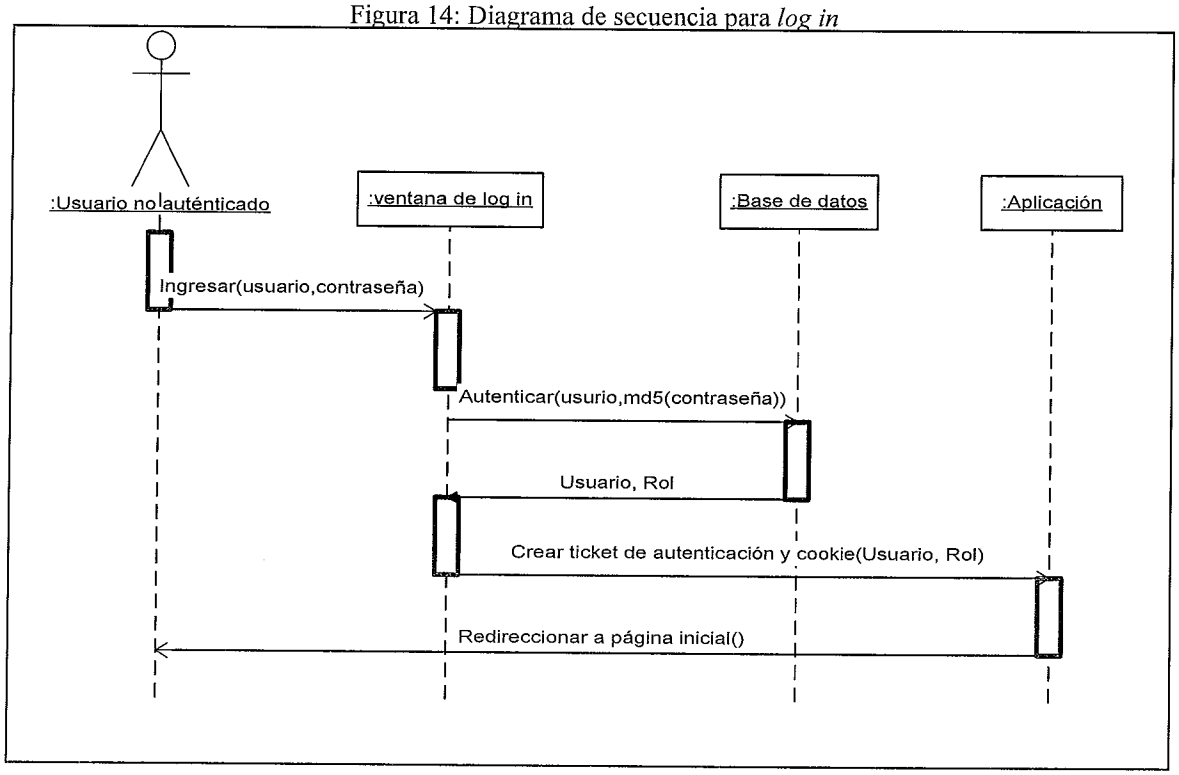
C. Log in

Para ingresar al sistema el usuario debe proveer como identificador de usuario su código de empleado y una contraseña. El sistema autenticará al usuario, verificando que los datos proporcionados existan asociados en la base de datos paralela.

En esta base de datos las contraseñas se almacenan encriptadas con una función *hash* de una dirección implementada por el algoritmo md5. Para comparar si la contraseña y nombre de usuario ingresados por el usuario son correctos, la aplicación debe encriptar con el mismo algoritmo la contraseña ingresada y enviarla al servidor junto con el nombre del usuario para verificar si existe el usuario en la tabla de usuarios y si la contraseña es igual a la almacenada en la base de datos. Si estas dos condiciones se cumplen el usuario es autenticado y se devuelve su rol.

En .NET se utilizará la autenticación de formularios (apéndice C) para la cual, luego de que se verificaron los datos ingresados, se debe de crear un ticket de autenticación el cual almacena entre otras cosas el nombre de usuario y el rol del usuario. Este ticket es encriptado y utilizado para crear la *cookie* del usuario. Por último el usuario es redireccionado a la página principal de la aplicación a la cual se le envía la cookie del usuario actual.

Figura 14: Diagrama de secuencia para *log in*



La clase para el manejo de la sesión se muestra en la figura 15.

Figura 15: Clase session

Sesión
+AutenticarUsuario(in usuario: string, in contrasena: string):int +CrearCookie(in usuario: string, in Rol: int, in TiempoExpiracion: int): int +TerminarSesion()

El método *AutenticarUsuario* recibe un usuario y una contraseña y lo autentica con la información en la base de datos. Si el usuario es autenticado devuelve un entero positivo que es el identificador del rol, de lo contrario devuelve un valor negativo.

El método *CrearCookie* recibe un usuario, su rol y un tiempo de expiración para crear el ticket de autenticación con estos parámetros, encriptarlo y crear la cookie utilizando este ticket, la función retorna la cookie de la sesión.

El método *TerminarSesion* termina la sesión utilizando la función *SignOut* que la autenticación de formularios de .NET ofrece.

D. Roles y módulos

Los roles que un usuario puede tener son:

- Usuario de gerencia.
- Usuario de producción.
- Usuario de despachos.
- Usuario asesor de ventas.
- Usuario coordinador de licitaciones.
- Usuario técnico.

Los módulos que de la solución son:

- Control de pedidos.
- Control de cotizaciones.
- Control de visitas.
- Control de licitaciones.

Los tipos de usuario que pueden utilizar cada módulo son:

- Control de pedidos: Usuario de gerencia, asesor de ventas, producción y despachos.
- Control de cotizaciones: Usuario asesor de ventas y usuario de gerencia.
- Control de visitas: Usuario asesor de ventas y usuario de gerencia.
- Control de licitaciones: Usuario coordinador de licitaciones, usuario técnico y usuario de gerencia.

Dependiendo del tipo de usuario se presentarán las opciones dentro de cada módulo.

E. Control de pedidos

1. Propósito. Presentar a los usuarios la información almacenada en el sistema de operaciones sobre pedidos pendientes de liberar, pendientes de despachar y despachados, y permitir complementar esta información, a los usuarios autorizados, para llevar control sobre el estado del pedido y las fechas en las que se inician las diferentes etapas. Presentar también resúmenes de pedidos despachados fuera de tiempo, tiempo promedio de entrega de un pedido y promedio de pedidos despachados diaria y mensualmente.

2. Diagramas de casos de uso. Los diagramas de caso de uso para esta aplicación se presentan a continuación:

Figura 16: Diagrama de casos de uso para usuario tipo asesor en la aplicación *control de pedidos*

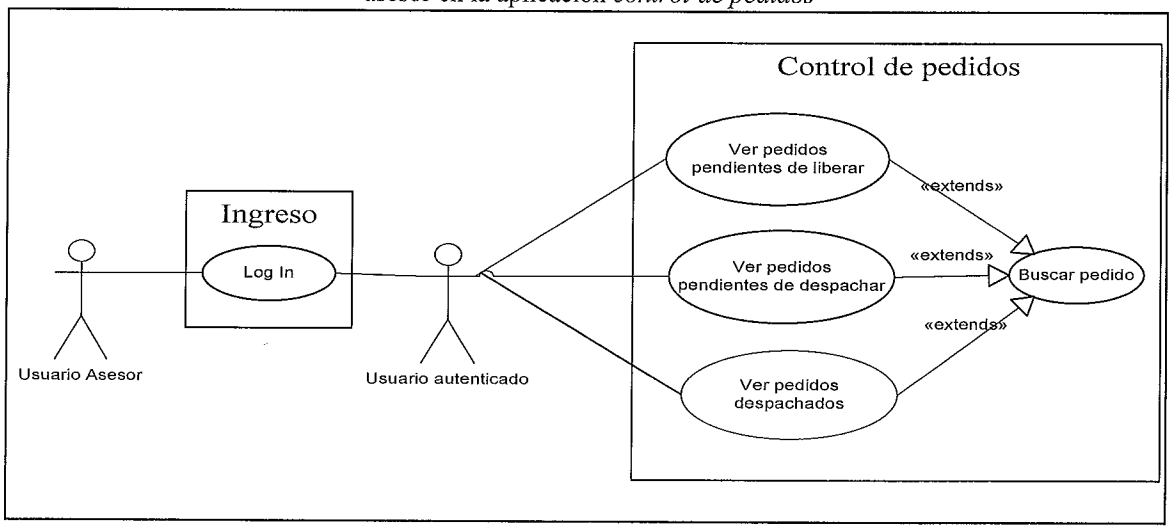


Figura 17: Diagrama de casos de uso para usuario tipo despachos en la aplicación *control de pedidos*

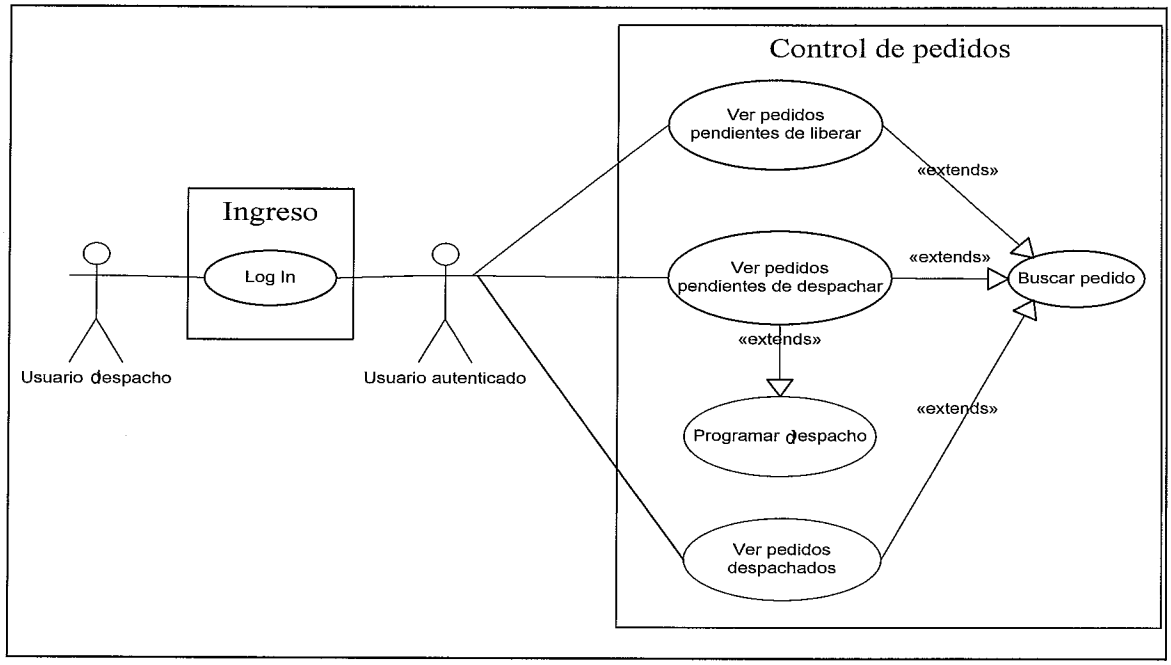


Figura 18: Diagrama de casos de uso para usuario tipo producción en la aplicación *control de pedidos*

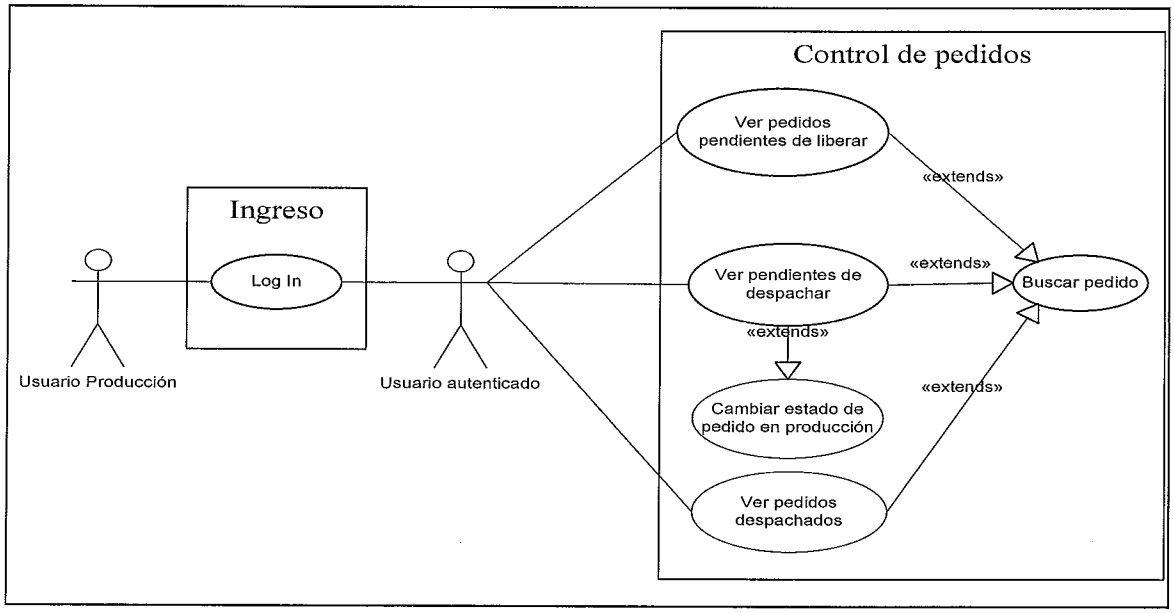
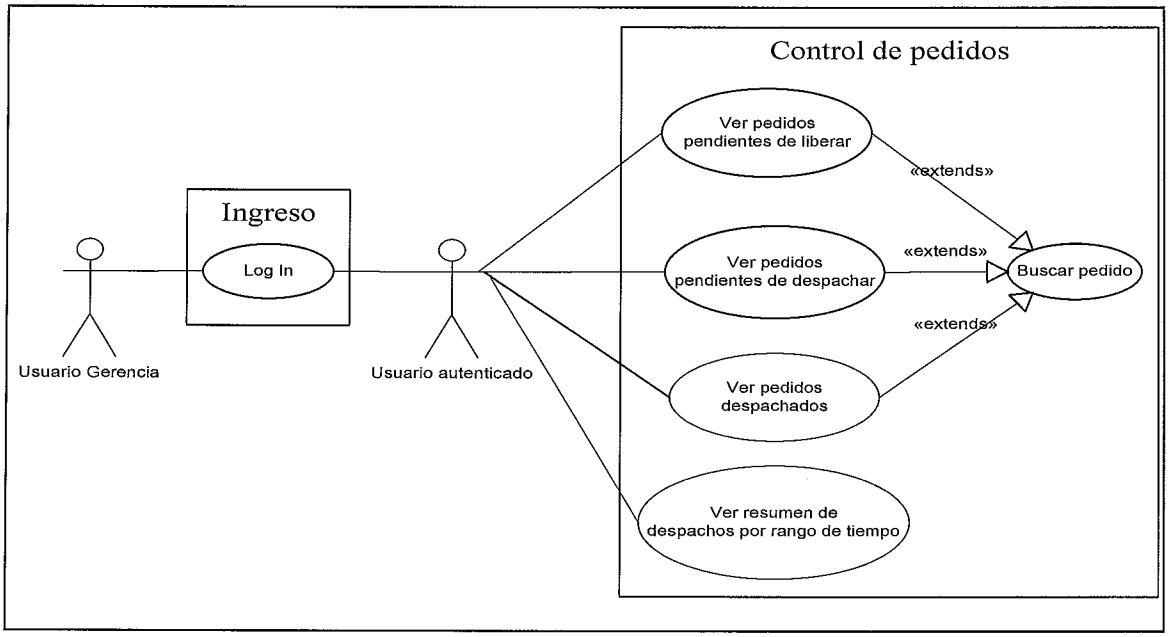


Figura 19: Diagrama de casos de uso para usuario de gerencia
en la aplicación *control de pedidos*



3. Documentos de casos de uso

Cuadro 13: Documento de caso de uso ver pedidos pendientes de liberar

Nombre:	Ver pedidos pendientes de liberar
Autor:	Rocío García Morales
Fecha:	24-09-2005

Descripción:

Presentará los pedidos que han sido ingresados al sistema, pero que aún no han sido liberados. La información general que se presenta por cada pedido y la tabla de la que se obtiene es:

- Referencia de pedido : tabla *PEDIDO*
- Referencia de orden de producción: tabla *OP*
- Cliente: tabla *PEDIDO, PROYECTO Y CLIENTES*
- Proyecto: tabla *PEDIDO Y PROYECTO*
- Vendedor: tabla *PEDIDO Y EMPLMAS*
- Fecha de creación: tabla *PEDIDO*

Además se presenta el detalle de cada pedido, las columnas que incluye el detalle son:

- Código de producto: tabla *OPDET* y *PRODUCTO*
- Nombre de producto: tabla *PRODUCTO*
- Cantidad pedida: tabla *OPDET*
- Tamaño: tabla *OPDET*

Toda la información es obtenida de la base de datos actual.

Actores: Los usuarios que pueden ver este listado son: ▪ Usuario de gerencia ▪ Usuario asesor de ventas ▪ Usuario de despachos ▪ Usuario de producción
Precondiciones: El usuario debe haber sido autenticado y autorizado para entrar al sistema. La aplicación debe haberse conectado a la base de datos del servidor para obtener la información y haberla filtrado por el campo *PUEDEDESPACHAR* que es el que indica si el pedido fue liberado.
Flujo normal: ▪ Usuario ingresa al sistema. ▪ Selecciona el módulo de pedidos. ▪ Selecciona la opción de pedidos pendientes de liberar. ▪ El sistema presenta los pedidos pendientes de liberar ▪ El usuario realiza las búsquedas y ordenamientos que desee.
Flujo alternativo: No existe.
Poscondiciones: El usuario puede ver el listado de pedidos pendientes de liberar ordenándolos por fecha, cliente, o vendedor y pudiendo buscar uno específico.

Cuadro 14: Documento de caso de uso ver pedidos pendientes de despachar: Cambio de estado y programación despacho

Nombre:	Ver pedidos pendientes de despachar: Cambio de estado y programación despacho
Autor:	Rocío García Morales
Fecha:	24-09-2005
Descripción: Presentará los pedidos que han sido ingresados y liberados en el sistema, pero que aún no han sido cien por ciento despachados. La información general que se presenta por cada pedido y la tabla de la que se obtiene es: ▪ Referencia de pedido : tabla *PEDIDO* ▪ Referencia de orden de producción: tabla *OP* ▪ Cliente: tabla *PEDIDO, PROYECTO Y CLIENTES* ▪ Proyecto: tabla *PEDIDO Y PROYECTO*	

- Vendedor: tabla *PEDIDO Y EMPLMAS*
- Fecha de entrega solicitada por el cliente: tabla *OP* campo *FECHAENTREGA*
- Fecha en planta, que indica la fecha en que se liberó y se autorizó su producción en planta: campo que debe incluirse en una tabla de la base de datos paralela.
- Fecha de inicio de producción: base de datos paralela.
- Fecha de fin de producción: base de datos paralela.
- Fecha en patio, que indica la fecha en la que el pedido se encuentra en patio listo para ser despachado: base de datos paralela.
- Referencia de orden de despacho: tabla *PEDIDO Y CAOD0DES.*
- Fecha en la que se emitió la orden de despacho: tabla *CAOD0DETDES.*
- Fecha de programación de despacho: base de datos paralela.

Además se presenta el detalle de cada pedido, las columnas que incluye el detalle son:

- Código de producto: tabla *OPDET* y *PRODUCTO.*
- Nombre de producto: tabla *PRODUCTO.*
- Cantidad pedida: tabla *OPDET.*
- Tamaño: tabla *OPDET.*
- Cantidad despachada: tabla *OPDET.*

La aplicación combina información de las dos bases de datos.

Los estados por los que un pedido liberado puede pasar son:

- En planta esperando inicio de producción.
- En producción.
- Producido esperando ser trasladado a patio.
- En patio esperando ser despachado.

Existe una fecha para cada uno de estos estados, si la fecha se encuentra nula el pedido aún no ha alcanzado dicho estado. Estas cuatro fechas únicamente pueden ser cambiadas por un usuario de producción y deben tener un orden lógico.

Una vez el pedido tiene asignada una fecha en patio, un usuario de despachos puede ingresar la fecha de programación de despacho del pedido en este listado.

Actores:

Los usuarios que pueden ver este listado son:

- Usuario de gerencia.
- Usuario asesor de ventas.
- Usuario de despachos.
- Usuario de producción.

Precondiciones:

El usuario debe haber sido autenticado y autorizado para entrar al sistema. La aplicación debe haberse conectado a la base de datos del servidor para obtener la información y filtrarla por el campo *PUEDE-DESPACHAR* que indica si el pedido fue liberado y por la diferencia entre el campo *APRODUCIR* y *DESPACHADOQTY* de cada producto que pertenece a una orden de producción para determinar si queda pendiente de despachar algún producto.

Flujo normal:

- Usuario ingresa al sistema.
- Selecciona el módulo de pedidos.
- Selecciona la opción de pedidos pendientes de despacho.
- El sistema presenta los pedidos pendientes de despacho
- El usuario realiza las búsquedas, agrupamientos y ordenamientos que desee
- Si el usuario es de producción puede modificar las fechas:
  - Fecha en planta.
  - Fecha de inicio producción.
  - Fecha de fin de producción.
  - Fecha en patio.
- Si el usuario es de despachos entonces puede modificar la fecha de programación de despacho que debe ser mayor o igual a la fecha en patio.

Flujo alternativo:

No existe.

Poscondiciones:
El usuario puede ver el listado de pedidos pendientes de despacho pudiendo ordenarlos o agruparlos por fecha, cliente, o vendedor y pudiendo buscar uno específico. Pueden existir cambios en las fechas de estado del pedido en producción y puede que se programe el despacho.

Cuadro 15: Documento de caso de uso ver pedidos despachados

Nombre:	Ver pedidos despachados
Autor:	Rocío García Morales
Fecha:	24-09-2005

Descripción:

Presentará los pedidos que han sido despachados en un rango de fechas ingresado por el usuario. La información general y el detalle que se presenta por cada pedido es la misma que se presenta para los pedidos pendientes de despacho. Lo que se despliega nuevo es la fecha de despacho y el número de envío y factura relacionado.

Actores:

Los usuarios que pueden ver este listado son:

- Usuario de gerencia.
- Usuario asesor de ventas.
- Usuario de despachos.
- Usuario de producción.

Precondiciones:

El usuario debe haber sido autenticado y autorizado para entrar al sistema. La aplicación debe haberse conectado a la base de datos del servidor para obtener la información y filtrarla para devolver únicamente

los pedidos despachados.
Flujo normal: ▪ Usuario ingresa al sistema. ▪ Selecciona el módulo de pedidos. ▪ Selecciona la opción de pedidos pendientes de despacho. ▪ Ingresa el rango de fechas (fecha de inicio y fecha de fin) ▪ El sistema presenta los pedidos despachados ▪ El usuario realiza las búsquedas, agrupamientos y ordenamientos que desee
Flujo alternativo: No existe.
Poscondiciones: El usuario puede ve el listado de pedidos despachados pudiendo ordenarlos o agruparlos por fecha, cliente, o vendedor y pudiendo buscar uno específico.

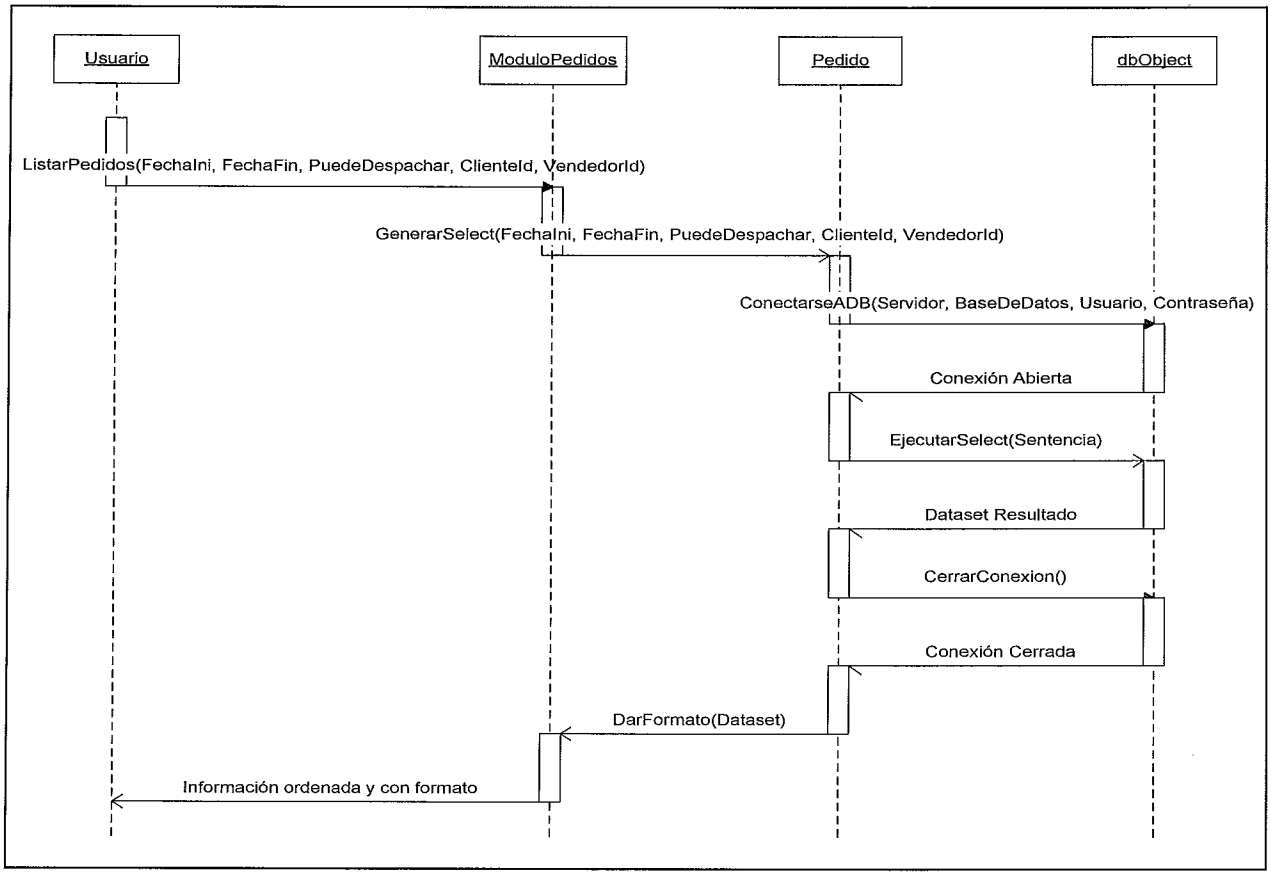
Cuadro 16: Documento de caso de uso ver resumen de despachos por rango de tiempo

Nombre:	Ver resumen de despachos por rango de tiempo
Autor:	Rocío García Morales
Fecha:	24-09-2005
Descripción: Según un rango de fechas ingresado por el usuario se generará un resumen con la siguiente información: ▪ Cantidad de despachos en ese rango de fechas. ▪ Cantidad nominal y porcentual de despachos entregados en fecha solicitada por el cliente: pudiendo ver los entregados a tiempo y los entregados tarde si se desea. ▪ Cantidad nominal y porcentual de despachos realizados en la fecha programada: pudiendo ver los realizados de acuerdo a la fecha programada y los realizados fuera de esa fecha. ▪ Tiempo promedio entre ingreso y despacho de pedido. ▪ Tiempo promedio en producción. Esta información se obtiene de las tablas de *MASMOV*, *PEDIDO* y *OPDET*.	
Actores: El único usuario que puede ver este resumen es: ▪ Usuario de gerencia.	
Precondiciones: El usuario debe haber sido autenticado y autorizado para entrar al sistema. La aplicación debe haberse conectado a la base de datos del servidor para obtener la información.	

Flujo normal: ▪ Usuario ingresa al sistema. ▪ Selecciona el módulo de pedidos. ▪ Selecciona la opción de resumen de despachos.
Flujo alternativo: No existe.

4. Diagramas de secuencia

Figura 20: Diagrama de secuencia para ver pedidos pendientes de liberar



El objeto *ModuloPedidos* forma parte de la capa de presentación, ofrece al usuario las opciones dentro del módulo de pedidos y presenta los resultados de la opción seleccionada. El objeto pedido de acuerdo a los parámetros que recibe para cada uno de sus métodos, elabora las sentencias de SQL para obtener la información sobre pedidos almacenada en las bases de datos, este objeto forma parte de la capa de aplica-

ción. El objeto dbObject ya fue descrito y es el que controla la conexión a la base de datos y permite ejecutar las sentencias que recibe de la capa de aplicación.

Entre las opciones que presenta el *ModuloPedidos* se encuentra la de ver pedidos pendientes de liberar. En este caso el usuario selecciona la opción y escoge el cliente y/o vendedor.

Se llama a la función *ListarPedidos* del objeto *ModuloPedidos* que en este caso recibe en los parámetros FechaIni y FechaFin valores nulos, en PuedeDespachar '0', pues el campo PuedeDespachar de la tabla Pedido es cero si el pedido no ha sido liberado, y en ClienteId y VendedorId puede recibir nulo o algún identificador y pueden darse las siguientes combinaciones:

- Ambos, ClienteId y VendedorId, son nulos. En este caso se presentará el listado de todos los pedidos pendientes de liberar.

- ClienteId es un identificador de cliente y VendedorId es nulo. En este caso se presentarán todos los pedidos pendientes de liberar de un cliente específico.

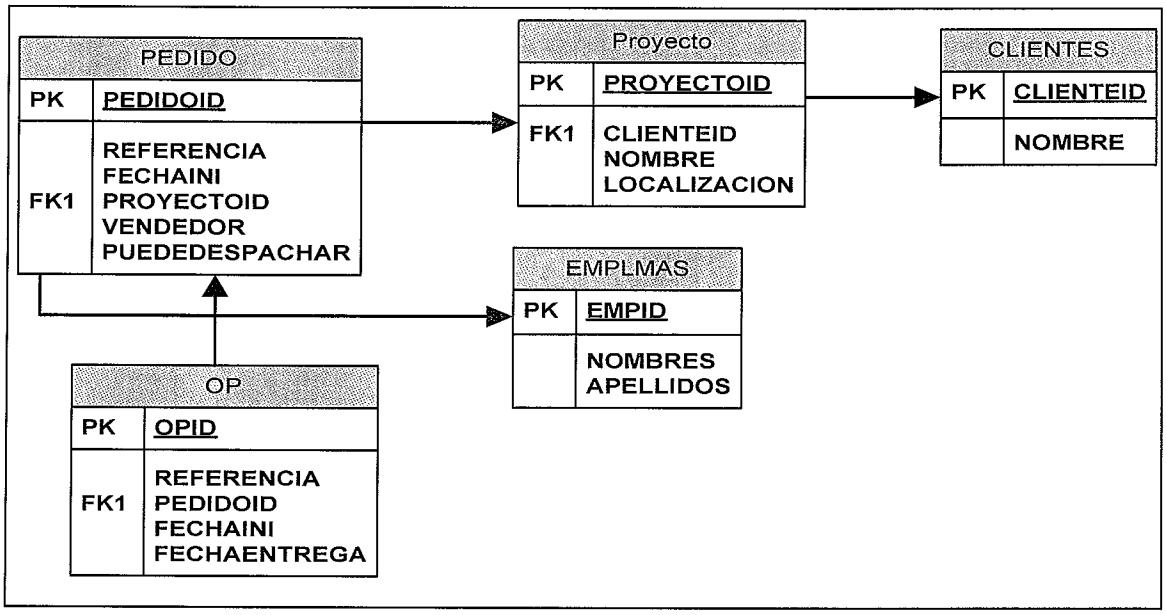
- ClienteId es nulo y VendedorId es un identificador de vendedor. En este caso se presentarán todos los pedidos pendientes de liberar de un vendedor específico.

- ClienteId es un identificador de cliente y VendedorId es un identificador de vendedor. En este caso se presentarán los pedidos pendientes de liberar de un cliente con un vendedor específico.

El método *ListarPedidos* envía los parámetros al método *GenerarSelect* del objeto *Pedido*, el cuál genera, con los parámetros que recibe, una sentencia de tipo *select* para traer los pedidos pendientes de liberar de la base de datos. Las tablas y campos de los que se obtiene la información se presentan en la figura 21.

Luego que la sentencia es preparada, el objeto *Pedido* solicita al objeto *dbObject* conectarse a la base de datos del sistema de Precon, si se logra establecer la conexión, se solicita que se ejecute la sentencia producida llamando al método *EjecutaSelect* que recibe como parámetro una cadena de consulta y devuelve a *Pedido* el resultado de la consulta en un dataset, se cierra la conexión y *Pedido* envía a *ModuloPedidos* el dataset para que presente la información en el formato correcto.

Figura 21: Tablas y campos para obtener información sobre pedidos pendientes de liberar

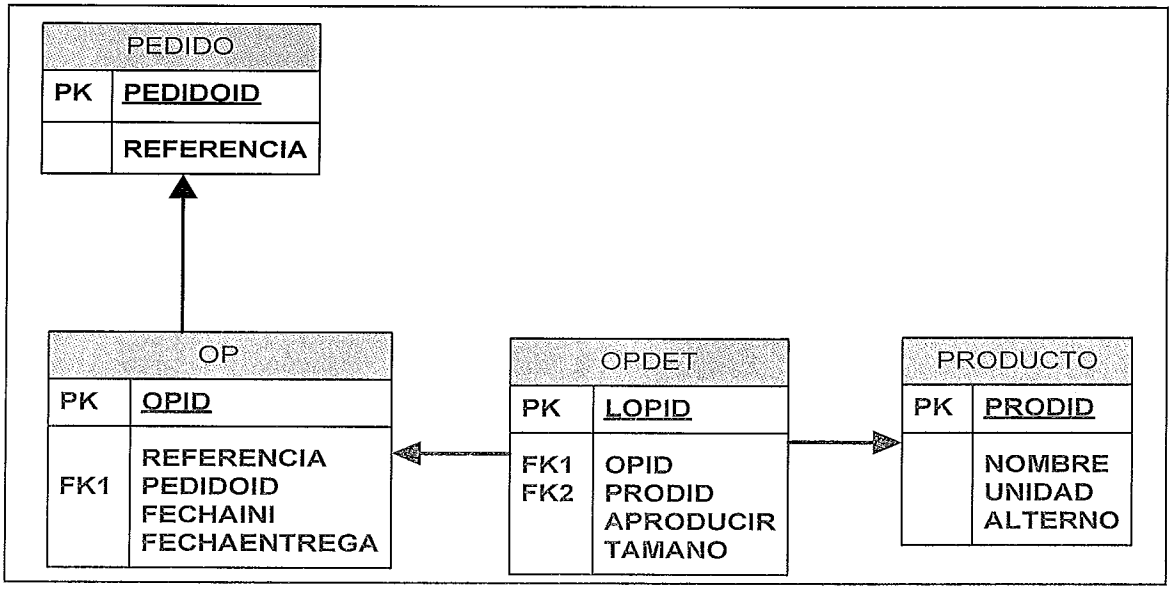


El encabezado del listado de pedidos pendientes de liberar será el siguiente:

Referencia pedido	Referencia OP	Fecha emi-sión	Cliente	Proyecto	Vendedor	Fecha entrega

En el listado presentado al usuario se ofrece la opción de ver el detalle de cada pedido, este detalle se obtiene de las siguientes tablas y campos de la base de datos:

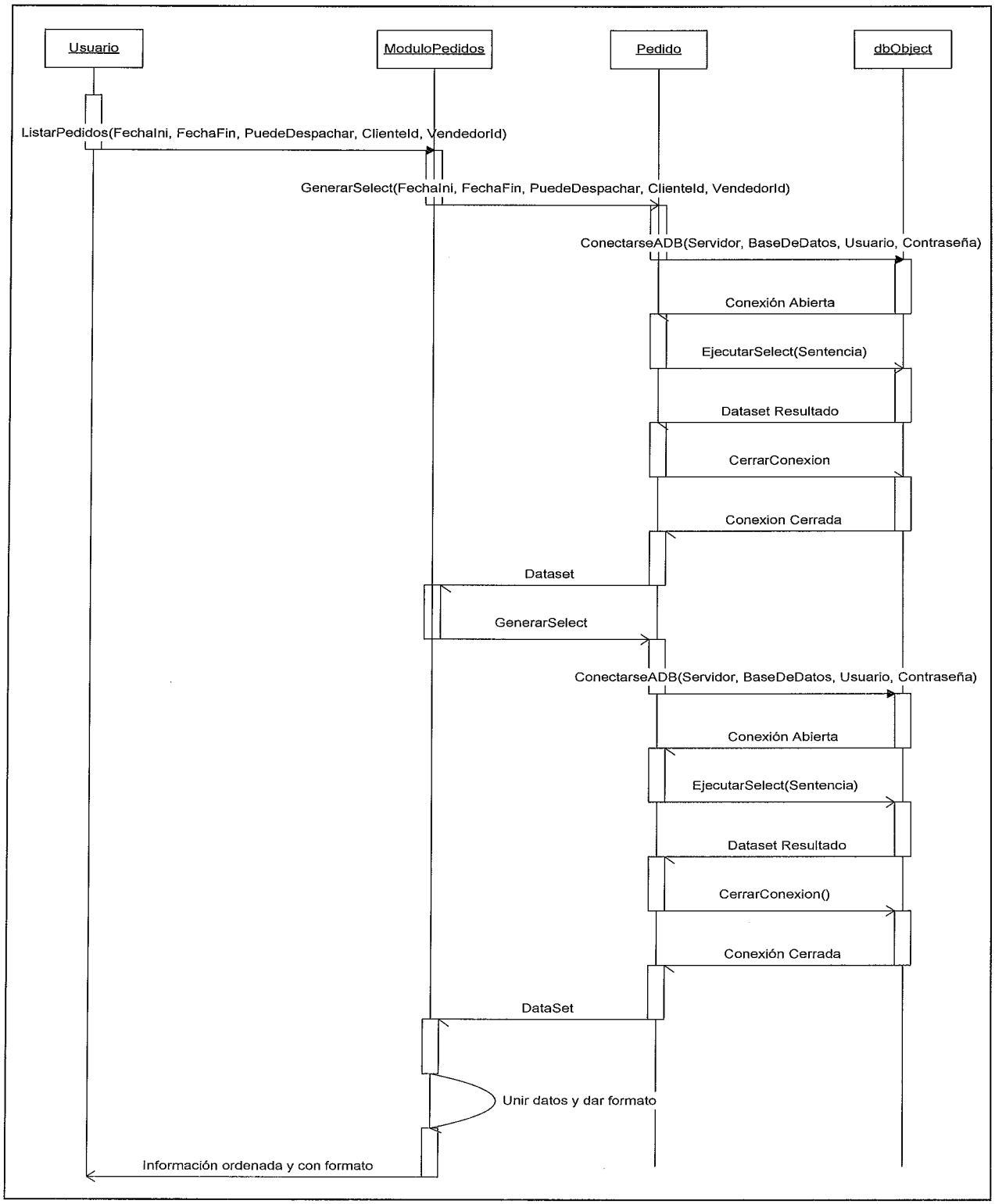
Figura 22: Tablas y campos de la base de datos para obtener el detalle de un pedido



El encabezado para el detalle del pedido es el siguiente:

Código producto	Nombre producto	Cantidad pedida	Tamaño

Figura 23: Diagrama de secuencia para ver pedidos pendientes de despachar y despachados

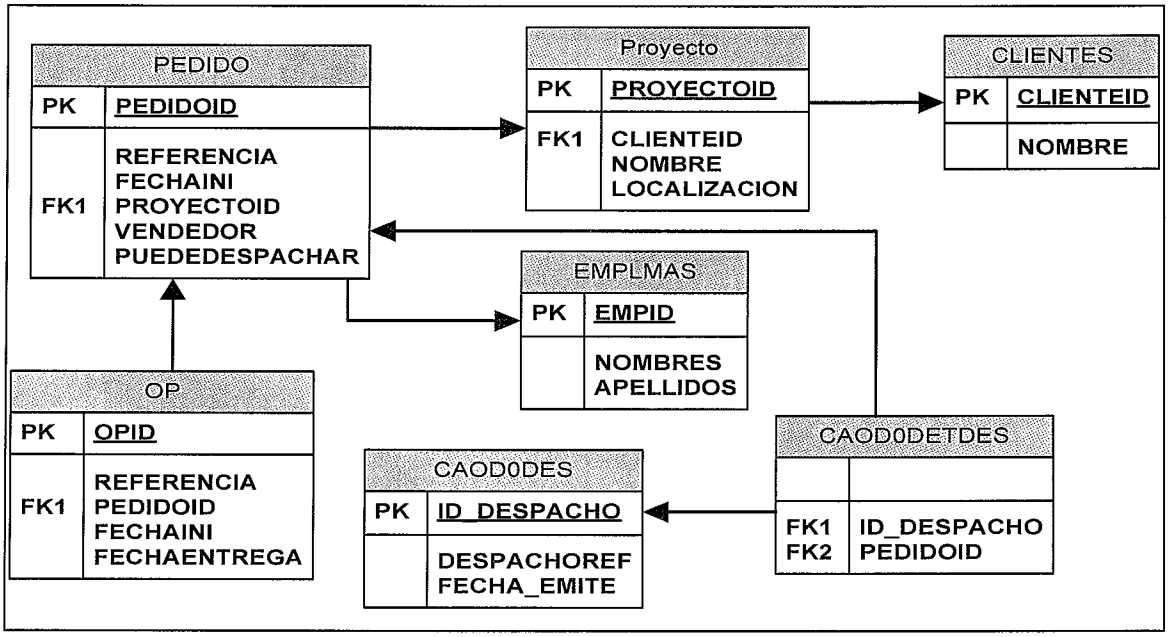


En este diagrama interaccionan los mismos objetos que en el diagrama anterior (ver pedidos pendientes de liberar), lo que cambia en este caso es que parte de la información se obtiene de la base datos actual y

el resto de la base de datos paralela, es por ello que el objeto *Pedido* llama al método *ConectarseADB* de *dbObject* dos veces, la información es devuelta en dos dataset diferentes y unida por el objeto *ModuloPedidos* para presentarla al usuario.

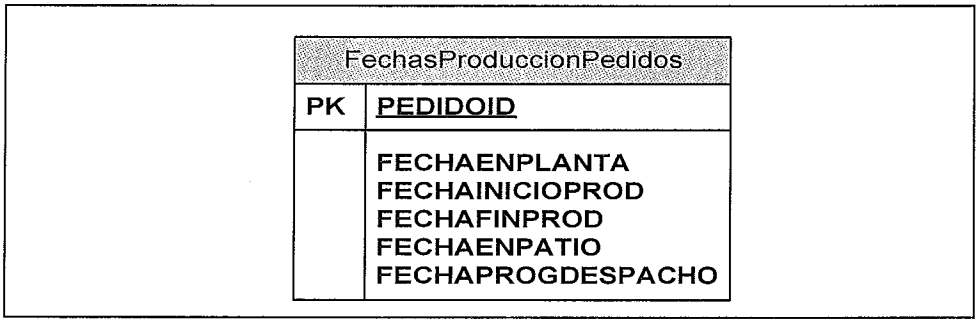
La información general presentada por cada pedido se obtiene de las tablas y campos de la base de datos del sistema actual, que se presentan en la figura que sigue.

Figura 24: Tablas y campos de la base de datos para pedidos pendientes de despachar



Adicionalmente se guardará en la base de datos paralela una tabla con la siguiente información:

Figura 25: Fechas de un pedido en planta pendiente de despachar



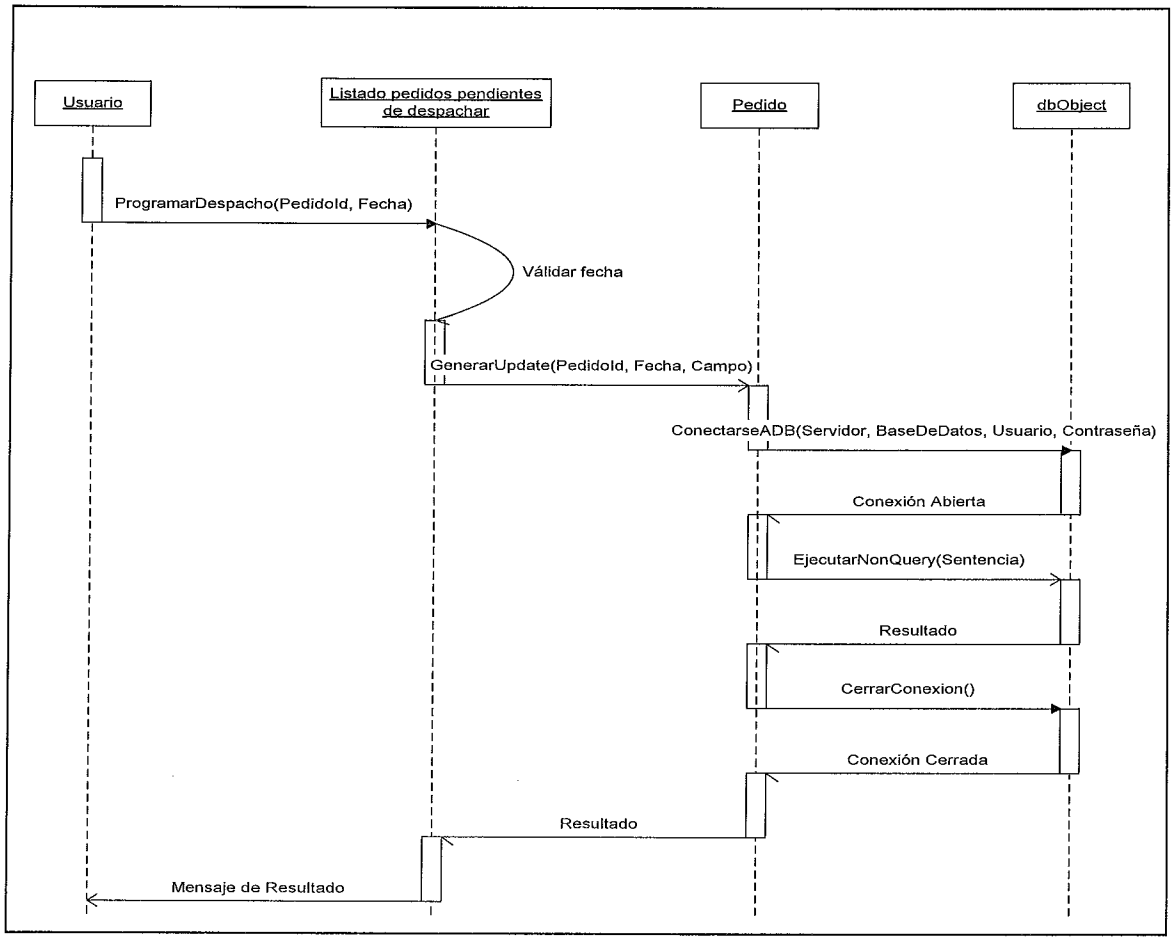
El formato del encabezado del listado de pedidos pendientes de despachar se presenta a continuación:

Referencia pedido	Referencia OP	Cliente	Vendedor	Fecha entrega	Fecha en planta	Fecha inicio producción	Fecha en patio	Orden despacho	Despacho prog.
Fecha emisión						Fecha fin producción		Fecha emisión	

El detalle para cada pedido se obtiene de las mismas tablas que el detalle para los pedidos pendientes de liberar, el formato del encabezado para el detalle es el siguiente:

Código producto	Nombre producto	Tamaño	Cantidad pedida	Cantidad despachada	Saldo

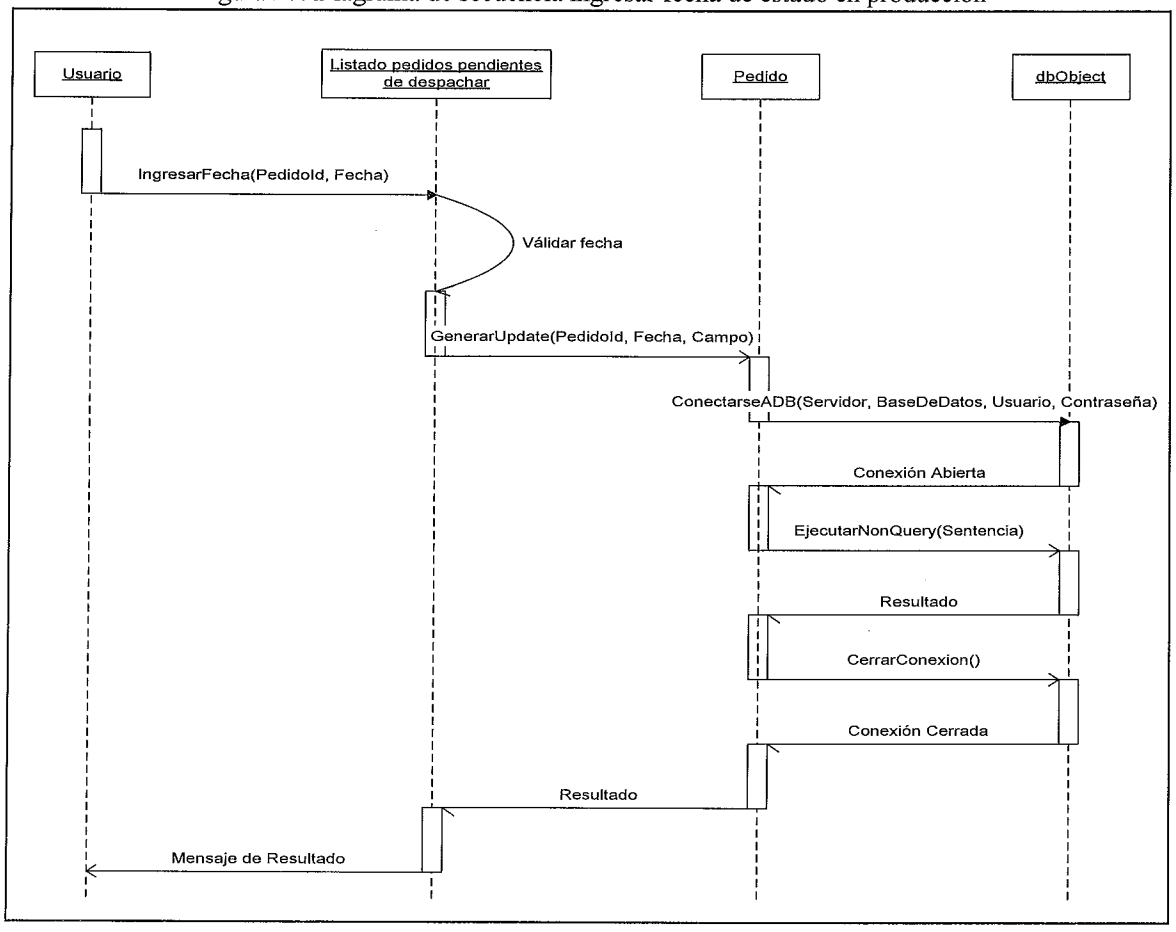
Figura 26: Diagrama de secuencia para programar despacho



En el listado de pedidos el usuario de despachos tiene la opción de ingresar la fecha de despacho de un pedido seleccionado. La fila del pedido está asociada al identificador *PEDIDOID* y el usuario únicamente ingresa la fecha en la que se programa el despacho.

Una vez la fecha es ingresada, la capa de presentación llama al método *GenerarUpdate* del objeto *Pedido* que en base al identificador que recibe como parámetro (PedidoId) genera una sentencia tipo *update* para ese pedido igualando el campo FechaProgDespacho a la fecha que recibe como parámetro. El objeto *Pedido* solicita a *dbObject* que inicie la conexión a la base de datos paralela y luego llama al método EjecutarNonQuery al que le envía como parámetro la cadena con la sentencia generada, *dbObject* le indica el número de filas afectadas por la sentencia y *Pedido* solicita cerrar la conexión. Por último, *Pedido* indica a la capa de presentación el resultado para que ésta despliegue un mensaje al usuario.

Figura 27: Diagrama de secuencia ingresar fecha de estado en producción



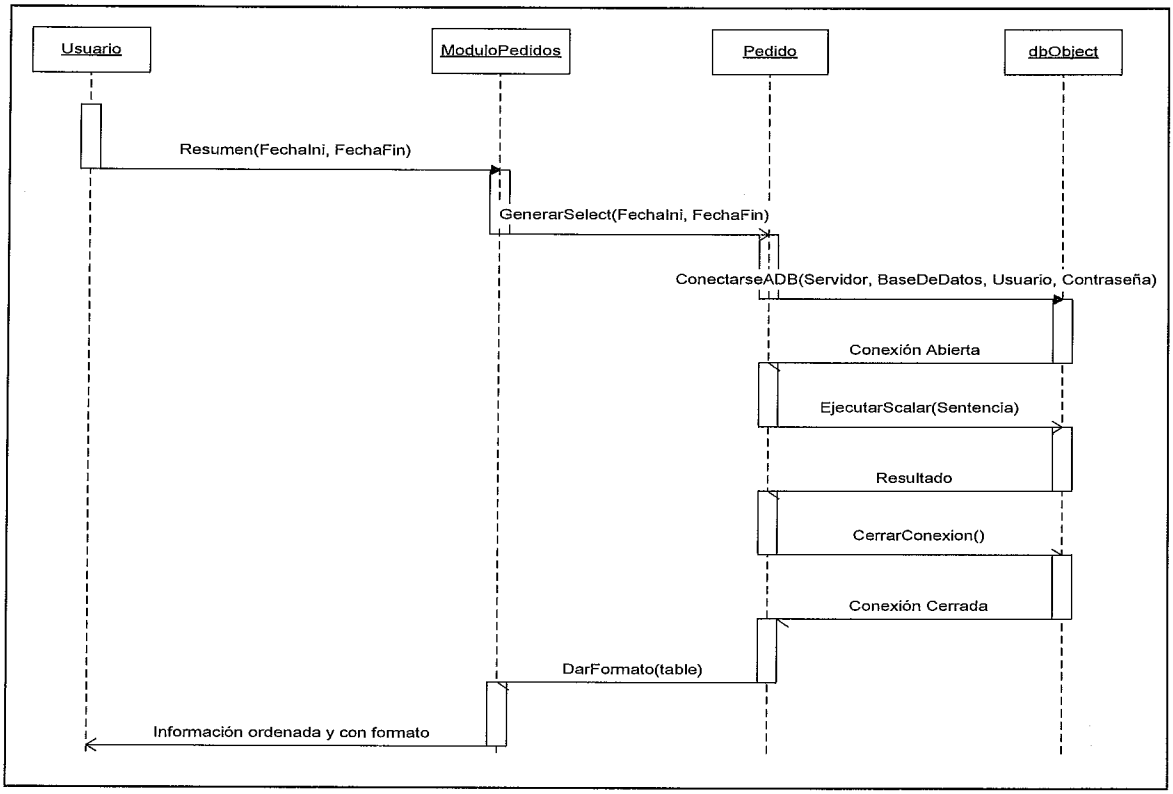
La secuencia del diagrama anterior es igual a la de programar despacho, excepto que para armar la sentencia *update* se envía a la función *GenerarUpdate* en el parámetro campo:

- FechaEnPlanta

- FechaInicioProd
- FechaFinProd
- FechaEnPatio

Según la fecha que el usuario haya solicitado ingresar, la sentencia generada iguala el campo recibido a la fecha que recibe como parámetro.

Figura 28: Diagrama de secuencia para ver resumen de despachos



En este diagrama intervienen los mismos objetos que en el diagrama para ver pedidos pendientes de despachar. Al seleccionar la opción de ver resumen de despachos *ModuloPedidos* solicita al usuario ingresar la fecha de inicio y de fin para el rango de despachos. Las fechas son proporcionadas por el usuario al módulo y el módulo solicita al objeto *Pedido* que genere, las sentencias necesarias para calcular los siguientes valores:

- Cantidad de despachos en ese rango de fechas.
- Cantidad nominal y porcentual de despachos entregados en fecha solicitada por el cliente: pudiendo ver los entregados a tiempo y los entregados tarde si se desea.
- Cantidad nominal y porcentual de despachos realizados en la fecha programada: pudiendo ver los realizados de acuerdo a la fecha programada y los realizados fuera de esa fecha.
- Tiempo promedio entre ingreso y despacho de pedido.

- Tiempo promedio en planta.
- Tiempo promedio en producción.

Esta información se obtiene de las tablas de *MASMOV*, *PEDIDO* y *OPDET*. Un tipo de movimiento almacenado en la tabla *MASMOV* son los despachos a los clientes, en caso de que sea un despacho, el registro de esta tabla se encuentra asociado a un pedido.

Se determinará si el pedido se despachó en la fecha solicitada por el cliente, comparando el campo *FECHAENTREGA* en la tabla de pedido y el campo *FECHA* del registro correspondiente a la primera entrega del pedido en la tabla *MASMOV*.

Se determinará si el pedido fue entregado en la fecha programada comparando el campo *FECHA-PROGRAMADA* de la base paralela y el campo *FECHA* del registro correspondiente a la primera entrega del pedido de la tabla *MASMOV*.

Para el tiempo promedio de despacho se encontrará la diferencia entre la fecha en que el pedido fue ingresado al sistema en la tabla *PEDIDO* y la fecha de la última entrega en la tabla *MASMOV*.

Para el tiempo promedio en planta se utilizarán los campos de la base de datos paralela que almacenan la fecha en que el pedido es liberado y la fecha en la que el pedido llega a patio.

Para el tiempo promedio en producción se utilizarán los campos de la base de datos paralela que almacenan la fecha en que el pedido inicia producción y la fecha en la que termina producción.

El método GeneraSelect del objeto *Pedido* genera una sentencia *Select* para cada uno de los valores anteriores, tomando en cuenta la fecha de inicio y de fin que recibe como parámetro, y solicita a dbObject que los ejecute a través del método EjecutarScalar al que le envía como parámetro cada sentencia. Los resultados son devueltos al módulo de pedidos en una tabla para que los presente en el formato correcto.

5. Especificación de clases. A continuación, presento las clases derivadas de los diagramas de secuencia presentados en el apartado anterior.

Figura 29: Clase pedidos

Pedidos
-dbObjects dbObject
+Pedidos() +GenerarSelect(in FechaIni: datetime, in FechaFin: datetime, in PuedeDespachar: int, in ClienteId: int, in VendedorId: int):DataSet +GenerarSelect(in FechaIni: in datetime,FechaFin: datetime): DataTable +GenerarUpdate(in PedidoId: int, in Fecha: datetime, in campo: string): int +DetallePedido(in PedidoId: int): DataTable

Esta clase pertenece a la capa de aplicación y posee como uno de sus miembros un objeto de tipo *dbObject* (figura 13) para ejecutar las sentencias que sus métodos producen sobre una base de datos.

- El constructor Pedidos() inicializa el objeto dbObject.

- El método GenerarSelect se encuentra sobrecargado y en el primer caso se utiliza para realizar una consulta en la que se obtienen los pedidos pendientes de liberar, pendientes de despachar o despachados, según los parámetros que el método recibe. Este método produce una sentencia SQL tipo *Select* con los parámetros recibidos y almacena los resultados en un dataset que devuelve al procedimiento que lo llamó. En el segundo caso el método se utiliza para ejecutar una sentencia en la que se traen los datos para el resumen de despachos y se regresa como resultado una tabla.

- El método GenerarUpdate genera las sentencias de tipo *update* y las ejecuta. Es utilizado para ingresar las fechas de estado en producción y la fecha de programación de despacho.

- El método detalle pedido recibe el identificador de un pedido y devuelve en una tabla el detalle de productos del pedido.

Figura 30: Clase ModuloPedidos

ModuloPedidos
-Pedidos Pedido
+ModuloPedidos() +ListarPedidos(in FechaIni: datetime, in FechaFin: datetime, in PuedeDespachar: int, in ClienteId: int, in VendedorId: int) +DetallePedido(in PedidoId: int) +PresentarResumen(in FechaIni: datetime, in FechaFin: datetime)

Esta clase pertenece a la capa de presentación y sus métodos dan formato y ordenan la información que reciben de las capas de aplicación. Posee como miembro a un objeto de tipo *Pedidos* que le provee la información. Los nombres de los métodos indican la función que éstos realizan.

F. Control de cotizaciones

1. Propósito. Presentar a los usuarios la información almacenada en el sistema de operaciones sobre cotizaciones aceptadas y rechazadas y permitir imprimir una cotización en el formato correcto o cambiarle el estado indicando que fue aceptada. Presentar al gerente resúmenes de cotizaciones por vendedor y generales, indicando la relación entre cotizaciones aceptadas y cotizaciones rechazadas.

2. Diagramas de casos de uso

Figura 31: Diagrama de casos de uso para usuario asesor en la aplicación control de cotizaciones

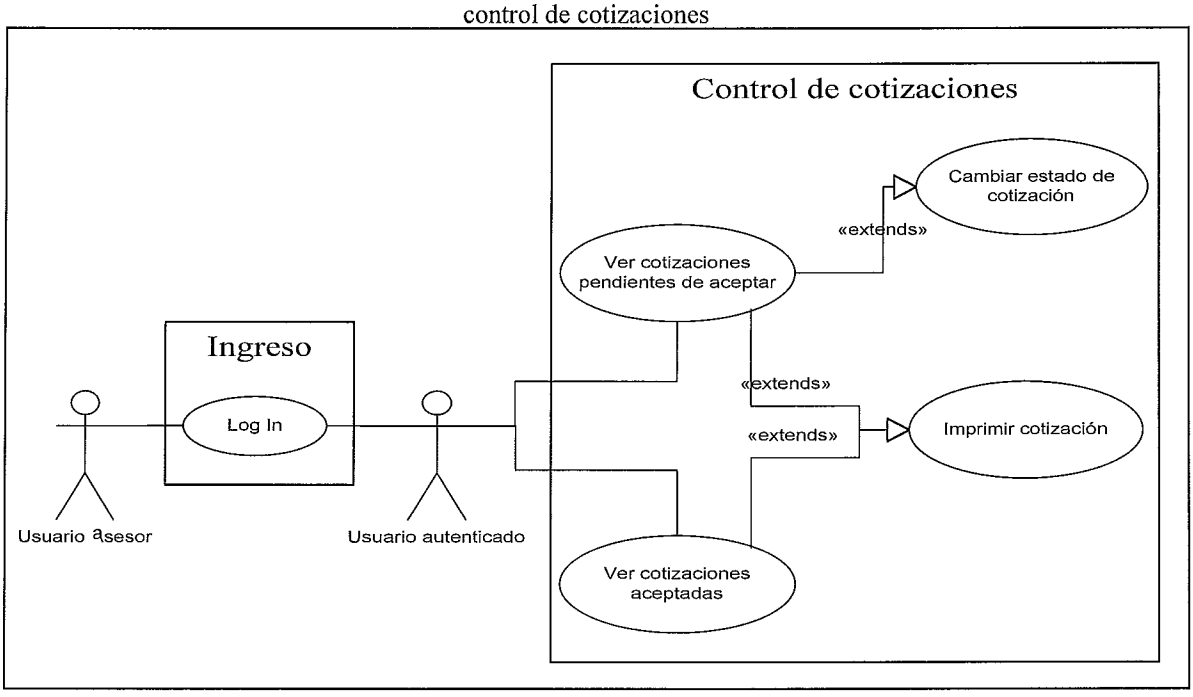
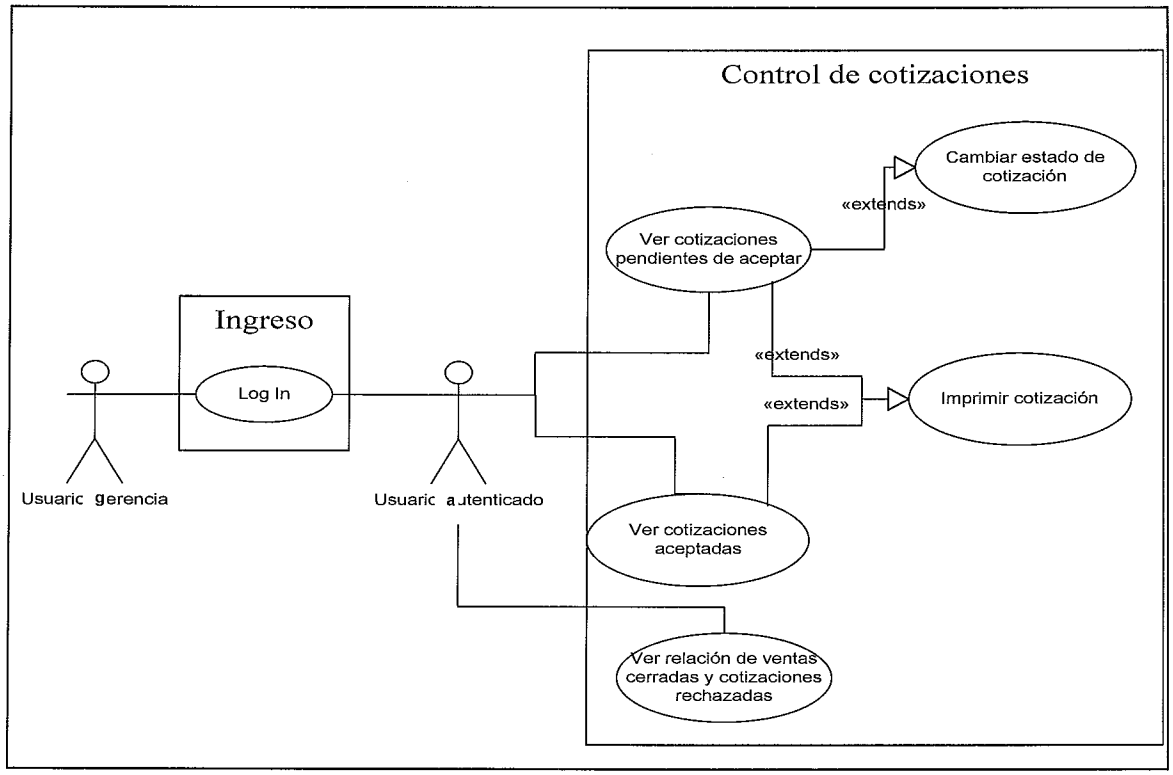


Figura 32: Diagrama de casos de uso para usuario de gerencia en la aplicación control de cotizaciones



3. Documentos de casos de uso

Cuadro 17: Documento de caso de uso ver cotizaciones pendientes de aceptar

Nombre:	Ver cotizaciones pendientes de aceptar
Autor:	Rocío García Morales
Fecha:	25-09-2005

Descripción:

Presenta las cotizaciones que han sido ingresados al sistema, pero que aún no han sido aceptadas. La información general que se presenta por cada cotización y la tabla de la que se obtiene es:

- Número de cotización : tabla *COTIZA*
- Fecha en la que ingreso al sistema: tabla *COTIZA*
- Nombre del proyecto: tabla *COTIZA*
- Localización: tabla *PROYECTO*
- Cliente: tabla *COTIZA Y CLIENTES*
- Vendedor: tabla *COTIZA Y EMPLMAS*
- Descuento: tabla *COTIZA*
- Fecha de entrega: tabla *COTIZA*

Además se presenta el detalle de cada cotización, las columnas que incluye el detalle son:

▪ Código de producto: tabla *COTIZADET* y *PRODUCTO* ▪ Nombre de producto: tabla *PRODUCTO* ▪ Cantidad pedida: tabla *COTIZADET* ▪ Tamaño: tabla *COTIZADET* ▪ Precio ofrecido: tabla *COTIZADET* Toda la información es obtenida de la base de datos actual.
Actores: Los usuarios que pueden ver este listado son: ▪ Usuario de gerencia ▪ Usuario asesor de ventas
Precondiciones: El usuario debe haber sido autenticado y autorizado para entrar al sistema. La aplicación debe haberse conectado a la base de datos del servidor para obtener la información y haberla filtrado por el campo *ESTADOID* que es el que indica si la cotización fue aceptada.
Flujo normal: ▪ Usuario ingresa al sistema. ▪ Selecciona el módulo de cotizaciones. ▪ Selecciona la opción de cotizaciones pendientes de aceptar. ▪ El sistema presenta las cotizaciones pendientes de aceptar. ▪ El usuario realiza las búsquedas u ordenamientos que desee y si desea puede imprimir o cambiar el estado de una cotización específica.
Flujo alternativo: No existe.

Cuadro 18: Documento de caso de uso ver cotizaciones aceptadas

Nombre:	Ver cotizaciones aceptadas
Autor:	Rocío García Morales
Fecha:	25-09-2005
Descripción: Presenta las cotizaciones que han sido ingresados al sistema y aceptadas. La información general que se presenta por cada cotización y la tabla de la que se obtiene es: ▪ Número de cotización : tabla *COTIZA.* ▪ Fecha en la que ingreso al sistema: tabla *COTIZA.* ▪ Nombre del proyecto: tabla *COTIZA.* ▪ Localización: tabla *PROYECTO.* ▪ Cliente: tabla *COTIZA Y CLIENTES.* ▪ Vendedor: tabla *COTIZA Y EMPLMAS.* ▪ Descuento: tabla *COTIZA.* ▪ Fecha de entrega: tabla *COTIZA.* ▪ Fecha de aceptación: tabla *COTIZA.*	

Además se presenta el detalle de cada cotización, las columnas que incluye el detalle son: ▪ Código de producto: tabla *COTIZADET* y *PRODUCTO.* ▪ Nombre de producto: tabla *PRODUCTO.* ▪ Cantidad pedida: tabla *COTIZADET.* ▪ Tamaño: tabla *COTIZADET.* ▪ Precio ofrecido: tabla *COTIZADET.* Toda la información es obtenida de la base de datos actual.
Actores: Los usuarios que pueden ver este listado son: ▪ Usuario de gerencia. ▪ Usuario asesor de ventas.
Precondiciones: El usuario debe haber sido autenticado y autorizado para entrar al sistema. La aplicación debe haberse conectado a la base de datos del servidor para obtener la información y haberla filtrado por el campo *ESTADOID* que es el que indica si la cotización fue aceptada.
Flujo normal: ▪ Usuario ingresa al sistema. ▪ Selecciona el módulo de cotizaciones. ▪ Selecciona la opción de cotizaciones aceptadas. ▪ El sistema presenta las cotizaciones aceptadas. ▪ El usuario realiza las búsquedas u ordenamientos que desee y si desea puede imprimir o cambiar el estado de una cotización específica.
Flujo alternativo: No existe.

Cuadro 19: Documento de caso de uso cambiar estado de cotización

Nombre:	Cambiar estado de cotización
Autor:	Rocío García Morales
Fecha:	25-09-2005
Descripción: En el listado de cotizaciones pendientes de aceptar permite cambiar el estado de una cotización para indicar que fue aceptada. Lo que se cambia es el estado de la cotización en la tabla *COTIZA*. Si el usuario es asesor debe ingresar un número de recibo provisional como garantía de que la cotización fue aceptada.	
Actores: Los usuarios que pueden cambiar el estado son: ▪ Usuario de gerencia. ▪ Usuario asesor de ventas.	

Precondiciones: El usuario debe haber sido autenticado y autorizado para entrar al sistema. La aplicación debe haberse conectado a la base de datos del servidor de Precon para obtener la información.
Flujo normal: ▪ Usuario ingresa al sistema. ▪ Selecciona el módulo de cotizaciones. ▪ Selecciona la opción cotizaciones pendientes de aceptar. ▪ El sistema presenta las cotizaciones pendientes de aceptar. ▪ El usuario cambia el estado de la cotización que desee. ▪ Si es usuario asesor ingresa el número de recibo provisional para que le sea permitido cambiar el estado.
Flujo alternativo: No existe.
Poscondiciones: Al menos una cotización cambió su estado y está garantizada con el número de recibo.

Cuadro 20: Documento de caso de uso imprimir cotización

Nombre:	Cambiar estado de cotización
Autor:	Rocío García Morales
Fecha:	25-09-2005
Descripción: En el listado de cotizaciones pendientes de aceptar o aceptadas permite imprimir la una cotización específica en el formato para presentar al cliente.	
Actores: Los usuarios que pueden imprimir cotizaciones son: ▪ Usuario de gerencia. ▪ Usuario asesor de ventas.	
Precondiciones: El usuario debe haber sido autenticado y autorizado para entrar al sistema. La aplicación debe haberse conectado a la base de datos de Precon del servidor para obtener la información.	
Flujo normal: ▪ Usuario ingresa al sistema. ▪ Selecciona el módulo de cotizaciones. ▪ Selecciona la opción cotizaciones pendientes de aceptar o cotizaciones aceptadas. ▪ El sistema presenta las cotizaciones. ▪ El usuario busca e imprime la cotización deseada.	

Flujo alternativo: No existe.
Poscondiciones: Al menos una cotización se imprimió.

Cuadro 21: Documento de caso de uso
Ver relación de ventas cerradas y cotizaciones rechazadas

Nombre:	Ver relación de ventas cerradas y cotizaciones rechazadas
Autor:	Rocío García Morales
Fecha:	25-09-2005
Descripción: Según un rango de fechas ingresado por el usuario se generará un resumen con la siguiente información por vendedor: ▪ Cantidad de cotizaciones emitidas en ese rango de tiempo. ▪ Porcentaje de cotizaciones aceptadas. ▪ Monto en quetzales de cotizaciones aceptadas. ▪ Monto en quetzales de cotizaciones rechazadas. También de forma general se presentará la cantidad de cotizaciones emitidas y el porcentaje de cotizaciones aceptadas. La información se obtiene de las tablas de *COTIZA* y *COTIZADET*. Si la cotización fue aceptada el campo *ESTADOID* es '1' y esta información se utiliza para calcular los porcentajes de cotizaciones aceptadas y rechazadas; el monto se calcula de la suma de cada producto de una cotización en la tabla *COTIZADET* más el descuento indicado en la tabla *COTIZA*.	
Actores: El único usuario que puede ver este resumen es: ▪ Usuario de gerencia.	
Precondiciones: El usuario debe haber sido autenticado y autorizado para entrar al sistema. La aplicación debe haberse conectado a la base de datos del servidor para obtener la información.	
Flujo normal: ▪ Usuario ingresa al sistema. ▪ Selecciona el módulo de cotizaciones. ▪ Selecciona la opción de resumen de cotizaciones .	
Flujo alternativo: No existe.	

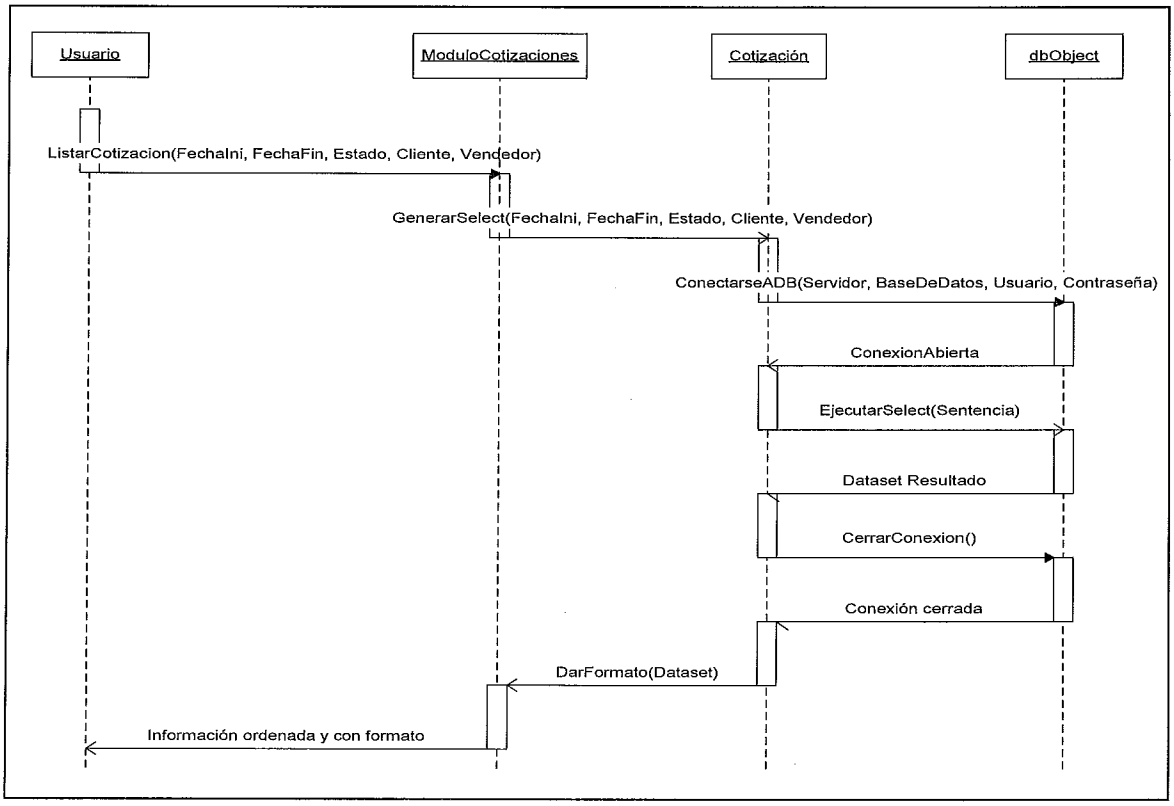
4. Diagramas de secuencia. Los objetos que intervienen en los diagramas de secuencia para el control de cotizaciones son:

- ModuloCotizaciones: contiene los métodos para presentar y recibir información del usuario. Ofrece las opciones del módulo y presenta los resultados de las solicitudes del usuario de forma ordenada y entendible. Se comunica con la capa de aplicación para que ésta le devuelva los resultados de las solicitudes.

- Cotización: uno de sus miembros es un objeto de tipo *dbObjects*, al cual solicita ejecutar las sentencias SQL que sus métodos generan para obtener información sobre cotizaciones en las bases de datos.

- dbObject: objeto para la comunicación a las bases de datos.

Figura 33: Diagrama de secuencia ver listado de cotizaciones pendientes de aceptar



Entre las opciones del módulo de cotizaciones se encuentra la de ver las cotizaciones pendientes de aceptar. Para ver este listado el usuario puede seleccionar un conjunto de filtros. Si el usuario es tipo asesor los filtros que selecciona son:

- Rango de fechas: para ver las cotizaciones pendientes de aceptar ingresadas en un rango de fecha.
- Cliente: para ver las cotizaciones pendientes de aceptar de un cliente.

Si el usuario no selecciona ningún filtro, podrá ver todas sus cotizaciones pendientes de aceptar.

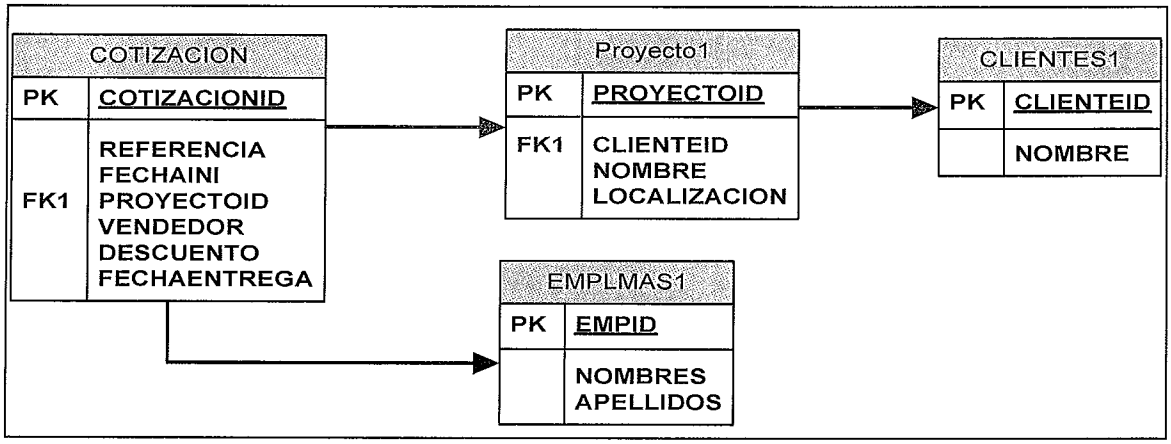
Adicionalmente el usuario de gerencia puede seleccionar el filtro vendedor, para ver las cotizaciones pendientes de aceptar de un vendedor específico. Si el usuario de gerencia no ingresa ningún filtro, podrá ver todas las cotizaciones pendientes de aceptar de todos los vendedores.

Los valores de los filtros son ingresados por el usuario a través de la capa de aplicación, representada en los diagramas por el objeto *ModuloCotizaciones* y método *ListarCotizaciones,* que además de los filtros recibe como parámetro el estado de las cotizaciones, que en este caso es '0' pues se desean ver las cotizaciones pendientes de aceptar.

El objeto de la capa de aplicación indica al objeto *Cotización* que genere la sentencia *Select,* con los filtros especificados en los parámetros, para obtener los datos de las cotizaciones de la base de datos utilizando el objeto dbObject.

Los datos son obtenidos de las tablas que se muestran en la figura 32.

Figura 34: Tablas y campos para obtener información general de cotizaciones



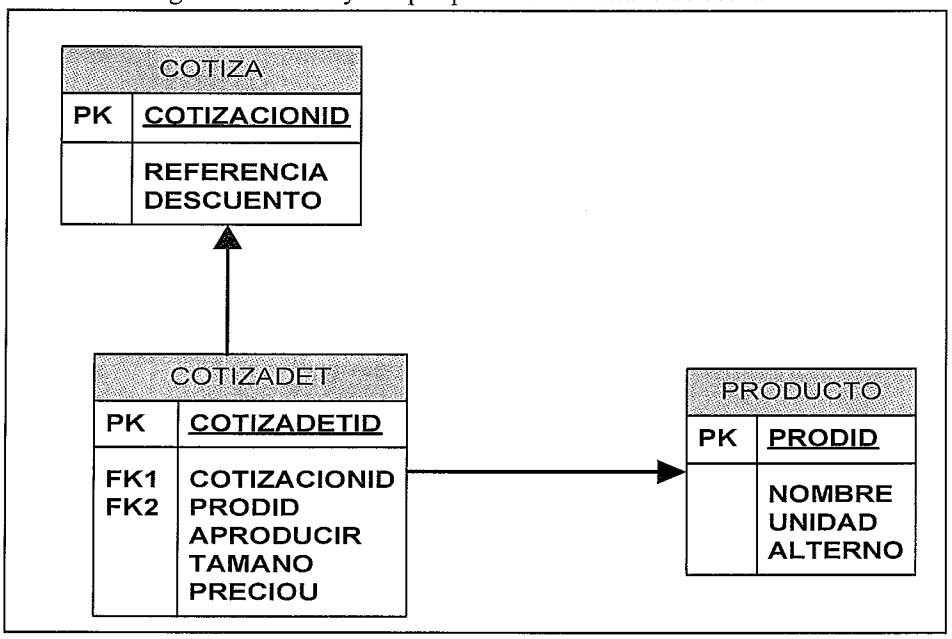
Los datos son devueltos de la capa de datos a la capa de aplicación y de ésta a la de presentación, a través de un dataset resultado de la ejecución de la sentencia.

El listado de las cotizaciones presenta el encabezado que se presenta a continuación:

Cotización	Cliente	Proyecto	Localización	Fecha entrega	Vendedor	Descuento	Fecha emisión

Cada cotización dentro del listado ofrecerá la opción de ver su detalle que se obtiene de las tablas que se muestran en la figura 33.

Figura 35: Tablas y campos para obtener detalle de cotizaciones



Y presenta el siguiente encabezado:

Código producto	Descripción	Tamaño	Cantidad	Precio unitario	Descuento	Total

El diagrama de la figura 36 presenta la secuencia para ver un listado de cotizaciones, el de cotizaciones aceptadas, por lo cual se debe leer también la base de datos paralela para obtener el número de recibo asociado a la cotización aceptada. Los objetos que interactúan realizan las mismas funciones que en el diagrama anterior a excepción de *Cotización* que ahora trabaja sobre dos bases de datos.

En el listado de cotizaciones aceptadas se presentará además de la información del listado de cotizaciones pendientes de aceptar, la fecha en la que se aceptó la cotización y el número de recibo asociado. El detalle por cotización sigue siendo el mismo.

Figura 36: Diagrama de secuencia ver listado de cotizaciones aceptadas

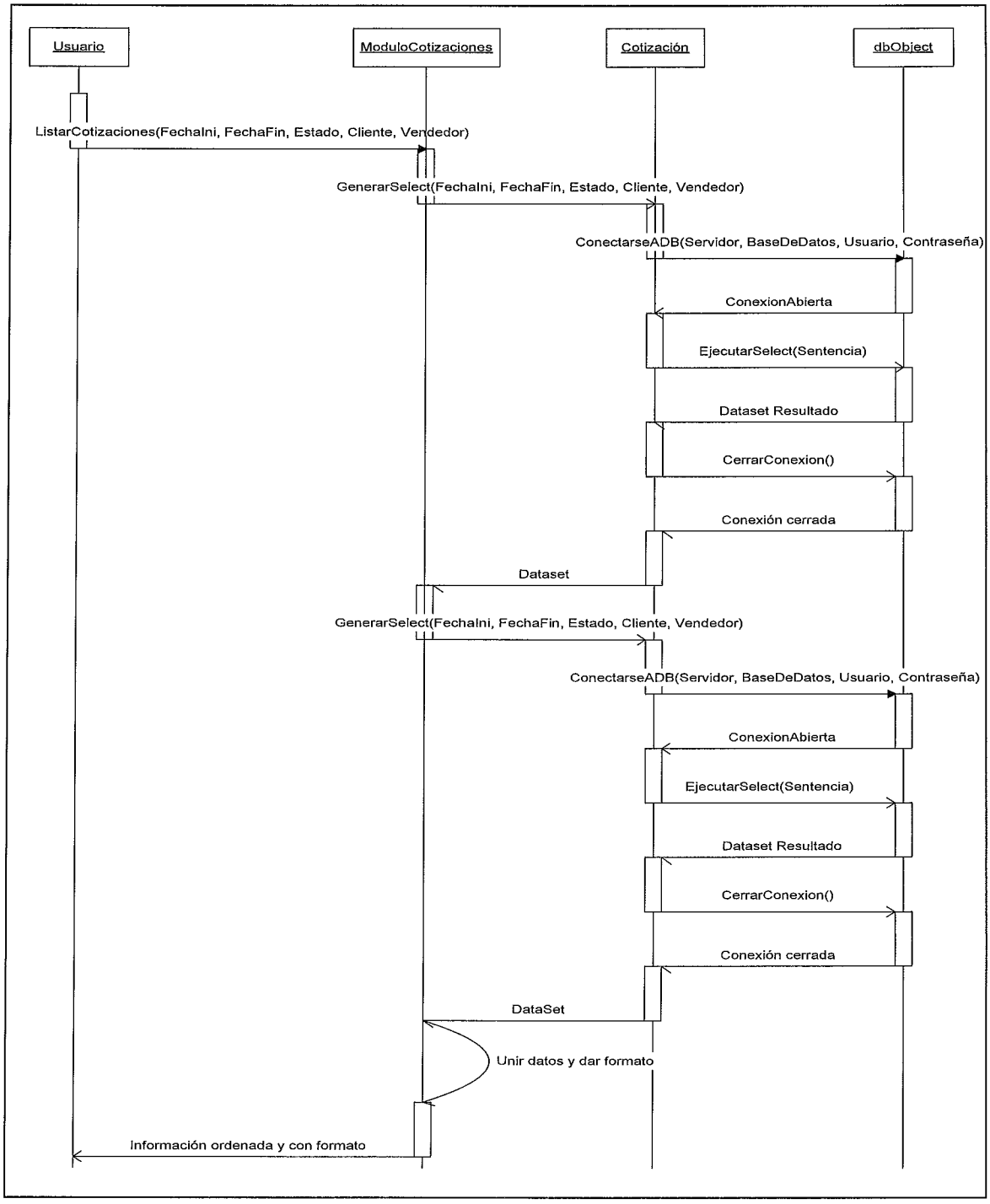
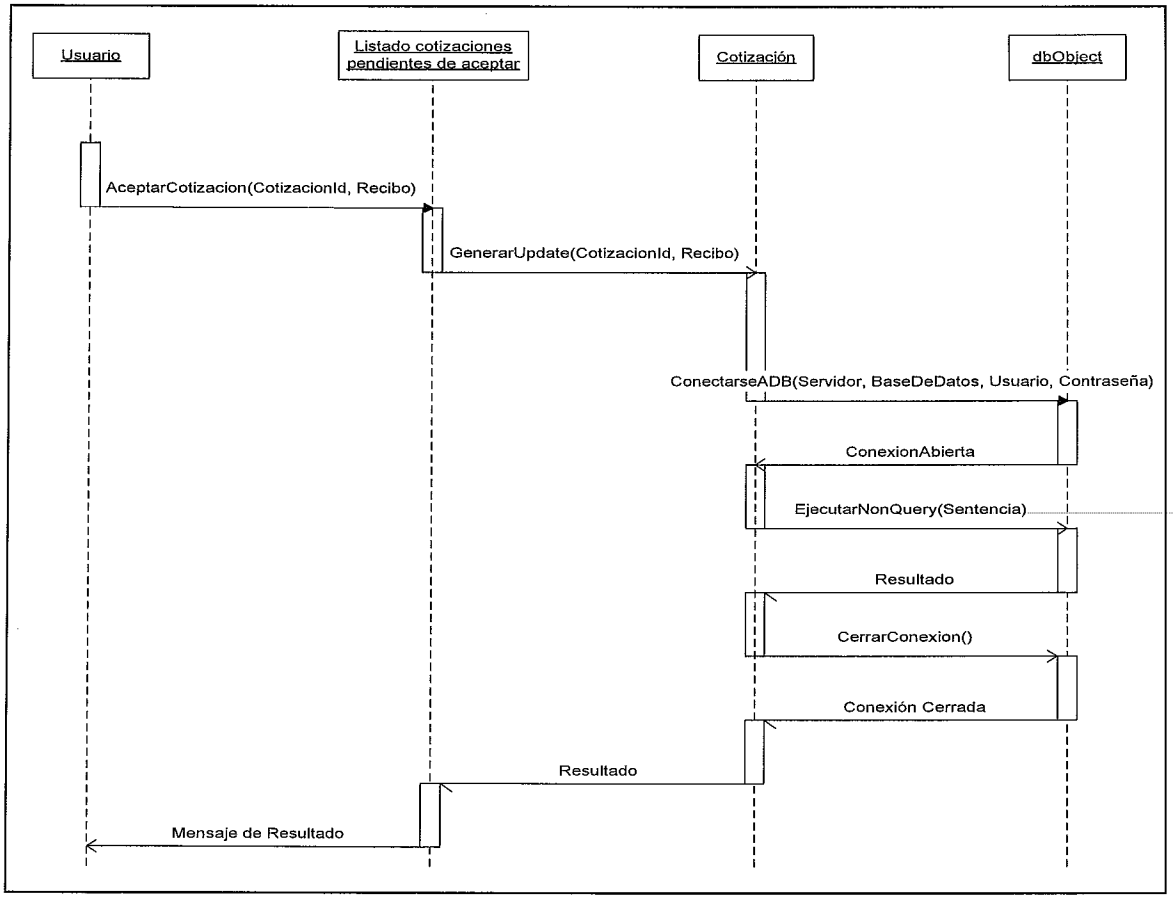
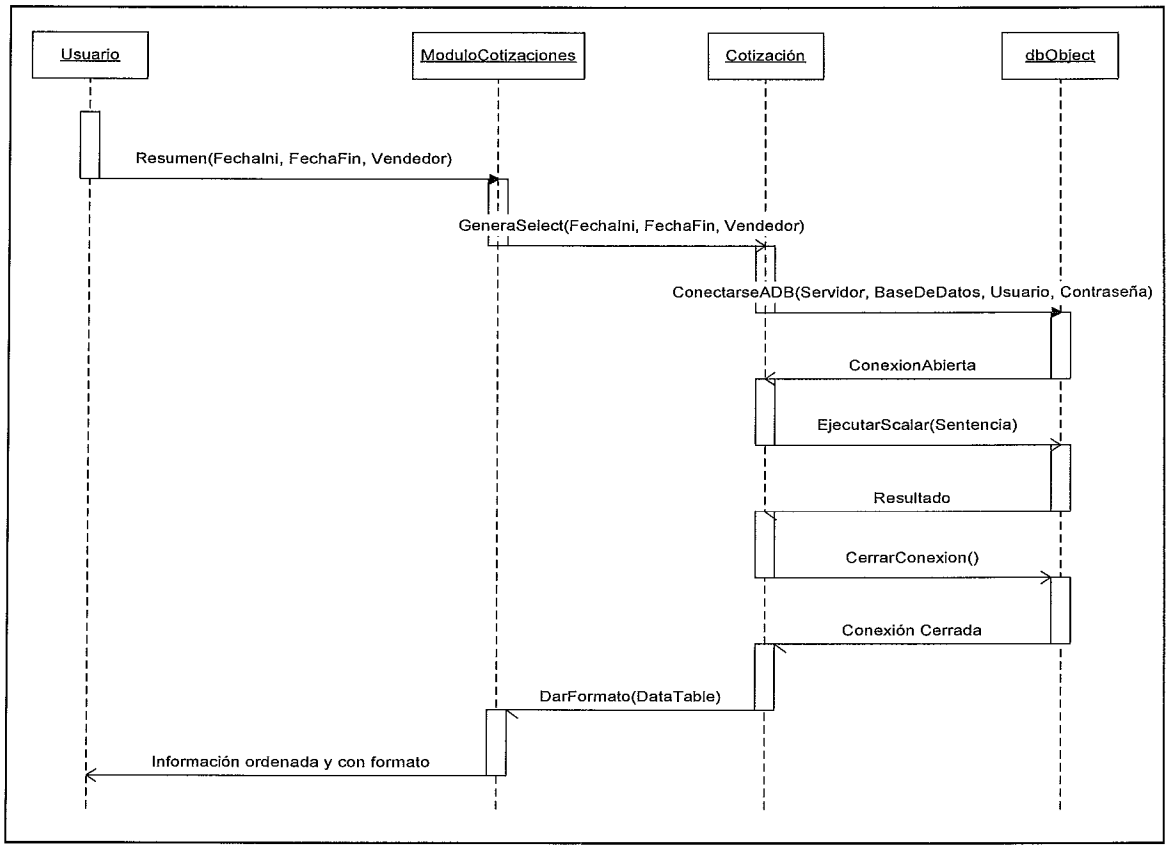


Figura 37: Diagrama de secuencia para cambiar estado de cotización



Dentro del listado de cotizaciones pendientes de aceptar, el usuario podrá seleccionar una cotización a la que desea cambiarle el estado. La fila de la cotización está asociada al identificador y el usuario únicamente debe marcar la cotización para aceptarla e ingresar el número de recibo provisional si es usuario tipo asesor. La capa de presentación solicita al objeto *Cotización* que genere la sentencia de actualización en la tabla *COTIZA* para cambiar el estado de la cotización que le envía como parámetro y para que almacene el número de recibo asociado. *Cotización* genera las sentencias SQL y solicita a *dbObject* que las ejecute a través del método EjecutaNonQuery al que le envía como parámetro cada sentencia. EjecutaNonQuery devuelve a *Cotización* el número de filas afectadas y el mismo resultado es devuelto a la capa de presentación para que presente un mensaje de resultado al usuario. La cotización deberá desaparecer del listado de cotizaciones pendientes de aceptar.

Figura 38: Diagrama de secuencia para ver relación ventas cerradas y cotizaciones rechazadas



Dentro del módulo de cotizaciones, un caso de uso es el de ver el resumen de ventas cerradas en relación a las cotizaciones rechazadas. El usuario deberá seleccionar la opción e ingresar el rango de fechas para el resumen y un vendedor para el cuál desea ver el resumen. Si no se selecciona ningún vendedor, podrá ver el resumen de todos los vendedores. Por vendedor se presentará la siguiente información:

- Número de cotizaciones emitidas en el rango de fechas.
- Número y porcentaje de cotizaciones aceptadas.
- Número y porcentaje de cotizaciones pendientes de aceptar o rechazadas.
- Monto de cotizaciones pendientes de aceptar o rechazadas.
- Monto de cotizaciones aceptadas.

La cual es obtenida a partir de conteos en la tabla *COTIZA* y sumatorias en la tabla *COTIZADET.*

El usuario ingresa el rango de fechas y vendedor a través de la capa de presentación y ésta llama al método GenerarSelect del objeto cotización, que genera los *Selects* necesarios para obtener cada dato presentado en el resumen, las sentencias son ejecutadas a través del método EjecutarScalar que devuelve un resultado escalar al ejecutar la sentencia que recibe como parámetro.

Los resultados son devueltos al módulo de cotizaciones en un datatable para que éste los presente al usuario.

5. Especificación de clases. Las clases utilizadas dentro del módulo de cotizaciones son dos y su especificación es la que sigue:

Figura 39: Clase cotizaciones

Cotizaciones
-dbObjects dbObject
+Cotizaciones() +GenerarSelect(in FechaIni: datetime, in FechaFin: datetime, in Estado: int, in Cliente: int, in Vendedor: int):DataSet +GenerarSelect(in FechaIni: in datetime,FechaFin: datetime, in Vendedor: int): DataTable +GenerarUpdate(in CotizacionId: int, in Recibo: int): int +DetalleCotización(in CotizacionId: int): DataTable +ImprimirCotizacion(in CotizacionId: int): int

Esta clase es muy similar a la clase *Pedidos*; posee como miembro un objeto tipo dbObject, al que solicita que ejecute sentencias SQL en la base de datos.

- El método GenerarSelect es sobrecargado y en el primer caso es utilizado por los métodos de la capa de presentación para generar los datos en los listados de cotizaciones. El método arma una sentencia *Select,* con los parámetros que recibe, para leer datos de las tablas *COTIZA, COTIZADET, CLIENTES, PROYECTO y EMPLMAS.* Solicita a *dbObject* que ejecute la sentencia a través del método EjecutarSelect el cual devuelve el dataset con los resultados de la ejecución, este dataset es devuelto por el método a la capa de presentación.

- En el segundo caso GenerarSelect arma y ejecuta las sentencias para obtener los datos presentados en el resumen de cotizaciones y devuelve los resultados en un datatable a la capa de presentación.

- GenerarUpdate es llamado para actualizar el estado de una cotización, y únicamente recibe el número de cotización y el número de recibo provisional. Arma una sentencia con esos parámetros para cambiar el estado en la base actual y otro para guardar el número de recibo asociado en la base paralela.

- Detalle cotización devuelve el detalle de la cotización que recibe como parámetro.

Figura 40:Clase ModuloCotizaciones

ModuloCotizaciones
-Cotizaciones Cotizacion
+ModuloCotizaciones() +ListarCotizaciones(in FechaIni: datetime, in FechaFin: datetime, in Estado: int, in ClienteId: int, in VendedorId: int) +DetalleCotizacion(in CotizacionId: int) +PresentarResumen(in FechaIni: datetime, in FechaFin: datetime, in Vendedor: int)

Esta clase pertenece a la capa de presentación y es la encargada de capturar y validar la información del usuario y presentar los resultados de las peticiones del usuario. Posee como miembro un objeto de la clase *Cotizaciones*, a la que entrega los datos del usuario para generar los resultados. Los métodos que la componen ya fueron descritos en los diagramas de secuencia.

G. Control de visitas

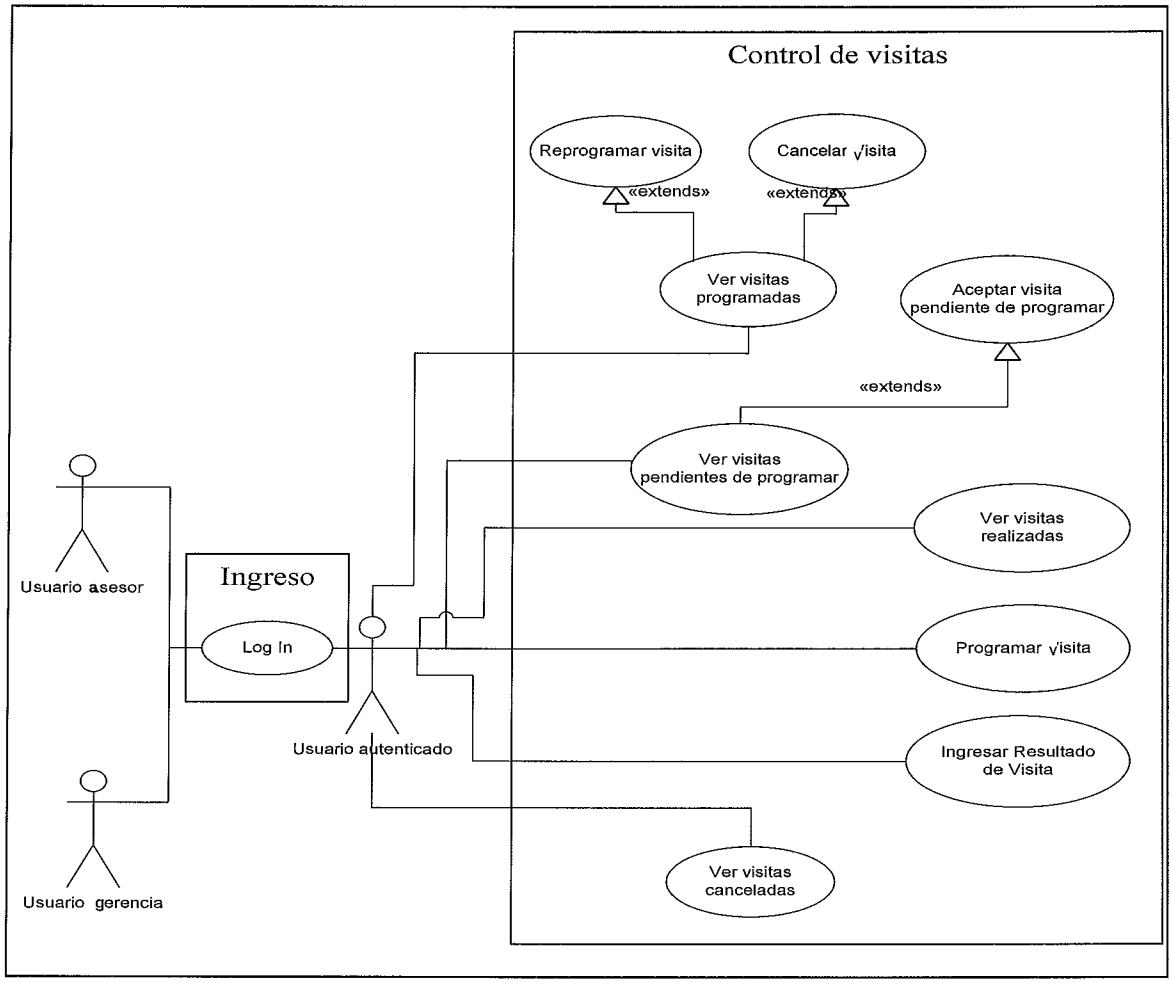
1. Propósito. Permitir a los asesores de ventas y al gerente, programar, reprogramar y cancelar las visitas con los clientes. Ingresando, luego de que la visita fue realizada, los resultados obtenidos para retroalimentación del gerente. Permitir al gerente ver las visitas programadas, realizadas y canceladas por vendedor y por cliente para analizar los resultados de un conjunto de visitas del mismo cliente.

Una visita puede tener los siguientes estados:

- Programada.
- Realizada.
- Cancelada.
- Pendiente de programar.

2. Diagramas de casos de uso

Figura 41: Diagrama de casos de uso en la aplicación *control de visitas*



3. Documentos de caso de uso

Cuadro 22: Documento de caso de uso programar visita

Nombre:	Programar visita
Autor:	Rocío García Morales
Fecha:	25-09-2005
Descripción: Permitirá al usuario programar una visita, la información que debe ingresar para una visita nueva es: ▪ Fecha en la que ser realizará. ▪ Hora.	

▪ Motivo que lo escogerá de un conjunto predefinido. ▪ Cliente. ▪ Lugar, se escoge de un conjunto predefinido o se ingresa un texto indicando el lugar. ▪ El nombre del contacto. ▪ Teléfono del contacto (opcional). ▪ Correo del contacto (opcional). Al crear la visita, el sistema automáticamente le asocia un número correlativo como identificador único, le asocia como vendedor al usuario conectado al sistema y le asigna como estado predeterminado *programada.*
Actores: ▪ Usuario de gerencia ▪ Usuario asesor de ventas
Precondiciones: El usuario debe haber sido autenticado y autorizado para entrar al sistema. No debe existir ninguna visita programada para el mismo día y la misma hora o alguna visita el mismo día que empiece menos de media hora antes que la programada. La fecha de programación debe ser mayor o igual a la fecha en la que se creo la visita.
Flujo normal: ▪ Usuario ingresa al sistema. ▪ Selecciona el módulo visitas. ▪ Selecciona la opción crear visita. ▪ Ingresa los datos de la visita. ▪ Si se ingresaron todos los campos requeridos, el sistema verifica que la fecha de la visita sea válida y que no esté programada otra visita que se cruce con la programada. ▪ Si la fecha es correcta, la información es almacenada en la base de datos paralela.
Flujo alternativo: Desde el listado de visitas, seleccionar la opción de crear una nueva visita.
Poscondiciones: Existirá una visita nueva para el usuario conectado al sistema.

Cuadro 23: Documento de caso de uso reprogramar visita – cancelar visita

Nombre:	Reprogramar visita – cancelar visita
Autor:	Rocío García Morales
Fecha:	25-09-2005
Descripción: Permite cambiar la fecha u hora de la visita o cancelar completamente la visita. Si se cambia la fecha u hora, el sistema siempre verifica que no exista algún cruce y que la nueva fecha sea válida.	
Actores:	

▪ Usuario de gerencia. ▪ Usuario asesor de ventas.
Precondiciones: El usuario debe haber sido autenticado y autorizado para entrar al sistema para ver únicamente sus visitas programadas. No puede reprogramar o cancelar la visita de otro usuario.
Flujo normal: ▪ Usuario ingresa al sistema. ▪ Selecciona el módulo visitas. ▪ Al ingresar al módulo de visitas podrá ver el listado de visitas programadas y se le permitirá buscar una específica o las visitas de un día. ▪ Al encontrar la visita deseada se permitirá cambiar la fecha, hora o lugar o cancelarla según desee el usuario ▪ Si se reprogramó la visita, el sistema verifica que la fecha de la visita sea válida y que no esté programada otra visita que se cruce con la programada. ▪ Si la fecha es correcta, la información es almacenada en la base de datos paralela. ▪ El sistema guarda en su la bitácora asociada a la visita, la fecha y la hora en la que fue reprogramada. ▪ Se indica al usuario que el cambio fue realizado con éxito.
Flujo alternativo: Si la fecha de la visita ya pasó y el usuario no ha ingresado los resultados de la misma, el sistema le preguntará si fue cancelada o realizada cambiando en el sistema el estado de la visita.
Poscondiciones: Al menos una visita será reprogramada o cancelada.

Cuadro 24: Documento de caso de uso ingresar resultados de visita

Nombre:	Ingresar resultados de visita
Autor:	Rocío García Morales
Fecha:	25-09-2005
Descripción: Permitirá al usuario ingresar los resultados de una visita. Los resultados se ingresan como un texto breve, la hora a la que inició la visita y la hora a la que terminó. Al ingresar los resultados, se le presenta la opción de indicar la fecha de la próxima visita, si el usuario ingresa esta fecha el sistema crea la visita en estado pendiente de programar para ingresar los detalles posteriormente.	
Actores: ▪ Usuario de gerencia. ▪ Usuario asesor de ventas.	
Precondiciones:	

El usuario debe haber sido autenticado y autorizado para entrar al sistema y ver únicamente sus visitas programadas. No puede ingresar resultados en ninguna visita de otro usuario.
Flujo normal: ▪ Usuario ingresa al sistema. ▪ Selecciona el módulo visitas. ▪ Cuando ingresa al módulo de visitas, podrá ver el listado de visitas programadas y se le permitirá buscar una específica o las visitas de un día. ▪ Al encontrar la visita deseada, se permitirá ingresar los resultados de la visita ▪ Si la fecha y hora de la visitan han pasado se guarda la información ▪ Se indica al usuario que el cambio fue realizado con éxito.
Flujo alternativo: Si la fecha y hora de la visita no han pasado aún, primero debe reprogramarse la visita y luego ingresar los resultados.
Poscondiciones: La visita cambia de estado *Programada a Realizada.* Si se ingresa la fecha de la próxima visita se habrá creado una visita en estado pendiente de programar con el mismo cliente.

Cuadro 25: Documento de caso de uso ver visitas pendientes de programar y aceptar visita pendiente de programar

Nombre:	Ver visitas pendientes de programar Aceptar visita pendiente de programar
Autor:	Rocío García Morales
Fecha:	25-09-2005
Descripción: Habiendo seleccionado una visita pendiente de programar, el usuario puede ingresar la hora de la visita, el lugar, motivo, y la información del contacto, convirtiendo la visita en una visita programada.	
Actores: ▪ Usuario de gerencia. ▪ Usuario asesor de ventas.	
Precondiciones: Debe existir la visita en estado pendiente de programar.	
Flujo normal: ▪ Usuario ingresa al sistema. ▪ Selecciona el módulo visitas. ▪ Seleccionar la opción de visitas pendientes de programar y buscar la visita deseada. ▪ Al encontrar la visita se permitirá ingresar los campos vacíos de la misma o cambiar la fecha si se desea.	

▪ Se guarda la información y se indica al usuario que el cambio fue realizado con éxito.
Flujo alternativo: Cuando se ingresan los resultados de una visita, si se ingresa la fecha de la próxima visita, se pregunta al usuario si desea programarla o dejarla en estado pendiente de programar.
Poscondiciones: La visita cambia de estado *Pendiente de Programar a Programada.*

Cuadro 26: Documento de caso de uso ver visitas programadas

Nombre:	Ver visitas programadas
Autor:	Rocío García Morales
Fecha:	25-09-2005
Descripción: Permite al usuario ver el listado de visitas programadas ordenadas por fecha y hora respectivamente u organizadas en una matriz calendario. Si selecciona la opción del listado, se puede ver el detalle completo de la visita y si selecciona la opción de calendario únicamente se puede ver la hora y el cliente de la visita. Únicamente aparecerán en este listado las visitas programadas cuya fecha sea mayor o igual a la actual, las visitas programadas cuya fecha sea menor a la actual y no tengan resultados ingresados, no aparecen en el listado (para estas visitas el sistema genera un listado preguntando si cambia el estado de la visita a realizada o cancelada). Las visitas del día aparecerán resaltadas en color rojo. El usuario puede seleccionar una visita para ver el detalle, reprogramarla o cancelarla.	
Actores: ▪ Usuario de gerencia. ▪ Usuario asesor de ventas.	
Precondiciones: El usuario debe haber ingresado al sistema para ver sus visitas programadas.	
Flujo normal: ▪ Usuario ingresa al sistema. ▪ Selecciona el módulo visitas. ▪ Si el usuario es de gerencia selecciona si desea ver las visitas programadas de un asesor específico.	
Flujo alternativo: No existe.	

Cuadro 27: Documento de caso de uso ver visitas realizadas

Nombre:	Ver visitas realizadas
Autor:	Rocío García Morales
Fecha:	25-09-2005

Descripción:

Permite al usuario ver el listado de visitas realizadas entre dos fechas, ordenadas por fecha y hora. Al seleccionar una visita, se podrá ver el detalle y los resultados obtenidos. También se podrán buscar las visitas de un cliente en un rango de tiempo para analizar la secuencia de resultados en las diferentes visitas.

Actores:

- Usuario de gerencia.
- Usuario asesor de ventas.

Precondiciones:

El usuario debe haber ingresado al sistema y seleccionado la opción de visitas realizadas.

Flujo normal:

- Usuario ingresa al sistema.
- Selecciona el módulo visitas.
- Selecciona la opción de listado de visitas realizadas y la fecha de inicio y fin.
- Si el usuario es de gerencia selecciona si desea ver las visitas realizadas de un asesor o cliente específico.

Flujo alternativo:

No existe.

Cuadro 28: Documento de caso de uso ver visitas canceladas

Nombre:	Ver visitas canceladas
Autor:	Rocío García Morales
Fecha:	25-09-2005

Descripción:

Permite al usuario, ver el listado de visitas canceladas ordenadas por fecha de cancelación, entre dos fechas especificadas. Al seleccionar una visita se podrá ver el detalle de la visita y si se realizó posteriormente alguna visita con el mismo cliente.

Actores:

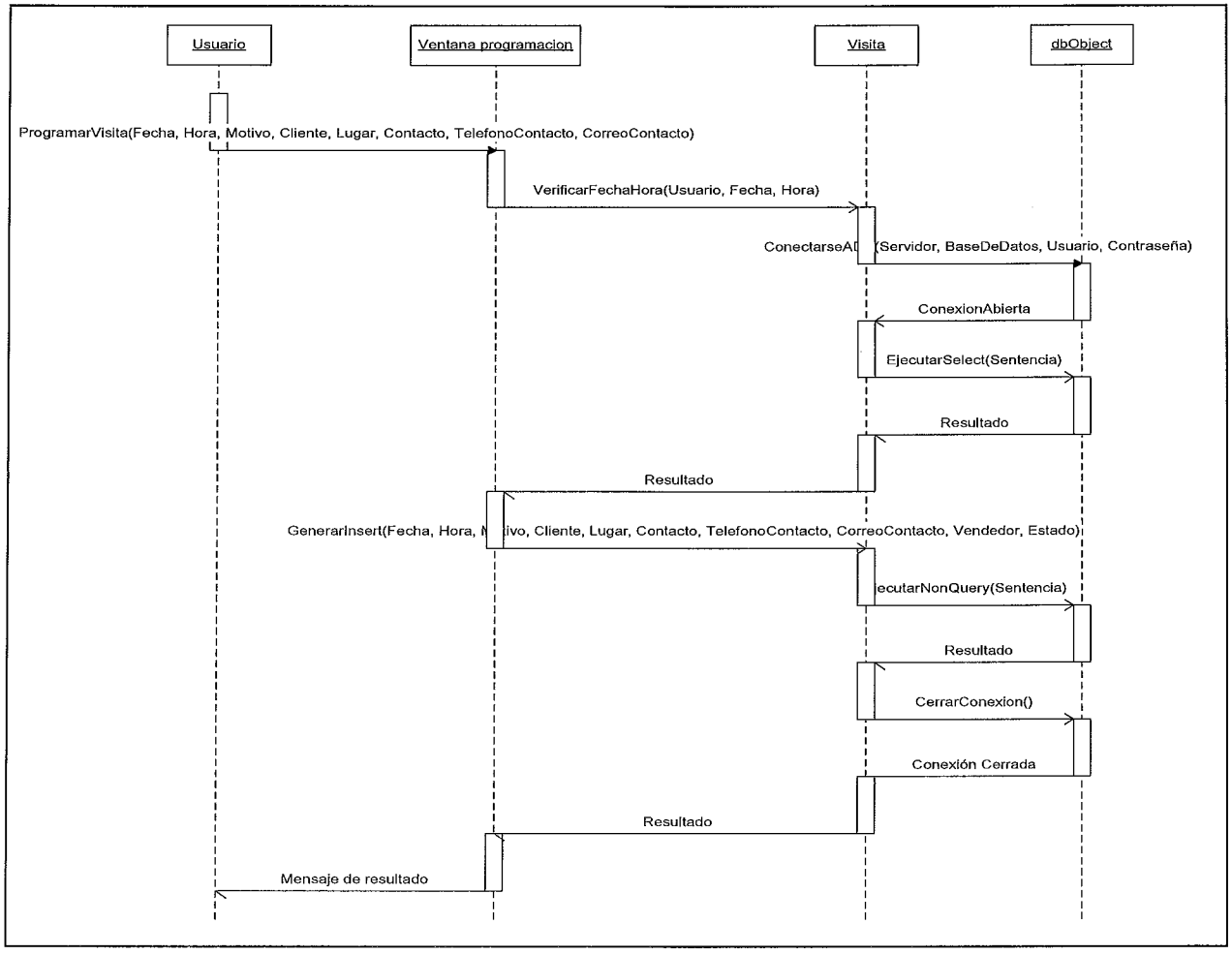
- Usuario de gerencia.
- Usuario asesor de ventas.

Precondiciones:

El usuario debe haber ingresado al sistema y seleccionado la opción de visitas canceladas.
Flujo normal: ▪ Usuario ingresa al sistema. ▪ Selecciona el módulo visitas. ▪ Selecciona la opción de listado de visitas canceladas y la fecha de inicio y fin. ▪ Si el usuario es de gerencia selecciona si desea ver las visitas canceladas de un asesor o cliente específico.
Flujo alternativo: No existe.

4. Diagramas de secuencia

Figura 42: Diagrama de secuencia para programar visita



Para programar una visita, el usuario debe seleccionar la fecha y hora de la visita y un cliente. El motivo de la visita puede escogerlo de un listado predeterminado que presenta las siguientes opciones:

- Presentación de productos y servicios.
- Recibir planos e información.
- Cierre de venta.
- Cobro.
- Entrega de documentos.
- Visita de obra.
- Mantenimiento de relación.
- Otros.

Si selecciona como motivo otro, debe ingresar un texto indicando el motivo.

El lugar, también lo selecciona de un listado predeterminado que únicamente presenta dos opciones Precon y fuera de Preco, para la que se ingresa un texto indicando el lugar.

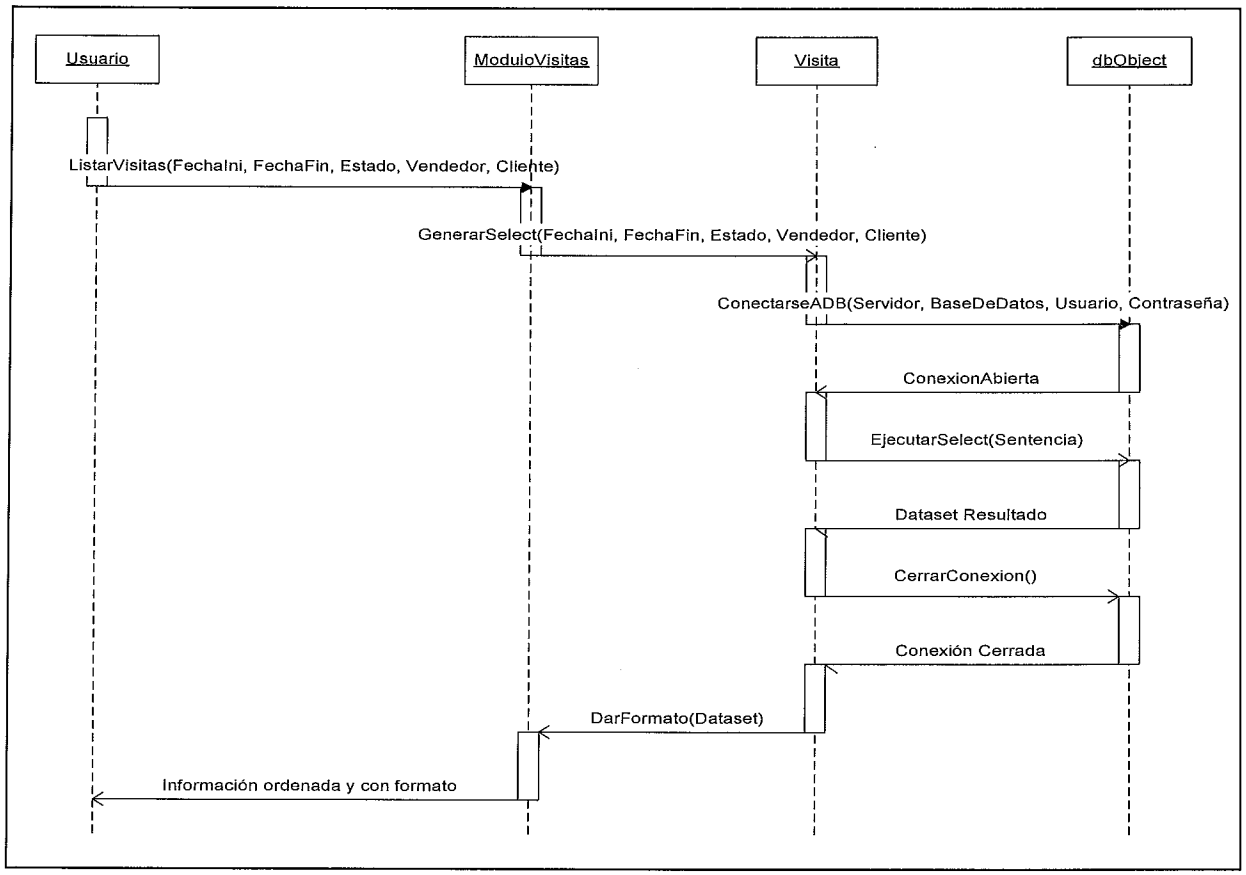
Luego, el usuario ingresa los datos del contacto (nombre, teléfono y correo) solicita programar la visita. La capa de presentación obtiene los datos a través de la ventana de programación y solicita al objeto *Visita* que verifique disponibilidad de fecha y hora para el usuario ingresado en el sistema. Para esto el objeto produce y ejecuta una sentencia *Select* que trae las visitas del usuario para la misma fecha cuya hora se encuentra en el rango [Hora Ingresada – 30 min, Hora Ingresada + 30 min.], si no existe ninguna visita que cumpla con esas condiciones entonces *Visitas* indica que se puede programar la visita. Se solicita que se genere la sentencia *Insert* que almacena los datos de la visita nueva en la tabla Visitas de la base de datos paralela. La sentencia es ejecutada a través del método EjecutaNonQuery de *dbObject* y el resultado es devuelto a través de las capas y finalmente se presenta al usuario un mensaje resultado.

Los datos almacenados para cada visita son:

- Estado: un entero que en este caso es '1' que representa el estado de visita programada.
- Fecha y hora.
- Identificador del cliente.
- Identificador del asesor: es el usuario ingresado en el sistema.
- Identificador del motivo.
- Texto del motivo.
- Identificador del lugar.
- Texto del lugar.

- Contacto.
- Teléfono del contacto.
- Correo del contacto.

Figura 43: Diagrama de secuencia para ver visitas



El diagrama presenta, la secuencia que se sigue para presentar al usuario las visitas en un estado. Los parámetros que la capa de presentación envía a la capa de aplicación son los siguientes:

- FechaIni y FechaFin: el usuario selecciona el rango de fechas en el cuál deben estar programadas las visitas. Si no selecciona ninguna de estas fechas, el listado contendrá todas las visitas del estado solicitado.

- Estado: si el usuario selecciona ver las visitas programadas el estado es '1', para las visitas reprogramadas es '2', para las canceladas '3', las pendientes de programar '0' y las realizadas '4'.

- Vendedor: si el usuario es tipo asesor, el parámetro contendrá el nombre del usuario en el sistema, si es gerente el usuario deberá escoger el vendedor de un listado y si ninguno es seleccionado podrá ver las visitas de todos los vendedores.

- Cliente: el usuario debe seleccionarlo de un listado y si no selecciona ninguno podrá ver las visitas de todos los clientes.

El objeto *Visita* recibe los parámetros anteriores y arma una cadena con la sentencia *Select* para leer de la base de datos, las visitas que cumplen con los valores de los parámetros que recibe. *Visita* solicita a *dbObject* que ejecute la sentencia a través del método EjecutarSelect que devuelve el resultado en un data-set a *Visita*, la cual lo devuelve a su vez al módulo de visitas en la capa de presentación para que presente la información resultante al usuario.

Las visitas son presentadas al usuario ordenadas por fecha y hora y el estado determina los datos que se presentan.

Encabezado para visitas programadas de un asesor

Fecha y hora	Cliente	Motivo	Lugar	Contacto	Correo	Teléfono

Encabezado para visitas canceladas y reprogramadas

Fecha y hora	Cliente	Motivo	Lugar	Contacto	Correo	Teléfono	Bitacora
							Ver datos

El botón ver datos despliega la bitácora de la visita, que guarda la fecha y hora en que la visita se creó, fecha y hora para cada cambio y fecha y hora en la que se canceló la visita.

Encabezado para visitas realizadas

Fecha y hora	Cliente	Motivo	Lugar	Detalle
				Ver datos

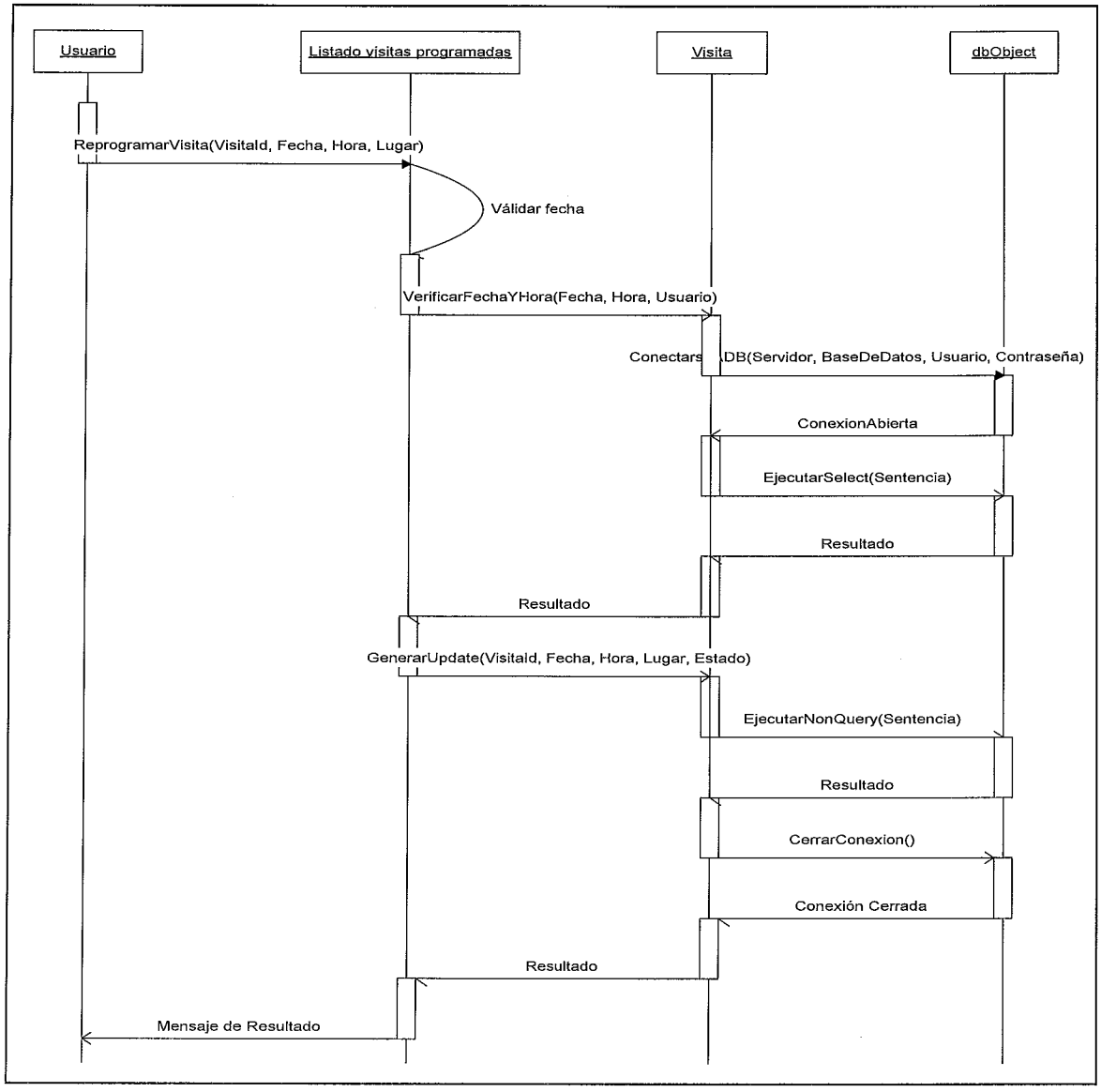
El botón ver datos despliega los datos del contacto, los resultados y hora a la que inició y terminó la visita, así como la bitácora de la misma.

Encabezado para visitas pendientes de programar

Fecha y hora	Cliente	Programar
		Programar

Si se selecciona el botón de programar, se ingresarán los demás datos de la visita.

Figura 44: Diagrama de secuencia para reprogramar visita

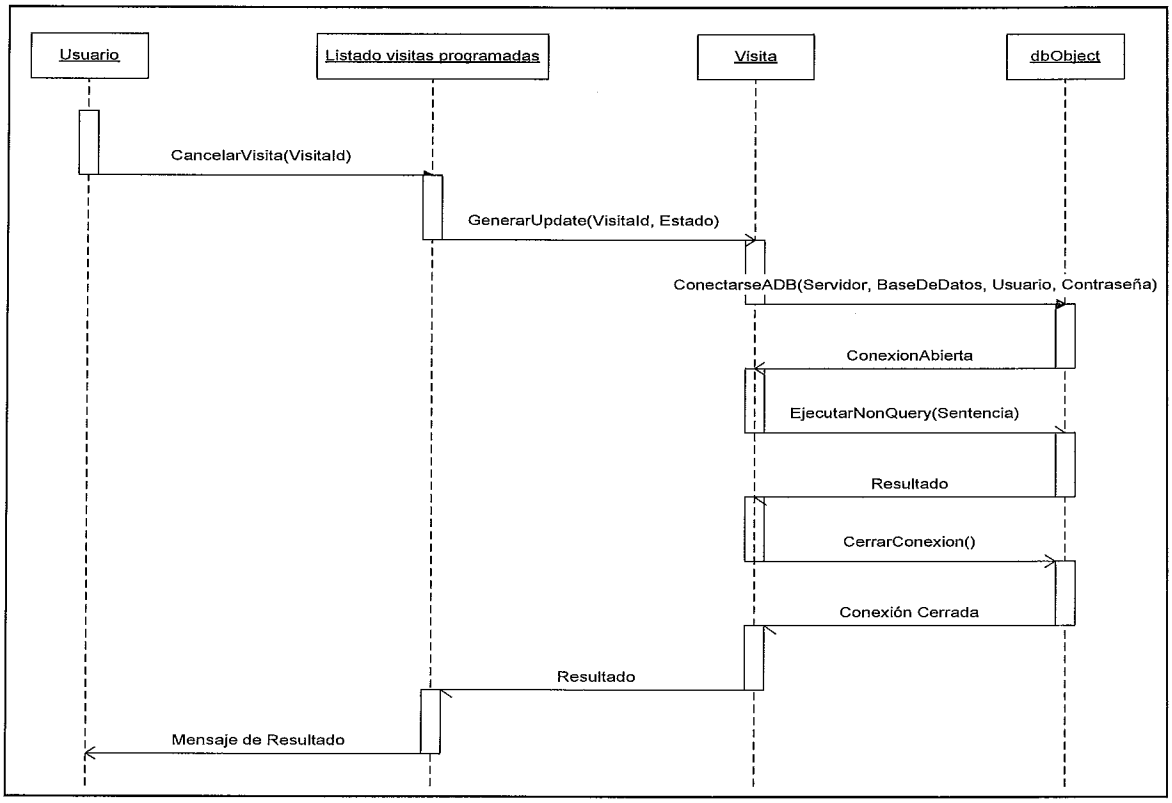


En el listado de visitas programadas se puede seleccionar una visita y reprogramarla. Esto significa cambiar la fecha, hora y/o lugar. La fila de la visita esta asociada al identificador por lo que el usuario debe seleccionar la visita y solicitar reprogramación, la aplicación presenta los valores actuales para los campos

fecha, hora y lugar y el usuario ingresa los nuevos. Al ingresar los nuevos valores la capa de presentación envía el identificador, fecha, hora y lugar a la capa de aplicación para que se verifique la disponibilidad de horario para el usuario ingresado. Si el horario está disponible se solicita al objeto *Visita* que prepare la sentencia *Update* para actualizar los datos de la visita. La sentencia es enviada como parámetro al método EjecutarNonQuery, que devuelve el resultado de la ejecución a *Visita* y ésta a la capa de presentación.

Al ejecutar el cambio en la base de datos, la bitácora es alimentada a través de un trigger en la tabla de visitas.

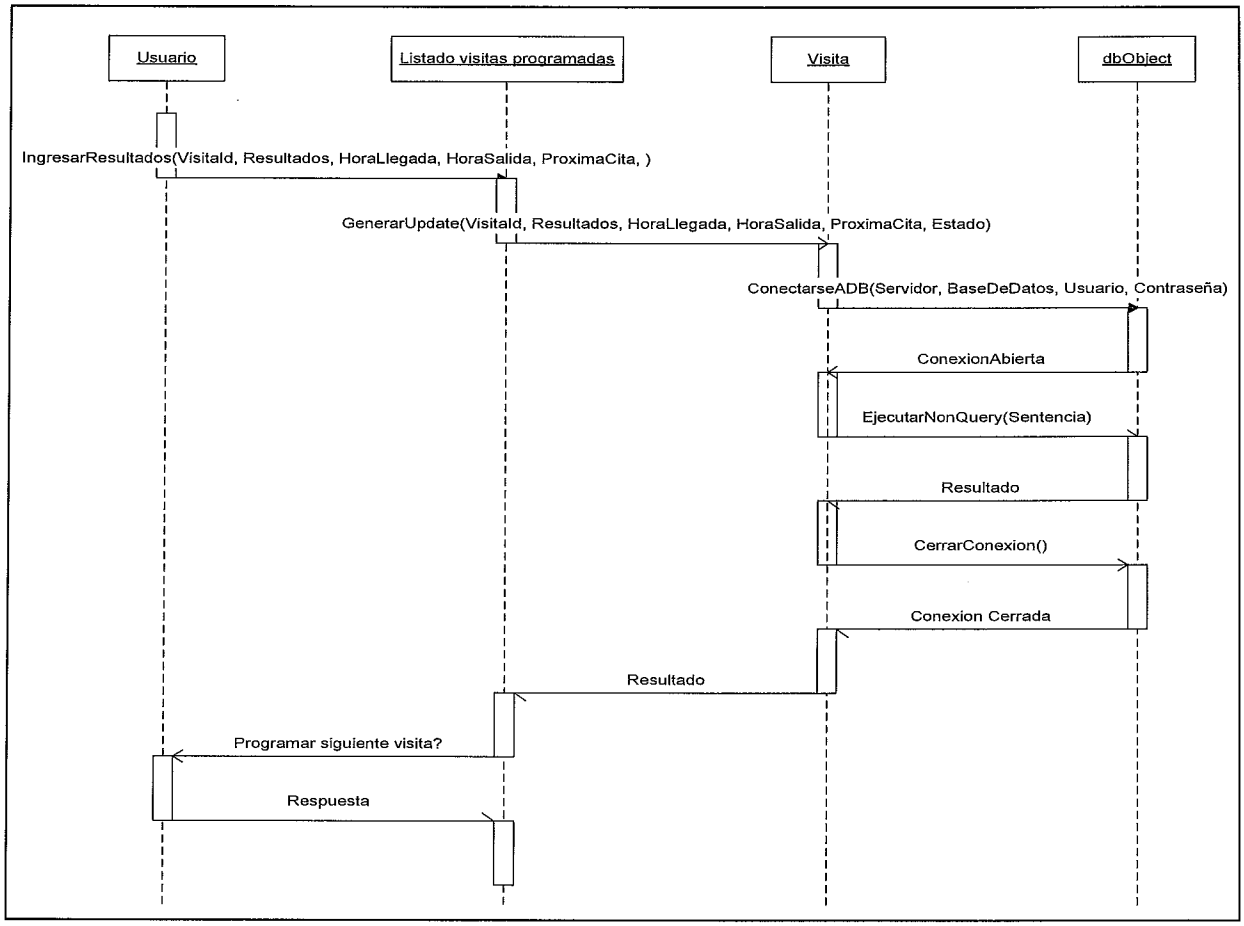
Figura 45: Diagrama de secuencia para cancelar visita



En el listado de visitas programadas, se puede seleccionar una visita y cancelarla. Esto significa cambiar el estado de la visita a '3'. La fila de la visita está asociada al identificador por lo que el usuario debe seleccionar la visita y solicitar cancelarla. La capa de presentación envía el identificador de la visita y el estado que desea asignársele. Se solicita al objeto *Visita* que prepare la sentencia *Update* para actualizar los datos de la visita. La sentencia es enviada como parámetro al método EjecutarNonQuery que devuelve el resultado de la ejecución a *Visita* y ésta a la capa de presentación.

Al ejecutar el cambio en la base de datos, la bitácora es alimentada a través de un trigger en la tabla de visitas.

Figura 46: Diagrama de secuencia para ingresar resultados de visita



De la misma forma en que se puede reprogramar una visita, se ingresan los resultados de la misma. Para esto el usuario selecciona la visita e ingresa un texto de resultado, la hora de inicio y fin de la visita y si existe una próxima visita, la fecha de la misma. La capa de presentación envía los datos que recibe del usuario a *Visita* que genera y ejecuta la sentencia *Upadte* para guardar los resultados de la visita con el identificador proveído y cambiarle el estado. Una vez más, la sentencia es ejecutada por el método EjecutarNonQuery de *dbObject*.

Luego de recibir el resultado, si se ingresó la fecha de la próxima visita, la capa de presentación pregunta al usuario si desea programar la visita o sólo dejarla pendiente de programación. Si el usuario desea programar la visita entonces se sigue la secuencia de la figura 40, en caso contrario la visita se crea únicamente con fecha, cliente y asesor, en estado pendiente de programar.

5. Especificación de clases. Las clases que intervienen en el módulo de visitas, se presentan a continuación.

Figura 47: Clase visitas

Visitas
-dbObjects dbObject
+Visitas() +GenerarSelect(in FechaIni: datetime, in FechaFin: datetime, in Estado: int, in Cliente: int, in Vendedor: int):DataSet +GenerarInsert(in Fecha: datetime, in hora: datetime, in Motivo: string, in Cliente: int, in Lugar: string, in Contacto: string, in TelefonoContacto: string, in CorreoContacto: String, in Vendedor: int, in Estado: int): int +GenerarUpdate(in VisitaId: int, in Fecha: datetime, in Hora: datime, in Lugar: string, in Lugar: string, in Estado: int): int +GenerarUpdate(in VisitaId: int, in estado: int): int +GenerarUpdate(in VisitaId: int, in Resultados: string, in HoraLlegada: datime, in HoraSalida: datetime, in ProximaCita: datetime, in estado: int): int +DetalleVisita(in VisitaId: int): DataTable +VerificarFechaYHora(in Fecha: datetime, in hora: datetime, in Vendedor: int): int

Al igual que todas las clases de la capa de aplicación, esta clase posee un miembro de tipo *dbObject* para controlar la conexión a la base de datos y ejecutar las sentencias SQL sobre la misma.

- El método GenerarSelect es llamado para devolver los datos que se presentan en los listados de visitas. Arma una cadena con la sentencia *Select* que obtiene los datos de las tablas de *Visitas, Clientes y Usuarios,* que cumplen con las condiciones indicadas en los parámetros que recibe. Luego envía esta setencia al método EjecutarSelect de su miembro *dbOject*, el cual le devuelve el resultado de la ejecución en un dataset, y el mismo es devuelto por GenerarSelect.

- El método GenerarInsert se encarga de armar y ejecutar la sentencia que inserta datos en la tabla de Visitas de la base de datos, los parámetros que recibe son los valores que guarda en los campos de la tabla.

- GenerarUpdate se encuentra sobrecargado y sus casos son:

    - Si recibe la visita, una fecha, una hora, un lugar (identificador y texto) y un entero que representa el estado, es llamado para reprogramar una visita. En este caso genera una sentencia *Update,* en la que la condición es que el identificador de la visita sea igual al parámetro *VisitaId*, y los campos fecha, hora, identificador de lugar, texto de lugar y estado son igualados a los parámetros que recibe con los mismos nombres. El método ejecuta la sentencia generada a través del método EjecutarNonQuery de *dbObject* y el resultado obtenido es devuelto a la capa de presentación.

- Si recibe la visita y el estado, es llamado para cambiar el estado de la visita a través de la sentencia *Update* que genera con los parámetros que recibe y ejecuta con el objeto *dbObject* de la clase.

- El tercer caso es utilizado para guardar los resultados de la visita, en cuyo caso también se produce una sentencia *Update* que actualiza los campos: Resultado, Fecha inicio, Fecha fin, Proxima Visita y Estado con los parámetros que recibe respectivamente, para la visita con identificador *VisitaId.* La sentencia es ejecutada y el valor de resultado devuelto a la capa de presentación.

- El método DetalleVisita genera y ejecuta una sentencia *Select* que obtiene y devuelve los datos de una visita.

- VerificarFechaYHora recibe tres parámetros (usuario, fecha y hora) y a través de *dbObject* ejecuta una sentencia *Select* para obtener las visitas del usuario que recibe como parámetro en la fecha que recibe como parámetro y que su hora de inicio sea treinta minutos o menos, antes que la hora del parámetro, o treinta minutos o menos después de la hora en el parámetro. Si al ejecutar el *Select* se lee al menos una visita, el método devuelve '0' indicando que no se puede programar una visita para el usuario en la fecha y hora recibidas. En caso contrario devuelve '1'.

Figura 48: Clase ModuloVisitas

ModuloVisitas
-Visitas Visita
+ModuloVisitas() +ListarVisitas(in FechaIni: datetime, in FechaFin: datetime, in Estado: int, in ClienteId: int, in VendedorId: int) +DetalleVisita(in VisitaId: int) +VentanaProgramacion()

Esta clase pertenece a la capa de presentación y al igual que las demás clases de esta capa, posee como miembro privado un objeto de la clase Visita que pertenece a la capa de aplicación. ModuloVisitas utiliza este miembro para comunicar y solicitar que sean ejecutadas las peticiones del usuario y para recibir los resultados de la ejecución de las peticiones.

- ListarVisitas es llamado cuando el usuario solicita ver el listado de visitas en un estado (programadas, canceladas, reprogramadas, realizadas o pendientes de programar) y envía al método GenerarSelect del objeto Visita los filtros seleccionados por el usuario (FechaInicio, FechaFin, Ven-

dedorId, ClienteId, Estado) para obtener el dataset con las visitas del listado. El dataset recibido lo presenta al usuario de forma ordenada y con formato.

- DetalleVisita envía al método, con el mismo nombre del objeto *Visita*, el identificador de una visita para obtener los datos de la visita y presentarlos al usuario.

- VentanaProgramación crea los objetos que se presentan para el ingreso de los datos en la programación de una visita y recibe los datos del usuario para enviarlos al método de la capa de aplicación que ejecuta la inserción de la nueva visita.

H. Control de licitaciones

1. Propósito. El propósito de ésta aplicación es proveer un medio para controlar las tareas que conforman una licitación y el avance de la misma. Permite al coordinador crear una licitación y sus respectivas tareas, asignándoles un responsable, fecha límite de entrega, peso dentro de la licitación y documentos asociados. La suma de los pesos de las tareas debe ser del 100% y la suma de los pesos de las tareas total o parcialmente terminadas, determinará el porcentaje terminado de la licitación. El único que puede ingresar el porcentaje terminado de una tarea es el coordinador. El técnico, únicamente podrá ver el detalle de las tareas que tiene asignadas en una licitación y subir los documentos asociados a su tarea. El personal de gerencia autorizado, podrá ver también el estado de las licitaciones y sus respectivas tareas.

2. Diagramas de casos de uso

Figura 49: Diagrama de casos de uso para usuario coordinador en la aplicación *control de licitaciones*

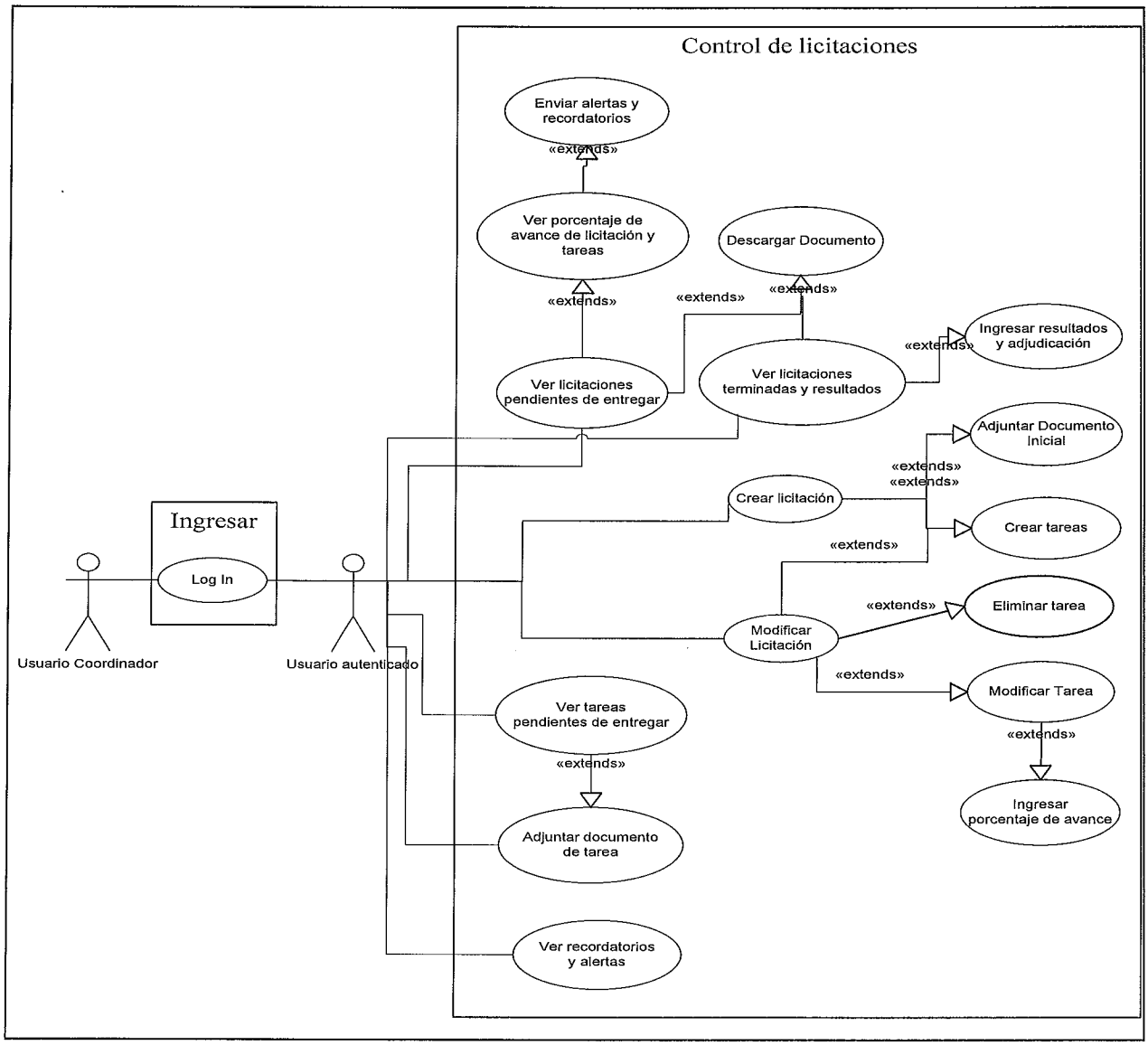


Figura 50: Diagrama de casos de uso para usuario técnico en la aplicación *control de licitaciones*

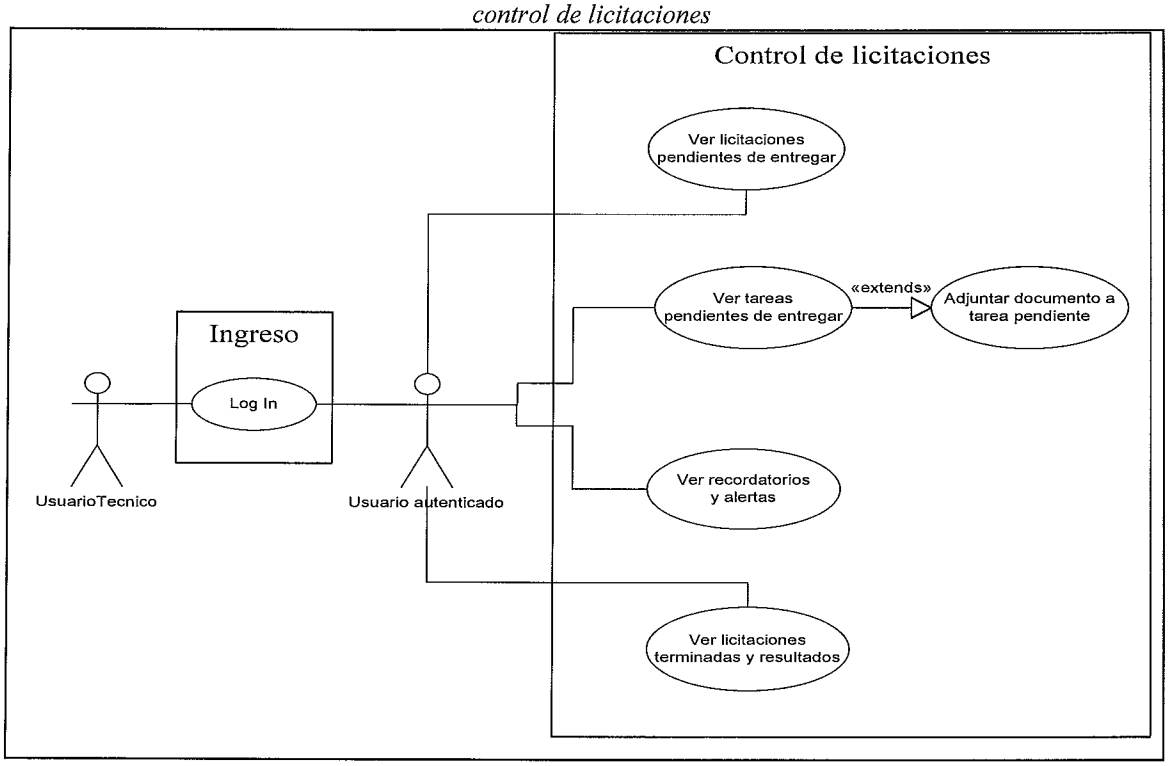
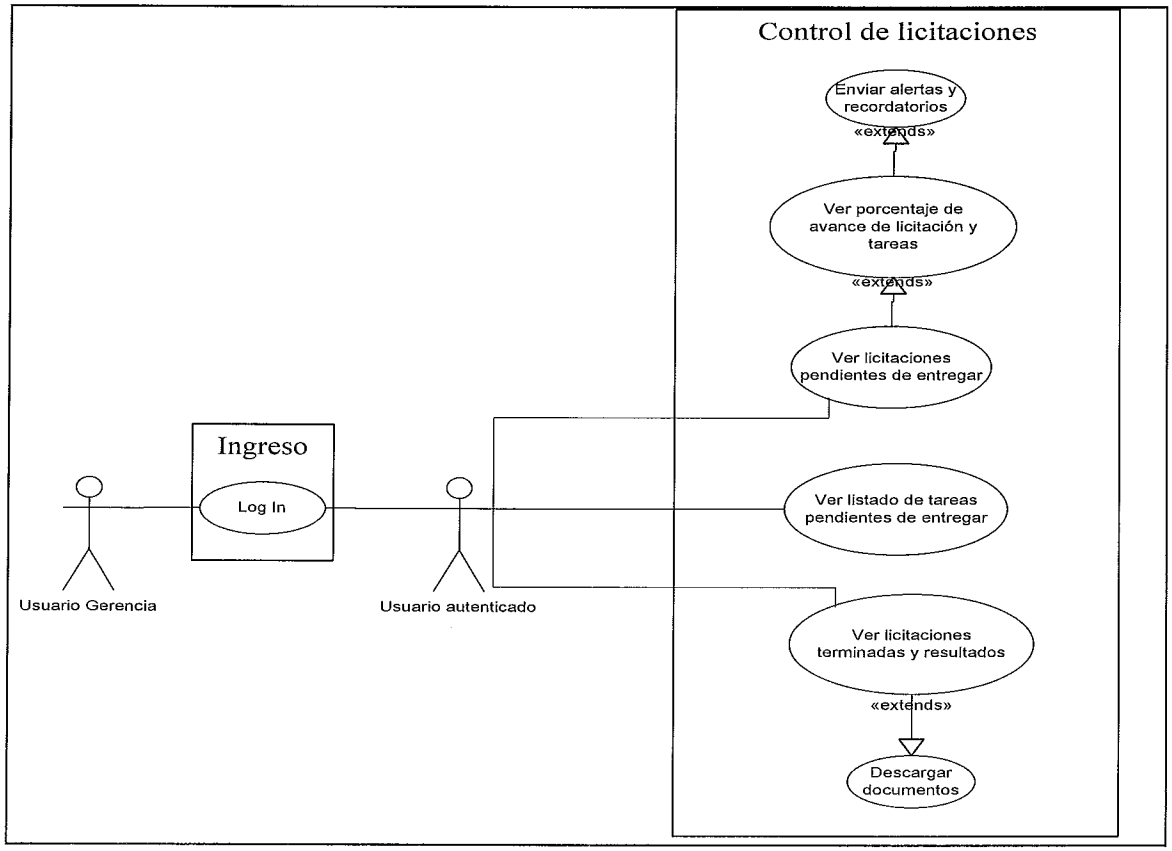


Figura 51: Diagrama de casos de uso para usuario de gerencia en la aplicación *control de licitaciones*



3. Documentos de casos de uso

Cuadro 29: Documento de caso de uso crear licitación y crear tareas

Nombre:	Crear licitación y crear tareas
Autor:	Rocío García Morales
Fecha:	26-09-2005

Descripción:

Creación de una nueva licitación con su documento adjunto, sus respectivas tareas asociadas y sus documentos adjuntos. Los datos generales que el coordinador ingresa respecto a la licitación son:

- Nombre de la licitación.
- Descripción.
- Fecha de entrega.
- Responsable.
- Documento asociado (se guarda en la base de datos en binario y es el único campo que no es requerido).

Luego de ingresar la información general de la licitación, el usuario debe crear las tareas asociadas. Por cada tarea se ingresa los siguientes datos:

- Descripción de la tarea.
- Responsable.
- Fecha límite de entrega.
- Peso porcentual en relación a todas las tareas de la licitación.
- Lista de documentos asociados, para los que se ingresa el titulo y la descripción, ya que el técnico adjunta posteriormente el documento asociado.

Al crear cada tarea, se guarda en la base de datos con un porcentaje inicial de avance de 0% y se asocia a la licitación actual. Los pesos de las tareas deben sumar el 100% de la licitación y la fecha límite de entrega de la tarea debe ser menor o igual a la fecha de entrega de la licitación.

Habiendo ingresado los datos anteriores la licitación puede ser creada.

Actores:

- Usuario coordinador.

Precondiciones:

El usuario debe haber sido autenticado y autorizado para ingresar al sistema.

Flujo normal:

- Usuario ingresa al sistema.
- Selecciona el módulo licitaciones.
- Seleccionar la opción de crear una nueva licitación.
- Ingresa los campos requeridos en la forma de creación de licitaciones y adjunta el documento asociado si éste existe.
- El usuario crea una lista de tareas que componen la licitación indicando para cada una el responsable, fecha límite de entrega, descripción y el peso porcentual.
- Para cada tarea que se crea, se debe indicar los documentos asociados, si es que los tiene.

▪ El usuario solicita guardar los datos. ▪ El sistema revisa si los pesos de las tareas suman el cien por ciento y si todos los campos requeridos contienen información válida. ▪ Si no hubo ningún error en el paso anterior, la información es almacenada en la base de datos paralela en la que se crea un identificador único para la licitación, se informa al usuario que se creó la licitación con éxito y se envía un correo a los responsables de las tareas indicando que existe una licitación nueva (con su identificador) en la que tiene una tarea asignada. ▪ En caso contrario se indica al usuario el error encontrado.
Flujo alternativo: No existe.
Poscondiciones: Existe una nueva licitación en el sistema.

Cuadro 30: Documento de caso de uso ver licitaciones pendientes de entregar

Nombre:	Ver licitaciones pendientes de entregar
Autor:	Rocío García Morales
Fecha:	26-09-2005
Descripción: Permitirá al usuario de gerencia y coordinador ver el listado de todas las licitaciones pendientes de entregar, ordenadas por fecha de entrega, y la opción de ver su detalle. El detalle de la licitación contiene los datos generales que fueron ingresados por el coordinador (nombre, descripción, fecha de entrega, responsable y documento asociado), la fecha de creación de la licitación en el sistema y la descripción de las tareas, ordenadas también por fecha de entrega. En el detalle de cada tarea, se incluye el porcentaje terminado, si la tarea ya debió ser entregada y el porcentaje terminado aún no es de 100%, la tarea aparece resaltada con rojo y se ofrecerá al usuario coordinador o de gerencia la opción de enviar un mensaje recordando que la tarea ya debió ser entregada. El usuario coordinador o de gerencia también podrá descargar los archivos que han sido guardados de las diferentes tareas. El usuario de gerencia o coordinador podrá ver el porcentaje de avance total de la licitación. Este porcentaje se obtiene de la siguiente forma: ▪ Cada tarea tiene un peso respecto al 100% de la licitación ▪ El porcentaje que aporta una tarea al total terminado de la licitación es: o Peso * (PorcentajeTerminado/100) ▪ La sumatoria del término anterior para cada una de las tareas de la licitación, indica el porcentaje terminado de la licitación. El usuario técnico únicamente podrá ver el listado de las licitaciones pendientes de entregar en las que el tiene alguna tarea asignada y dentro de cada licitación sólo podrá ver sus tareas.	
Actores:	

▪ Usuario gerencia. ▪ Usuario coordinador. ▪ Usuario técnico.
Precondiciones: El usuario debe haber sido autenticado y autorizado para ingresar al sistema; deben existir en el sistema licitaciones pendientes de entregar, es decir con fecha menor que la del sistema.
Flujo normal: ▪ Usuario ingresa al sistema. ▪ Selecciona el módulo licitaciones. ▪ El sistema obtiene el tipo de usuario y si el usuario es técnico filtra únicamente las licitaciones pendientes de entregar en las que el usuario tiene alguna tarea asignada. ▪ Lo primero que el usuario puede ver al ingresar al módulo es el listado de licitaciones pendiente de entregar ordenadas por fecha de entrega. Si la fecha de entrega de la licitación ya pasó y no se han ingresado resultados de la licitación, la licitación aparece resaltada con rojo. ▪ El usuario podrá filtrar las licitaciones en un rango de fechas específico u ordenarlas por algún otro campo.
Flujo alternativo: No existe.

Cuadro 31: Documento de caso de uso ver tareas pendientes de entregar

Nombre:	Ver tareas pendientes de entregar
Autor:	Rocío García Morales
Fecha:	26-09-2005
Descripción: El usuario podrá ver ordenadas por fecha límite de entrega las tareas en la que es él responsable y aún no tienen el 100% en porcentaje de avance. Podrá ver el detalle de la tarea y adjuntar algún documento asociado si desea. Las tareas cuya fecha límite de entrega ha vencido aparecerán resaltadas en rojo.	
Actores: ▪ Usuario gerencia. ▪ Usuario coordinador. ▪ Usuario técnico.	
Precondiciones: El usuario debe haber sido autenticado y autorizado para ingresar al sistema, deben existir en el sistema tareas pendientes de entregar, es decir con fecha límite de entrega menor que la del sistema.	
Flujo normal: ▪ Usuario ingresa al sistema. ▪ Selecciona el módulo licitaciones. ▪ Deberá seleccionar la opción de listado de tareas pendientes. ▪ El sistema obtiene el identificador del empleado conectado en el sistema y filtra únicamente las	

tareas de las que él es responsable. ▪ El usuario puede ver el listado de tareas.
Flujo alternativo: En el listado de licitaciones terminadas o pendientes de entregar por cada licitación el usuario podrá ver únicamente sus tares en la misma.

Cuadro 32: Documento de caso de uso modificar licitación, modificar tarea, ingresar porcentaje de avance

Nombre:	Modificar Licitación, Modificar tarea, Ingresar porcentaje de avance.
Autor:	Rocío García Morales
Fecha:	26-09-2005

Descripción:

Se podrán modificar los siguientes datos de una licitación:

- Fecha Entrega.
- Descripción.
- Responsable.
- Documento asociado.

Al realizarse una modificación, se guardará en su bitácora la fecha, hora y descripción de la modificación.

Modificar una licitación, en algunos casos, implicará modificar una tarea o eliminarla de la lista. La tarea se eliminará aún cuando ya se haya ingresado el porcentaje de avance de la misma y se tendrá que considerar que los pesos de las tareas nuevamente deben sumar 100.

Se permitirá cambiar los siguientes datos de una tarea:

- Responsable.
- Descripción.
- Fecha de límite de entrega.
- Peso.

Se podrá ingresar el porcentaje terminado de la tarea (un valor entre 0 y cien); si el porcentaje ingresado es 100, el sistema guardará la fecha en la que la tarea se terminó y si con el porcentaje de la tarea se alcanza también el 100% de la licitación, se guarda para la licitación la fecha en la que se terminó.

Dentro de una tarea específica, también se podrá modificar la lista de documentos asociados. Por cada documento se permitirá cambiar los siguientes datos:

- Título.
- Descripción.
- Documento.

Si se cambia el documento adjunto, se guardará en la bitácora una entrada indicando la fecha y hora en la que se adjuntó el archivo.

También se podrá crear una nueva tarea (caso de uso explicado anteriormente) para la licitación.

Actores:

- Usuario coordinador

Precondiciones:

El usuario debe haber sido autenticado y autorizado para ingresar al sistema. La licitación debe haber sido creada en el sistema con sus respectivas tareas y el porcentaje terminado debe ser menor al 100%, de lo contrario no se podrá modificar la licitación.

Flujo normal:

- Usuario ingresa al sistema.
- Selecciona el módulo licitaciones.
- En el listado de licitaciones pendientes de entregar
- Selecciona la licitación deseada para modificación
- El usuario puede:
  - Modificar los datos generales de la licitación.
  - Modificar las tareas de la licitación.
    - Cambiar nombre, descripción, responsable o fecha límite de entrega.
    - Ingresar porcentaje de avance de la tarea.
    - Cambiar datos de documentos asociados a una tarea o cambiar documento adjunto.
  - Eliminar alguna tarea de la lista de tareas.
  - Crear una nueva tarea.
- Guarda los cambios .
- Si los datos son válidos, el sistema informa al usuario que la tarea ha sido modificada satisfactoriamente.

Flujo alternativo:

Seleccionar la opción de modificar licitación ingresando el número de licitación explícitamente.

Poscondiciones:

Al menos una licitación o tarea ha sido modificada.

Cuadro 33: Documento de caso de uso ver licitaciones terminadas y resultados

Nombre:	Ver licitaciones terminadas y resultados
Autor:	Rocío García Morales
Fecha:	26-09-2005

Descripción:

Permitirá al usuario coordinador y de gerencia ver el listado de todas las licitaciones terminadas (con porcentaje de avance de 100%) ordenadas por fecha de entrega en un rango de fechas determinado por el usuario.

Por cada licitación en el listado, se presentará la opción de ver el detalle, que mostrará:

- Los datos generales (nombre, descripción, fecha de entrega, responsable y documento asociado).
- La fecha de creación de la licitación en el sistema.

- Fecha en que se terminó.
- El listado de tareas, ordenadas también por fecha de entrega. Este listado también presentará la opción de ver el detalle de cada tarea que incluye:
    - Datos generales de la tarea (Nombre, descripción, responsable, fecha límite de entrega, peso).
    - Fecha en la que se terminó.
    - Lista de documentos asociados, permitiendo la descarga de los mismos.

  Las tareas terminadas fuera de tiempo serán resaltadas con rojo
- La bitácora de modificaciones.
- Resultados y adjudicación, si ya fueron ingresados, en caso contrario se permitirá al usuario coordinador ingresar un texto de resultados de la licitación y cambiar el estado de la licitación a *adjudicada a Precon,* si ese es el caso.

Se marcarán en rojo las licitaciones que se terminaron después de la fecha de entrega.

Al final se indicará el porcentaje de tareas terminadas antes o en la fecha límite de entrega y el listado de los responsables que entregaron sus tareas en la fecha establecida y los que entregaron sus tareas tarde.

El usuario técnico, únicamente podrá ver las licitaciones terminadas en las que él tiene alguna tarea asignada y dentro de la licitación únicamente podrá ver sus tareas y si fueron entregadas en tiempo o no.

Actores:

- Usuario coordinador.
- Usuario asesor.
- Usuario gerencia.

Precondiciones:

El usuario debe haber sido autenticado y autorizado para ingresar al sistema. Para que la licitación sea presentada en el listado, la suma del porcentaje de avance de las tareas que componen la licitación debe ser 100 y si el usuario es técnico debe tener una tarea asignada en la licitación. Además la fecha de entrega de la licitación debe de estar contenida en el rango de fechas indicado por el usuario.

Flujo normal:

- Usuario ingresa al sistema.
- Selecciona el módulo licitaciones.
- Seleccionar la opción de ver licitaciones terminadas y resultados.
- Ingresa el rango de fechas.
- El sistema filtra las licitaciones dependiendo el tipo de usuario y en base al rango ingresado.
- El usuario puede ver el listado de las licitaciones y su respectivo detalle.
- Si es usuario coordinador puede ingresar los resultados y adjudicación de la licitación

Flujo alternativo:

No existe.

Poscondiciones:

Pueden haberse ingresado los resultados y adjudicación de alguna licitación.

Cuadro 34: Documento de caso de uso ver recordatorios y alertas

Nombre:	Ver recordatorios y alertas
Autor:	Rocío García Morales
Fecha:	26-09-2005

Descripción:

Permitirá al usuario técnico, ver los recordatorios y alertas enviados por el coordinador o algún usuario de gerencia respecto a tareas no terminadas. Y al usuario coordinador los enviados por algún usuario de gerencia.

Los mensajes serán ordenados por fecha y hora en la que fueron enviados en dos categorías:

- Bandeja de entrada.
- Papelera.

El usuario podrá enviar un mensaje a la papelera o eliminarlo completamente. En base a lo anterior el mensaje puede tener los siguientes estados:

- No leído en bandeja de entrada.
- Leído en bandeja de entrada.
- No leído en papelera.
- Leído en papelera.
- Eliminado.

Actores:

- Usuario coordinador.
- Usuario técnico.

Precondiciones:

El usuario debe haber sido autenticado y autorizado para ingresar al sistema. El sistema debe traer los mensajes cuyo destinatario sea el empleado conectado al sistema.

Flujo normal:

- Usuario ingresa al sistema.
- Selecciona el módulo recordatorios y alertas.
- Selecciona bandeja de entrada o papelera.
- Selecciona de la lista de recordatorios o alertas el mensaje que desea leer o eliminar.

Flujo alternativo:

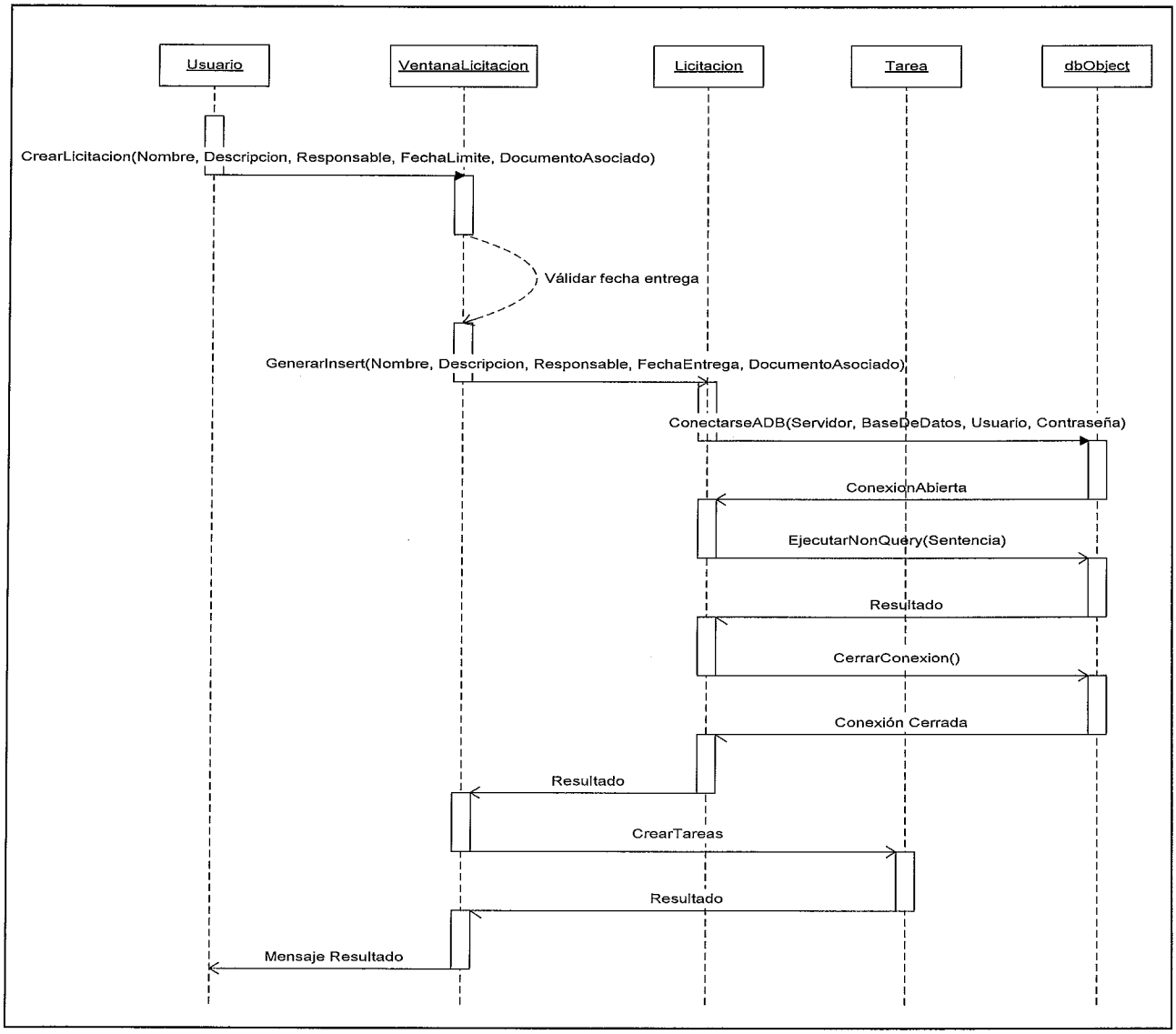
No existe.

Poscondiciones:

El estado del mensaje puede cambiar.

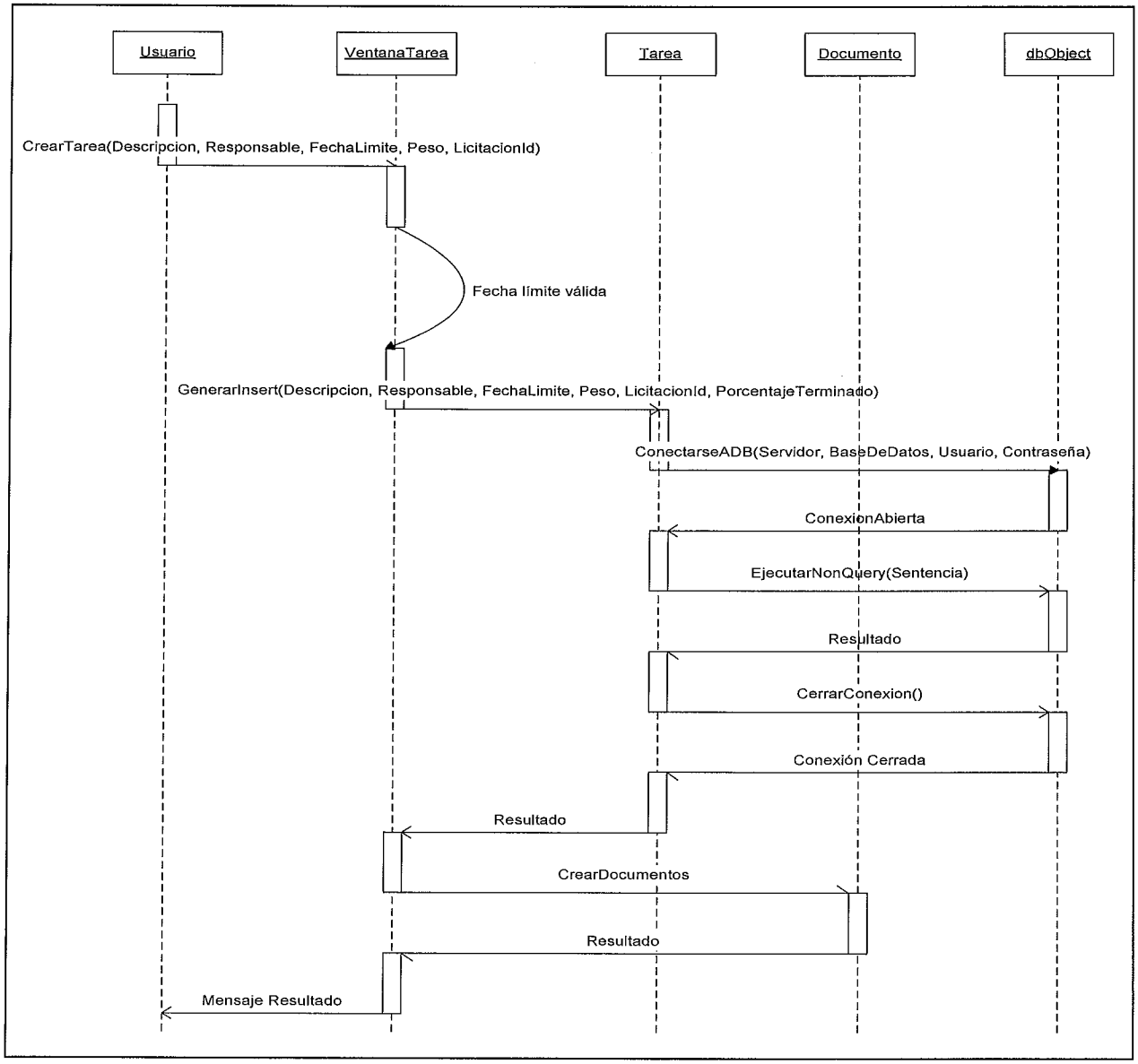
4. Diagramas de secuencia

Figura 52: Diagrama de secuencia para crear licitación



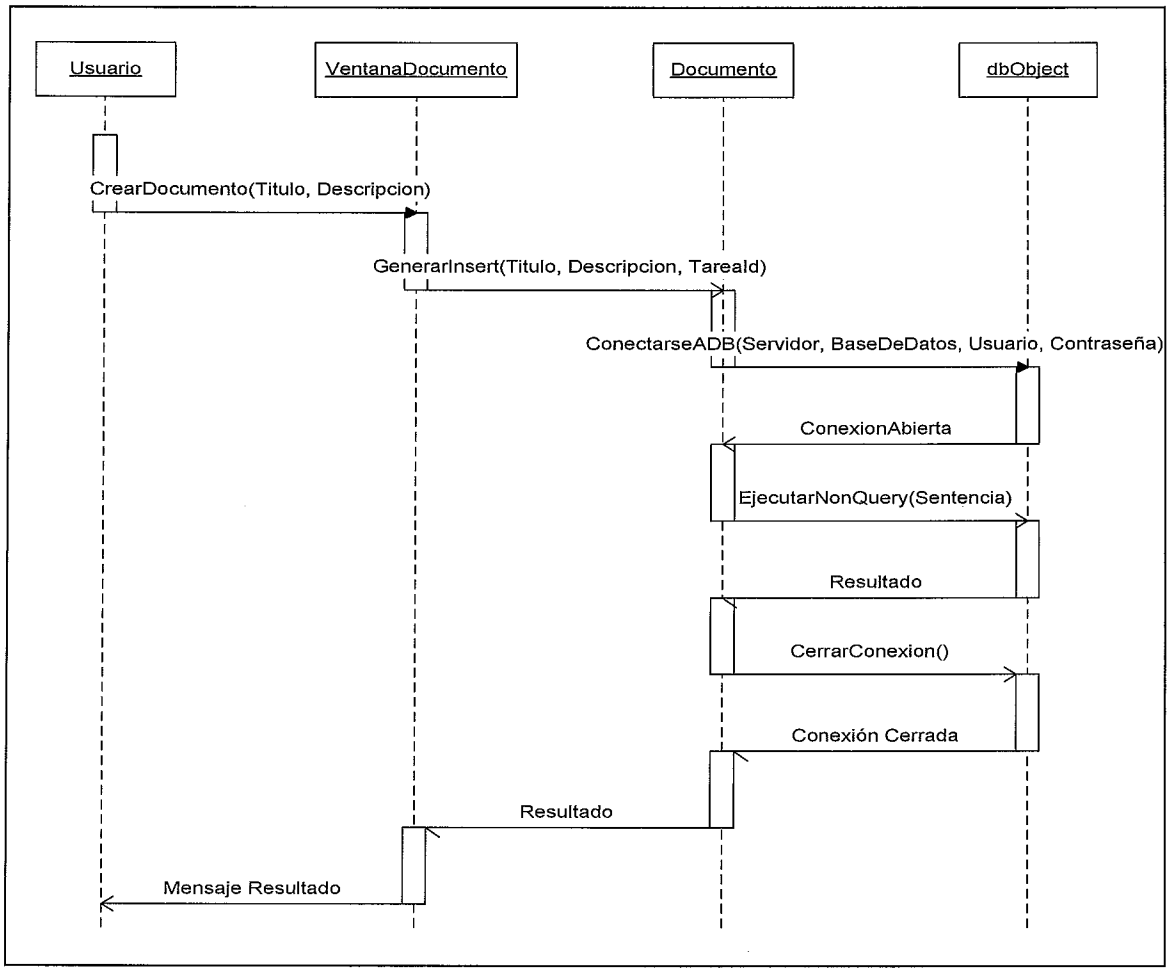
Para crear una licitación, el usuario utiliza la ventana licitación e ingresa los datos de la nueva licitación. Los datos son validados y si son correctos son enviados al método *GeneraInsert* del objeto *Licitación* para que genere la sentencia de inserción de los datos en la tabla de licitaciones y solicite al objeto *dbObject* que lo ejecute. Si la sentencia es ejecutada correctamente se procede a crear las tareas de la licitación cuya secuencia se presenta en la siguiente figura.

Figura 53: Diagrama de secuencia para crear tarea



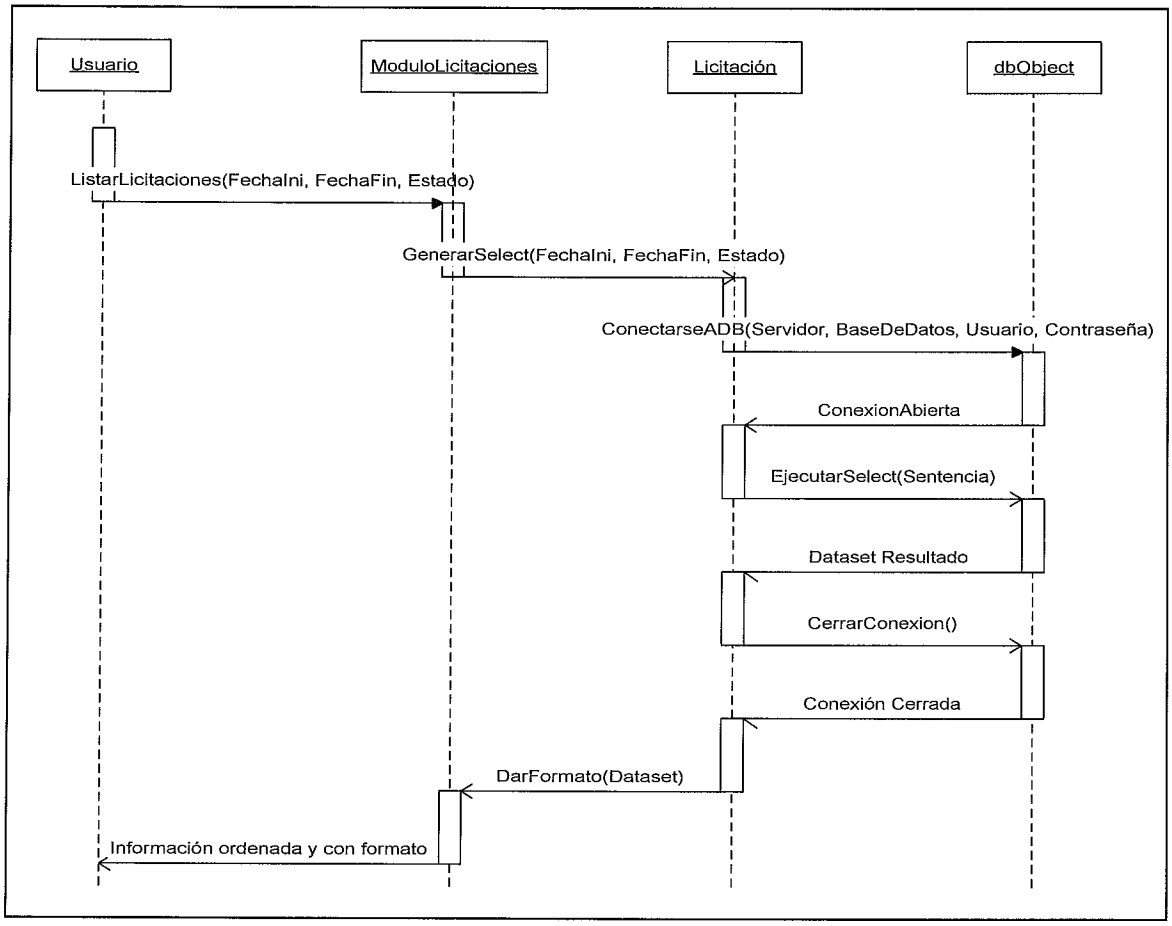
Para crear una tarea nueva el usuario utiliza la ventana tarea e ingresa los datos. Los datos son validados y si son correctos son enviados, junto con el identificador de la licitación, al método *GeneraInsert* del objeto *Tarea* para que genere la sentencia de inserción de los datos en la tabla de tareas y solicite al objeto *dbObject* que lo ejecute. Si la sentencia es ejecutada correctamente se procede a crear los documentos asociados a la tarea cuya secuencia se presenta en la siguiente figura.

Figura 54: Diagrama de secuencia para crear un documento



Para crear un documento, el usuario utiliza la ventana documento e ingresa los datos, únicamente ingresa el título y la descripción. Los datos son validados y si son correctos son enviados, junto con el identificador de la tarea, al método *GeneraInsert* del objeto *Documento* para que genere la sentencia de inserción de los datos en la tabla de documentos y solicite al objeto *dbObject* que lo ejecute. *DbObject* devuelve el resultado de la ejecución a *Documento* y éste a la ventana de documento para que devuelva un mensaje de resultado al usuario.

Figura 55:Diagrama de secuencia para ver listado de licitaciones



El módulo de licitaciones ofrece al usuario ver los siguientes listados de licitaciones:

- Pendientes de entregar.
- Terminadas.

Las licitaciones pendientes de entregar son aquellas para las que todas sus tareas tienen el 100% de porcentaje terminado.

El valor de la variable estado depende del listado que seleccione el usuario. El estado es '0' para licitaciones pendientes de entregar y '1' para terminadas. Luego el usuario selecciona un rango de fechas en el que debe estar la fecha de entrega de las licitaciones. Si no se ingresa el rango, el usuario puede ver todas las licitaciones del estado seleccionado. Los tres valores, estado, fechainicio y fechafin son enviados al método GeneraSelect del objeto licitación para que genere la sentencia *Select* que obtiene los datos de la tabla de licitaciones y usuarios que cumplen con las condiciones indicadas en los parámetros. El método

solicita la ejecución de la sentencia al objeto *dbObject,* que devuelve el dataset con el resultado. El dataset es devuelto a la capa de presentación para mostrar el listado al usuario.

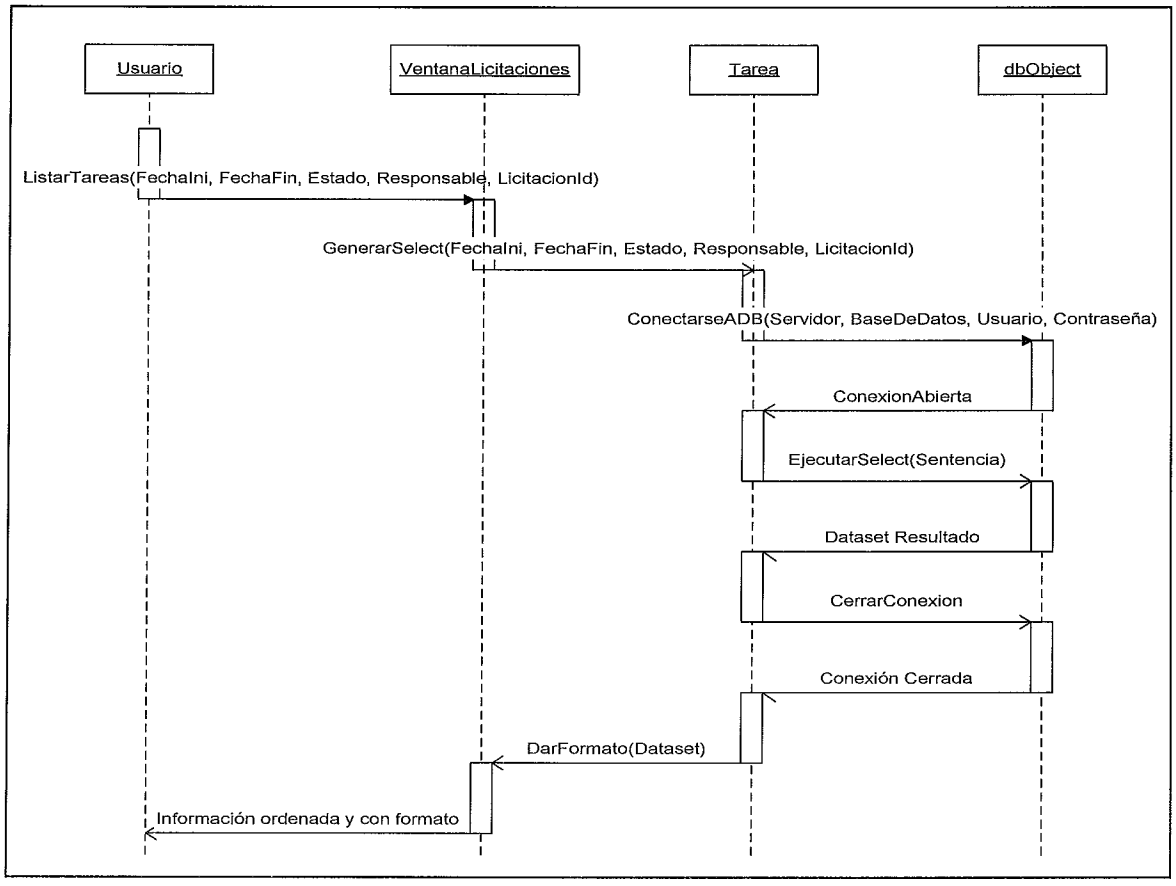
El encabezado del listado de licitaciones pendientes de entregar es el siguiente:

Id	Nombre	Fecha de entrega	Responsable	Documento	%terminado	Detalle
				Descargar		Ver detalle

Las licitaciones cuya fecha de entrega ha vencido y aún no tienen el cien por ciento terminado aparecerán resaltadas con rojo.

Para cada licitación del listado, se presenta la opción de descargar el documento asociado o ver su detalle. El detalle de la licitación incluye además de la información general presentada en el listado, la fecha de creación, descripción y el listado de tareas. La figura a continuación presenta el diagrama de secuencia para ver el listado de tareas.

Figura 56: Diagrama de secuencia para ver listado de tareas



Si se desea ver el listado de tareas de una licitación, existen varios casos:

1. El usuario es coordinador o gerente y desea ver las tareas de una licitación.
2. El usuario es técnico, coordinador o gerente y desea ver las tareas que tiene pendientes de entregar en una licitación.
3. El usuario es técnico, coordinador o gerente y desea ver las tareas que tiene pendientes de entregar en todas las licitaciones.
4. El usuario es técnico y desea ver las tareas que ha entregado en un rango de fecha.
5. El usuario es técnico, coordinador o gerente y desea ver todas las tareas entregadas en un rango de fecha.

Cada caso genera una combinación de parámetros distintos para generar el *Select* que trae los datos que cumplen con esas condiciones. Las combinaciones de los parámetros para los casos anteriores son las siguientes:

1. FechaIni = null, FechaFin= null, Estado=null, ResponsableId = null, LicitacionId = licitación actual.
2. FechaIni = null, FechaFin= null, Estado='0', ResponsableId = usuario actual, LicitacionId = licitación actual.
3. FechaIni = null, FechaFin= null, Estado='0', ResponsableId = usuario actual, LicitacionId = null.
4. FechaIni = fecha ingresada, FechaFin= fecha ingresada, Estado='1', ResponsableId = usuario actual, LicitacionId = null.
5. FechaIni = fecha ingresada, FechaFin= fecha ingresada, Estado='1', ResponsableId = null, LicitacionId = null.

Estos valores son enviados a la capa de aplicación para que el método GeneraSelect arme la sentencia que trae las tareas correspondientes a cada uno de los casos anteriores. La sentencia es ejecutada por el método EjecutaSelect de *dbObject* y se devuelve el dataset resultado para que las tareas sean presentadas al usuario.

El encabezado del listado de las tareas de una licitación, es el siguiente:

Fecha límite de entrega	Id. Tarea	Descripción	Responsable	Peso	%terminado	Documentos
						Ver listado

El encabezado del listado de tareas pendientes de entregar, de un usuario en una licitación es el siguiente:

Fecha límite de entrega	Id. Tarea	Descripción	Peso	%terminado	Documentos
					Ver listado

El encabezado del listado de tareas pendientes de entregar, de un usuario en todas las licitaciones:

Fecha límite de entrega	Nombre licitación	Id Tarea	Descripción	Peso	%terminado	Documentos
						Ver listado

El encabezado del listado de las tareas terminadas en un rango de fechas, de un usuario es el siguiente:

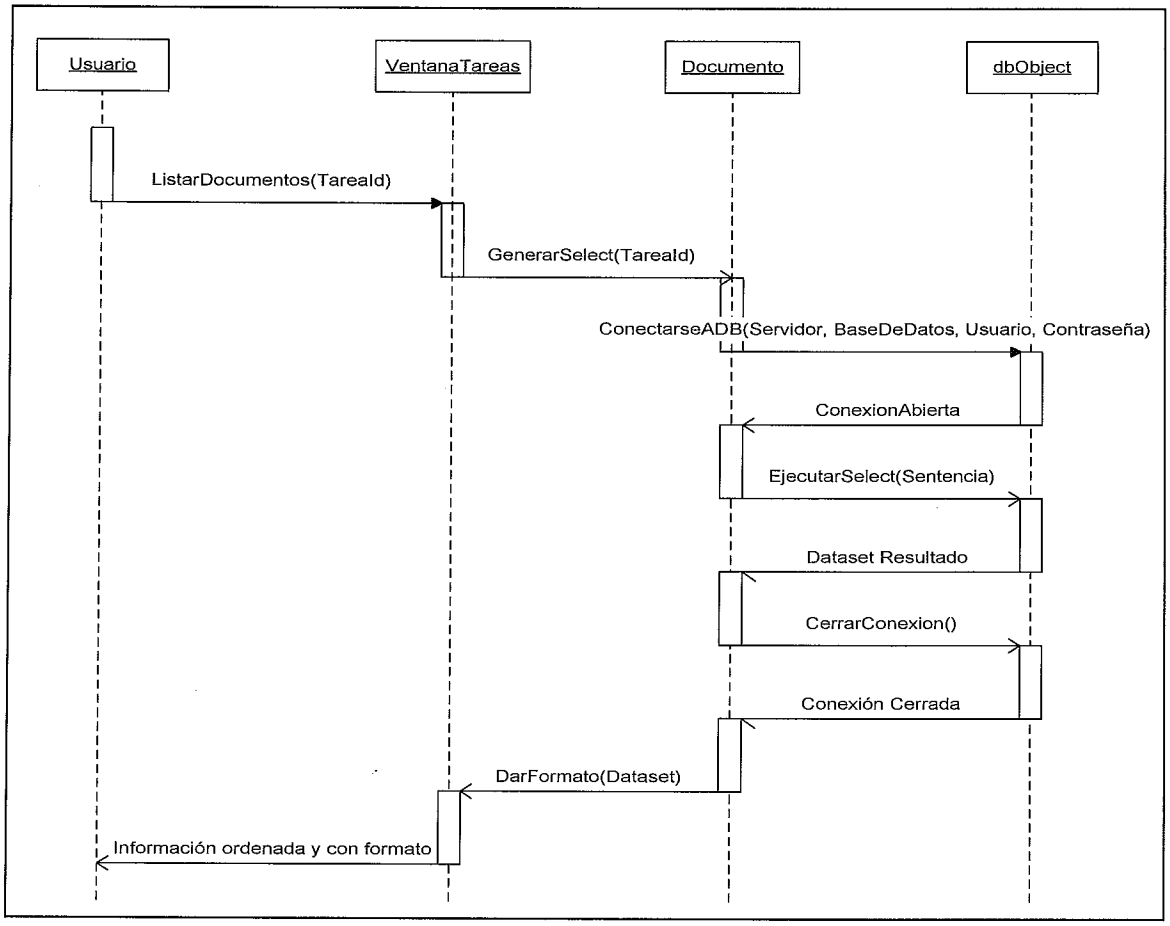
Fecha límite de entrega	Fecha terminada	Nombre licitación	Id. Tarea	Descripción	Peso	%terminado	Documentos
							Ver listado

El encabezado del listado de las tareas terminadas en un rango de fechas es:

Fecha límite de entrega	Fecha terminada	Nombre licitación	Id. Tarea	Descripción	Responsable	Peso	%terminado	Documentos
								Ver listado

En todos los listados cada tarea permite ver el listado de documentos asociados, el diagrama de secuencia para ver el listado es el que sigue.

Figura 57: Diagrama de secuencia para ver listado de documentos



En este diagrama interviene un objeto documento que recibe de la capa de presentación el identificador de la tarea a la que deberán pertenecer los documentos. Este objeto arma y ejecuta (a través de *dbObject*) la sentencia *Select* que trae los documentos de la tarea que recibe como parámetro. El resultado es devuelto en un dataset para ser presentado al usuario.

El encabezado del listado de documentos es el que sigue:

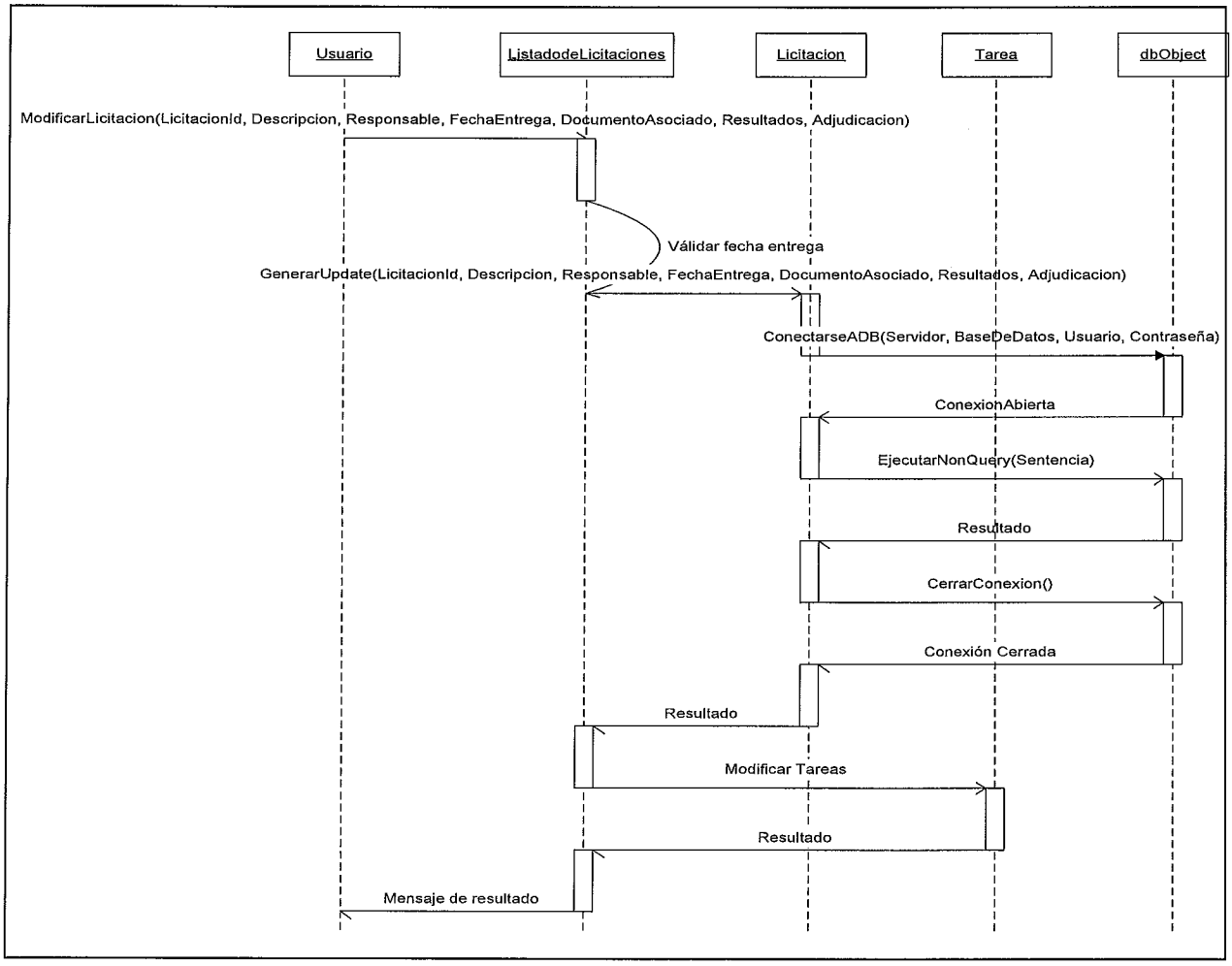
Identificador	Título	Descripción	Documento
			Descargar

Por cada documento del listado, se permitirá descargar el archivo adjunto si éste existe.

Se pueden modificar todos los datos de una licitación excepto el identificador. En el listado de licitaciones se selecciona la licitación, se ingresan los nuevos datos y se solicita guardar los cambios. Los datos

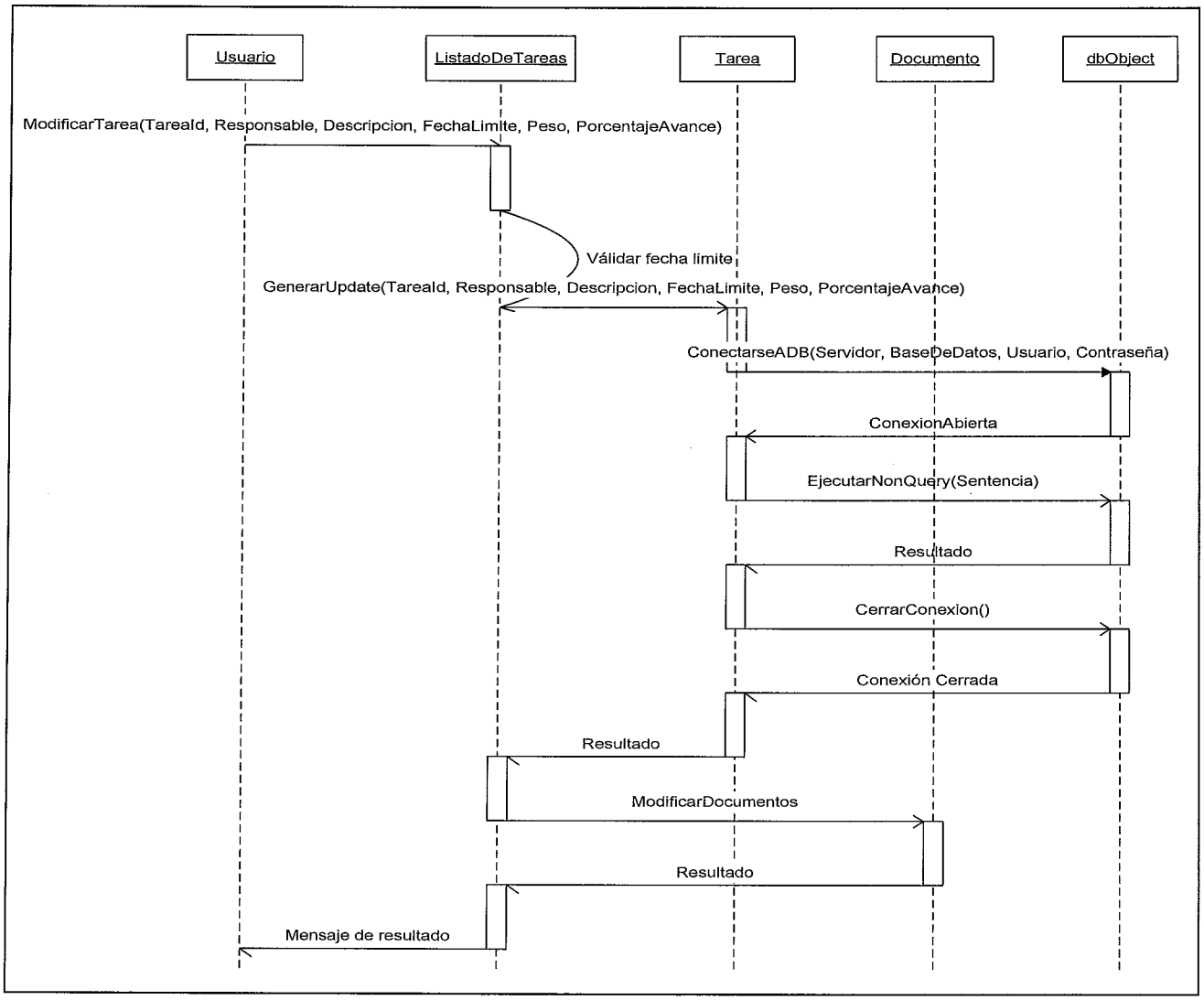
ingresados son verificados y se solicita al objeto *Licitación* que produzca la sentencia *Update* para actualizar la licitación con los datos que recibe en sus parámetros. Esta sentencia es enviada a la capa de datos para ser ejecutada a través del método EjecutarNonquery. Modificar una licitación puede implicar modificar sus tareas, el diagrama de secuencia para ese procedimiento se presenta a continuación.

Figura 58: Diagrama de secuencia para modificar licitación



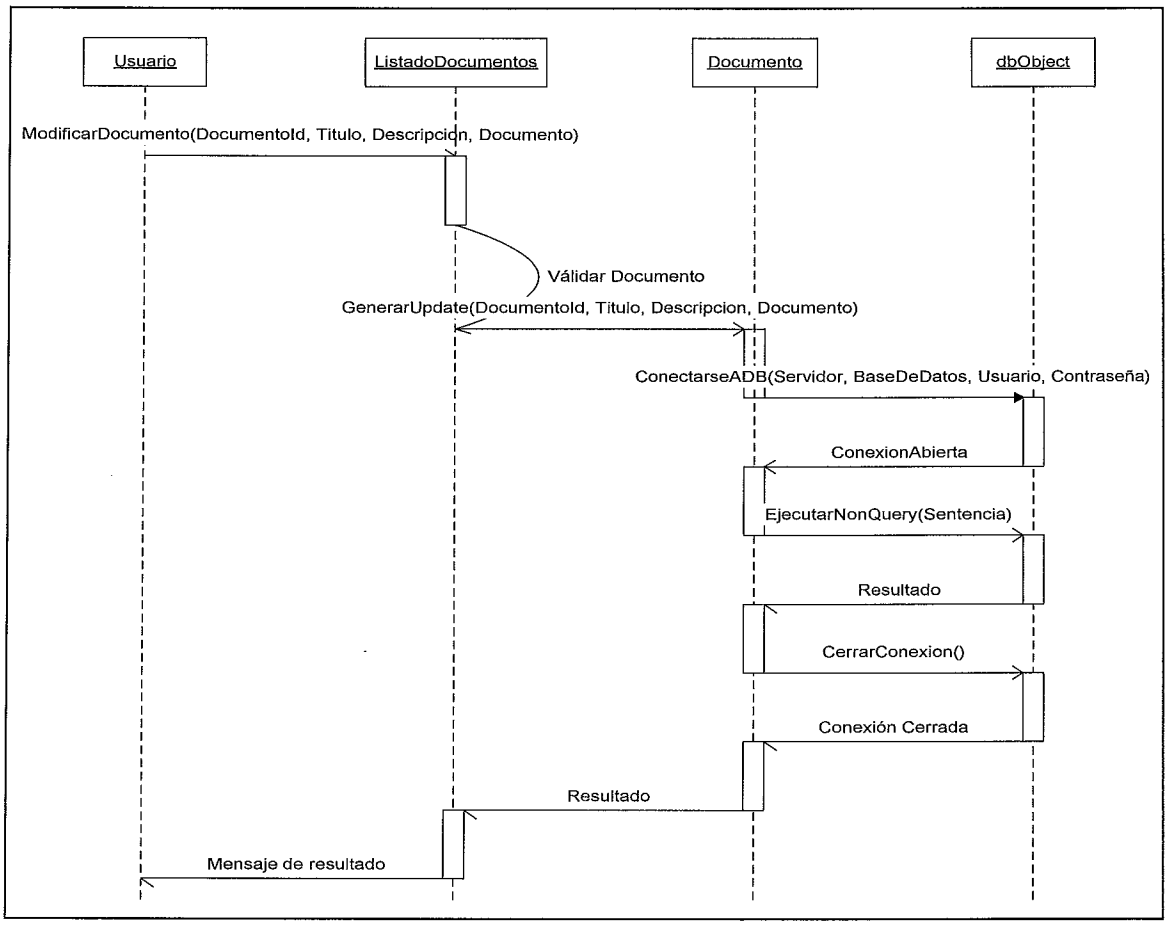
Se pueden modificar todos los datos de una tarea excepto el identificador. En el listado de tareas se selecciona la tarea, se ingresan los nuevos datos y se solicita guardar los cambios. Los datos ingresados son verificados y se solicita al objeto *Tarea* que produzca la sentencia *Update* para actualizar la tarea con los datos que recibe en sus parámetros. Esta sentencia es enviada a la capa de datos para ser ejecutada a través del método EjecutarNonquery. Modificar una tarea puede implicar, modificar sus documentos; el diagrama de secuencia para ese procedimiento se presenta a continuación.

Figura 59: Diagrama de secuencia para modificar tarea



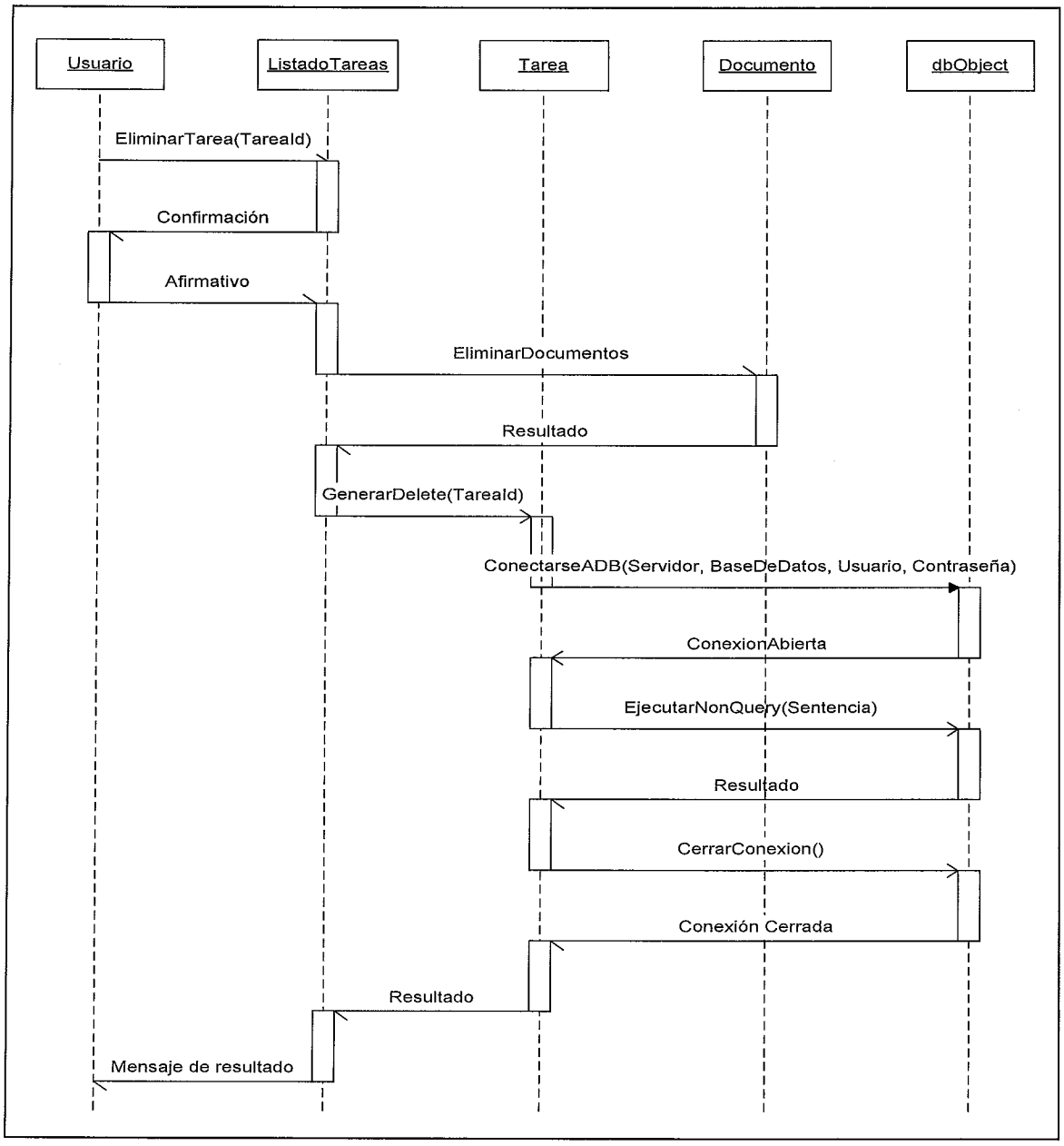
Se pueden modificar todos los datos de un documento excepto el identificador. En el listado de licitaciones se selecciona el documento, se ingresan los nuevos datos y se solicita guardar los cambios. Los datos ingresados son verificados y se solicita al objeto *Documento* que produzca la sentencia *Update* para actualizar el documento con los datos que recibe en sus parámetros. Esta sentencia es enviada a la capa de datos para ser ejecutada a través del método EjecutarNonquery.

Figura 60: Diagrama de secuencia para modificar documento



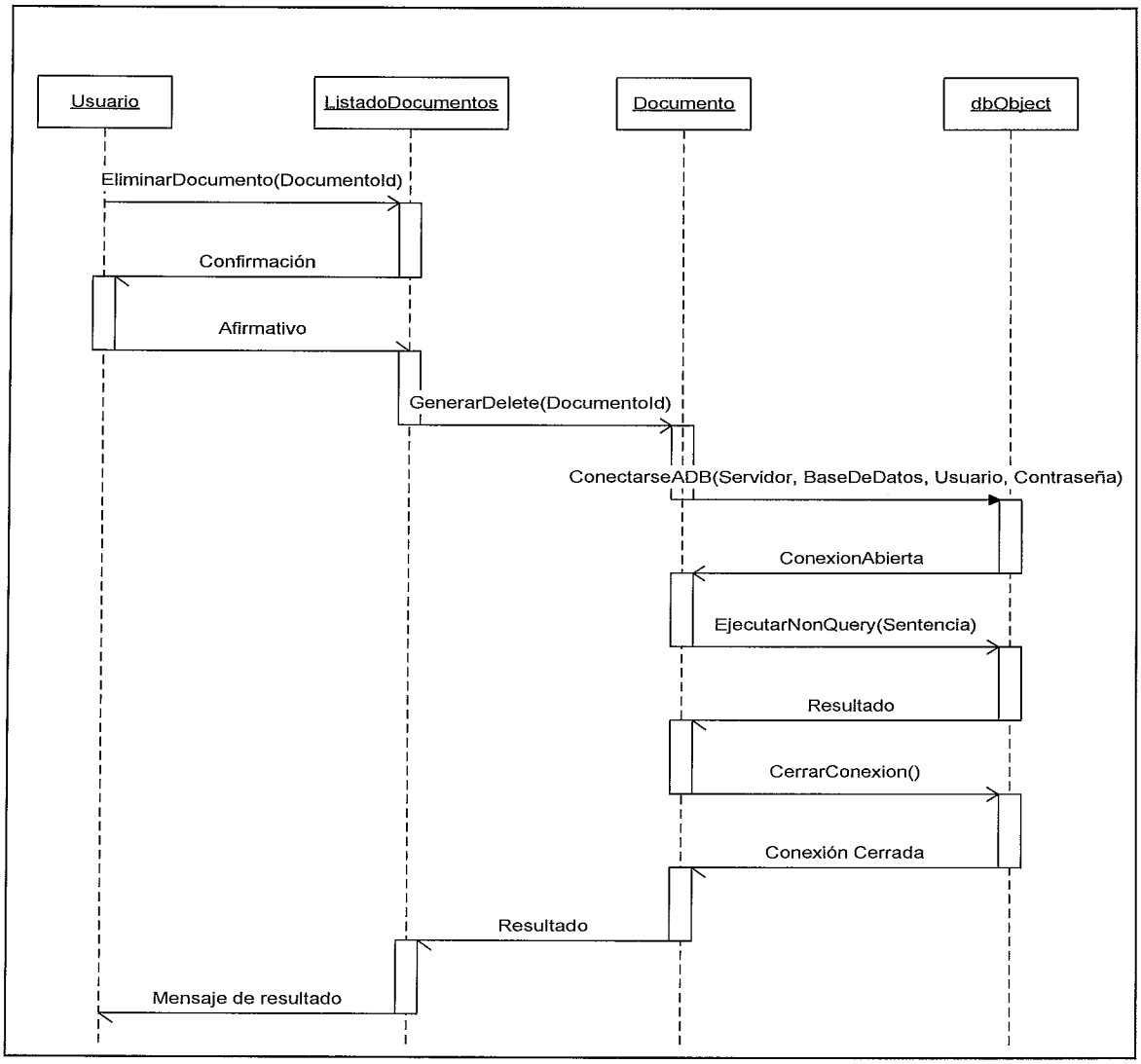
Se puede eliminar una tarea y esto implica eliminar antes sus documentos asociados. En el listado correspondiente se selecciona la tarea que desea eliminarse (se puede eliminar una tarea aún cuando haya sido terminada), se confirma la acción, se eliminan sus documentos y se genera y ejecuta la sentencia *Delete* para eliminar la tarea a través de los objetos *tarea* y *dbObject*.

Figura 61: Diagrama de secuencia para eliminar tarea



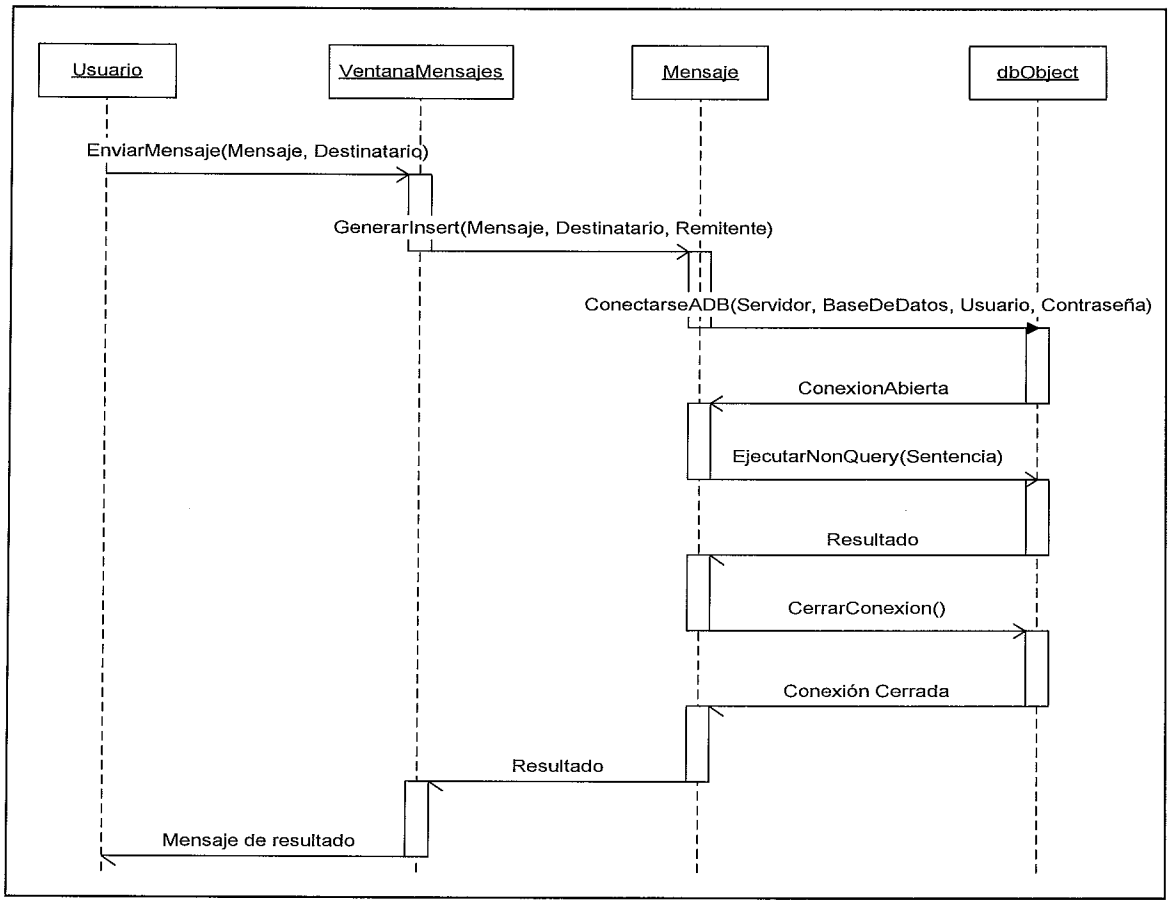
En el listado correspondiente, se selecciona el documento que desea eliminarse, se confirma la acción y se genera y ejecuta la sentencia *Delete* para eliminar el documento a través de los objetos *documento* y *dbObject*.

Figura 62: Diagrama de secuencia para eliminar documento



Para enviar un mensaje, el usuario selecciona en la ventana de mensajes el usuario destinatario, escribe el texto del mensaje y solicita enviarlo. La ventana envía los datos ingresados junto con el nombre del usuario como remitente al objeto *Mensaje* para que genere la sentencia de inserción de los datos en la tabla de mensajes, asignando como estado del mensaje el valor '0' para indicar que el mensaje aún no ha sido leído. La sentencia generada es ejecutada por *dbObject*.

Figura 63: Diagrama de secuencia para enviar mensaje



El usuario podrá seleccionar las siguientes opciones:

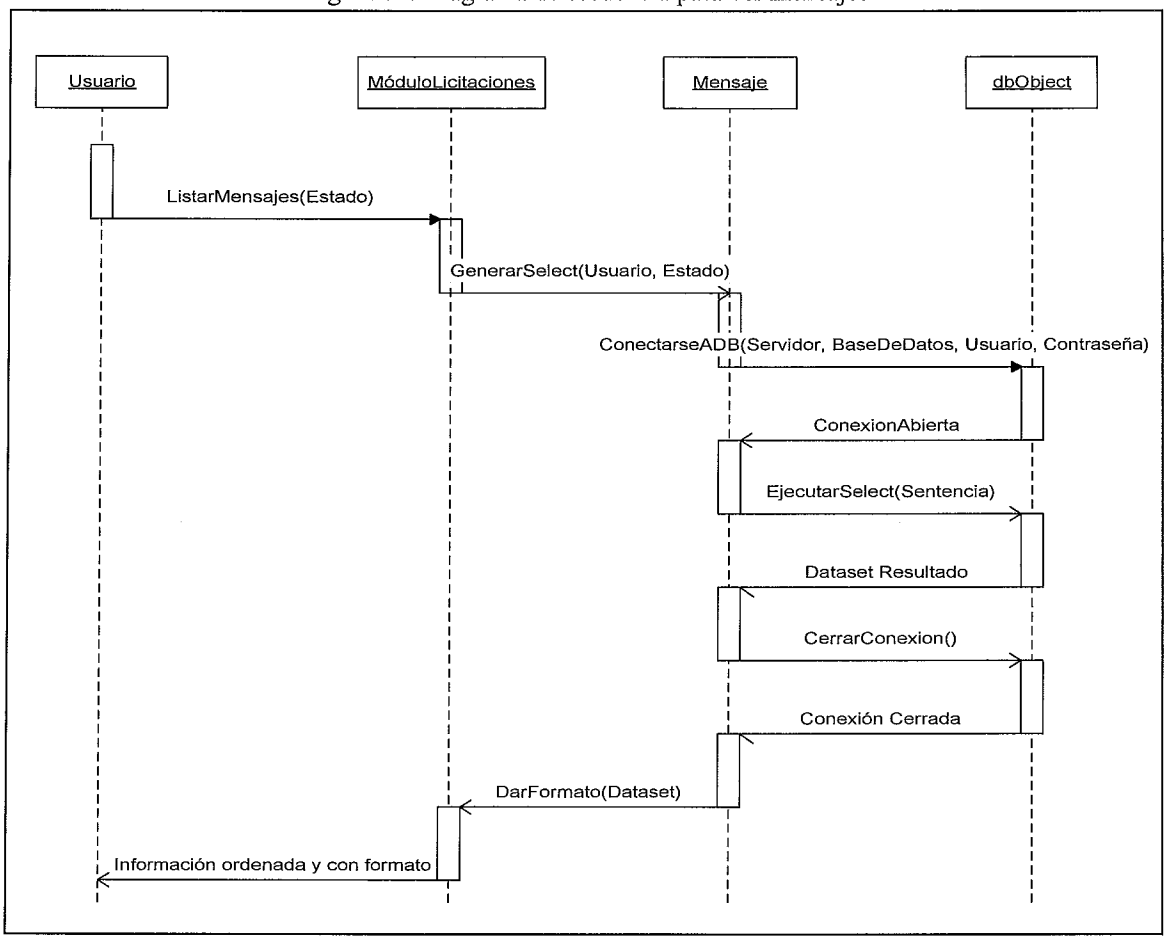
- Ver mensajes en bandeja de entrada.
- Ver mensajes en papelera.

Los mensajes no leídos en bandeja de entrada tienen estado '0', los mensajes leídos en bandeja de entrada tienen estado '1', los mensajes en papelera no leídos tienen estado '3' y los mensajes leídos en papelera tienen estado '4'.

Según la opción seleccionada, la capa de presentación envía el usuario y el estado de los mensajes al método GeneraSelect, de la capa de aplicación, para que genere la sentencia que trae los mensajes con el estado indicado y del usuario indicado. La sentencia es ejecutada por el método EjecutarSelect de *dbObject* y el resultado es devuelto en un dataset para que se presenten los datos al usuario.

Dentro del listado se puede ver cada mensaje y esto implica que se cambie el estado del mensaje a leído.

Figura 64: Diagrama de secuencia para ver mensajes

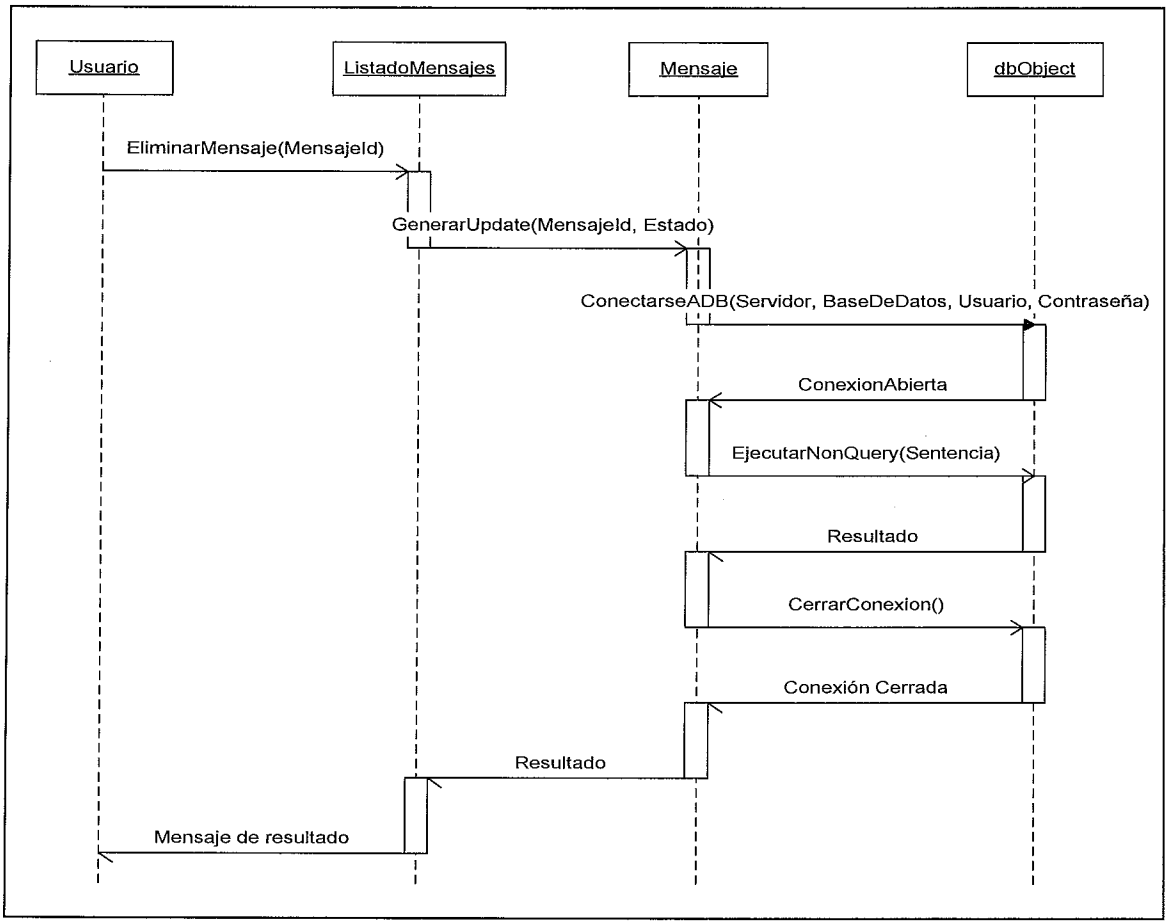


El mensaje puede ser eliminado de la bandeja de entrada o de la papelera, el usuario selecciona el mensaje del listado y solicita eliminarlo.

- Si el mensaje tiene estado '0' se cambia a '2'.
- Si el mensaje tiene estado '1' se cambia a '3'.
- Si el mensaje tiene estado '2' o '3' se cambia a '4' y ya no puede verse en ningún listado.

A la capa de aplicación se envía el identificador del mensaje y el estado al que cambia para que se genere y ejecute el *Update* correspondiente en la tabla de mensajes.

Figura 65: Diagrama de secuencia para eliminar mensaje



5. Especificación de clases. Las clases que participan en el módulo de licitaciones son:

- Clase licitaciones.
- Clase tareas.
- Clase documentos.
- Clase mensajes
- Clase modulolicitaciones.

Las primeras cuatro clases pertenecen a la capa de aplicación y todas poseen como miembro privado un objeto de tipo *dbObjects* para el manejo de la conexión a la base de datos y la ejecución de las sentencias SQL. Sus métodos producen las sentencias *Select, Insert, Update,* y *Delete* según lo indique su nombre, con los parámetros que reciben y solicitan al método apropiado de *dbObject* que ejecute la sentencia, enviada como parámetro, para obtener el resultado y regresarlo a la capa de presentación.

Las clases trabajan sobre las tablas de la base de datos paralela que llevan el mismo nombre que la clase. Por ejemplo Licitaciones trabaja sobre la tabla tbl_Licitaciones.

La clase Licitaciones tiene un miembro privado *tarea* y la clase Tarea tiene un miembro privado *documento*. Lo que implica que desde un objeto de *Licitaciones* se puede tener acceso a sus tareas y desde cualquier tarea a sus documentos.

La clase ModuloLicitaciones pertenece a la capa de presentación y es la que presenta las ventanas de creación de licitaciones, tareas y documentos y la de enviar mensajes. Además, posee los métodos que solicitan los listados de licitaciones, tareas, documentos y mensajes a la capa de aplicación para presentarlos al usuario. Posee también los métodos que solicitan a la capa de aplicación el detalle de una licitación, tarea, documento o mensaje y lo presentan al usuario.

La especificación de cada clase se presenta a continuación.

Figura 66: Clase licitaciones

Licitaciones
-dbObjects dbObject -Tareas tarea
+Licitaciones() +GenerarSelect(in FechaIni: datetime, in FechaFin: datetime, in Estado: int):DataSet +GenerarInsert(in Nombre: string, in descripcion: string, in Responsable: int, in FechaEntrega: datetime, in DocumentoAsociado: binary): int +GenerarUpdate(in LicitacionId: int, in descripcion: string, in Responsable: int, in FechaEntrega: datetime, in DocumentoAsociado: binary, in Resultados: string, in Adjudicacion: int): int + DetalleLicitacion(in LicitacionId: int): DataTable

Figura 67: Clase tareas

Tareas
-dbObjects dbObject -Documentos documento
+Tareas() +GenerarSelect(in FechaIni: datetime, in FechaFin: datetime, in Estado: int, in Responsible: int, in LicitacionId: int):DataSet +GenerarInsert(in descripcion: string, in Responsable: int, in FechaLimite: datetime, in Peso: int, in LicitacionId: int, PorcentajeTerminado:Int): int +GenerarUpdate(in TareaId: int, in descripcion: string in Responsable: int, in FechaLimite: datetime, in Peso: int, PorcentajeTerminado:Int): int +GenerarDelete(TareaId): int + DetalleTarea(in TareaId: int): DataTable

Figura 68: Clase docuementos

Documentos
-dbObjects dbObject
+Documentos() +GenerarSelect(in TareaId: int):DataSet +GenerarInsert(in Titulo: string, in Descripcion: string): int +GenerarUpdate(in DocumentoId: int, in Titulo: string, in Descripcion: string, in Documento: binary): int +GenerarDelete(DocumentoId): int + DetalleDocumento(in DocumentoId: int): DataTable + DescargarDocumento(in DocumentoId: int) + CargarDocumento(in DocumentoId: int)

Figura 69: Clase mensajes

Mensajes
-dbObjects dbObject
+Mensajes() +GenerarSelect(in Usuario: int, in Estado: int):DataSet +GenerarInsert(in Mensaje: string, in Destinatario: int, in Remitente: int): int +GenerarUpdate(in MensajeId: int, in Estado: int): int + DetalleMensaje(in MensajeId: int): DataTable

Figura 70: Clase ModuloLicitaciones

MóduloLicitaciones
-Licitaciones Licitacion -Mensajes Mensaje
+ModuloLicitaciones() +VentanaLicitacion() +VentanaTarea() +VentanaDocumento() +VentanaMensajes() +ListarLicitaciones(in FechaIni: datetime, in FechaFin: datetime, in Estado: int) +DetalleLicitacion(in LicitacionId: int) +ListarTareas(in FechaIni: datetime, in FechaFin: datetime, in Estado: int, in Responsible: int, in LicitacionId: int) +DetalleTarea (in TareaId: int) +ListarDocumentos(in TareaId: int) +DetalleDocumento(in DocumentoId: int) +ListarMensajes(in usuario: int, in Estado: int) + DetalleMensaje(in MensajeId: int): DataTable

I. Abstracción de módulos y operaciones por tipo de usuario

A continuación presento las operaciones que pueden realizarse dentro de un módulo por tipo de usuario.

Cuadro 35: Opciones por tipo de usuario en módulo control de pedidos

Módulo control de pedidos
Usuario gerencia
▪ Listado de pedidos pendientes de liberar o Buscar pedido ▪ Listado de pedidos pendientes de despachar o Buscar pedido ▪ Resumen de despachos
Usuario asesor de ventas
▪ Listado de pedidos pendientes de liberar o Buscar pedido ▪ Listado de pedidos pendientes de despachar o Buscar pedido
Usuario producción
▪ Listado de pedidos pendientes de liberar o Buscar pedido ▪ Listado de pedidos pendientes de despachar o Buscar pedido ▪ Ingresar – Modificar fecha en planta ▪ Ingresar – Modificar fecha de inicio de producción ▪ Ingresar – Modificar fecha de fin de producción ▪ Ingresar – Modificar fecha en patio
Usuario despachos
▪ Listado de pedidos pendientes de liberar o Buscar pedido ▪ Listado de pedidos pendientes de despachar o Buscar pedido ▪ Programar despacho

Cuadro 36: Opciones por tipo de usuario en el módulo control de cotizaciones

Módulo control de licitaciones
Usuario gerencia
▪ Listado de cotizaciones pendientes de aceptar o Cambiar estado de cotización o Imprimir cotización ▪ Listado de cotizaciones aceptadas o Imprimir cotización ▪ Relación de ventas cerradas y cotizaciones perdidas
Usuario asesor de ventas
▪ Listado de cotizaciones pendientes de aceptar o Cambiar estado de cotización o Imprimir cotización ▪ Listado de cotizaciones aceptadas o Imprimir cotización

Cuadro 37: Opciones por usuario en el módulo control de visitas

Módulo control de visitas
Usuario gerencia y asesor de ventas
▪ Programar visita ▪ Reprogramar visita ▪ Cancelar visita ▪ Ingresar resultados de visita ▪ Listado de visitas programadas ▪ Listado de visitas canceladas ▪ Listado de visitas pendiente de programar o Aceptar visita ▪ Listado de visitas realizadas

Cuadro 38: Opciones por usuario en el módulo control de licitaciones

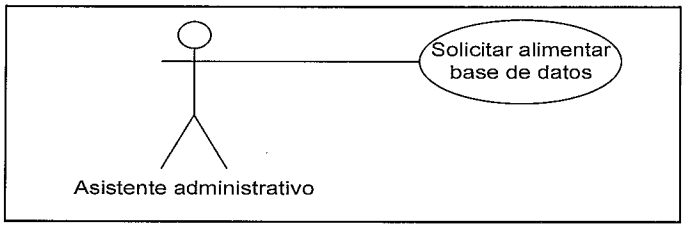
Módulo control de licitaciones
Usuario coordinador
▪ Crear licitación o Adjuntar documento inicial o Crear tareas • Crear lista de documentos asociados ▪ Modificar licitación o Crear tarea nueva o Eliminar tarea o Modificar tarea • Ingresar porcentaje de avance ▪ Listado de licitaciones pendientes de entregar o Descargar documentos adjuntos o Porcentaje de avance de licitaciones y tareas • Enviar recordatorios y alertas ▪ Listado de tareas pendientes de entregar o Adjuntar documento a tarea ▪ Listado de licitaciones terminadas y resultados o Ingresar resultados y adjudicación o Descargar documentos ▪ Ver recordatorios y alertas
Usuario gerencia
▪ Listado de licitaciones pendientes de entregar o Descargar documentos adjuntos o Porcentaje de avance de licitaciones y tareas • Enviar recordatorios y alertas ▪ Listado de tareas pendientes de entregar o Adjuntar documento a tarea ▪ Listado de licitaciones terminadas y resultados o Ingresar resultados y adjudicación o Descargar documentos o Ver recordatorios y alertas
Usuario técnico
▪ Listado de licitaciones pendientes de entregar ▪ Listado de tareas pendientes de entregar o Adjuntar documento a tarea ▪ Listado de licitaciones terminadas y resultados ▪ Ver recordatorios y alertas

J. Alimentador de base de datos secundaria

1. Propósito. Mensualmente se presentan reportes de ventas por producto, por cliente y por vendedor y se calculan márgenes de ganancia mensual y anualmente. Por motivos de confidencialidad no presento imágenes con ejemplos de los datos en los reportes. Estos reportes se generan a partir de una base de datos en Excel, que es alimentada manualmente por el asistente administrativo del departamento. El propósito de esta aplicación es alimentar la base de datos en Excel, a partir de la base de datos del sistema de operaciones en Firebird. La aplicación será una aplicación de Windows y el único usuario será el asistente administrador de ventas.

2. Funcionamiento. La aplicación únicamente posee un caso de uso, éste se presenta en la siguiente figura.

Figura 71: Caso de uso para alimentar base de datos de Excel



En el caso de uso, el actor es el asistente administrativo y es el que solicita que se alimente la base de datos en Excel a partir del rango de fechas que ingresa.

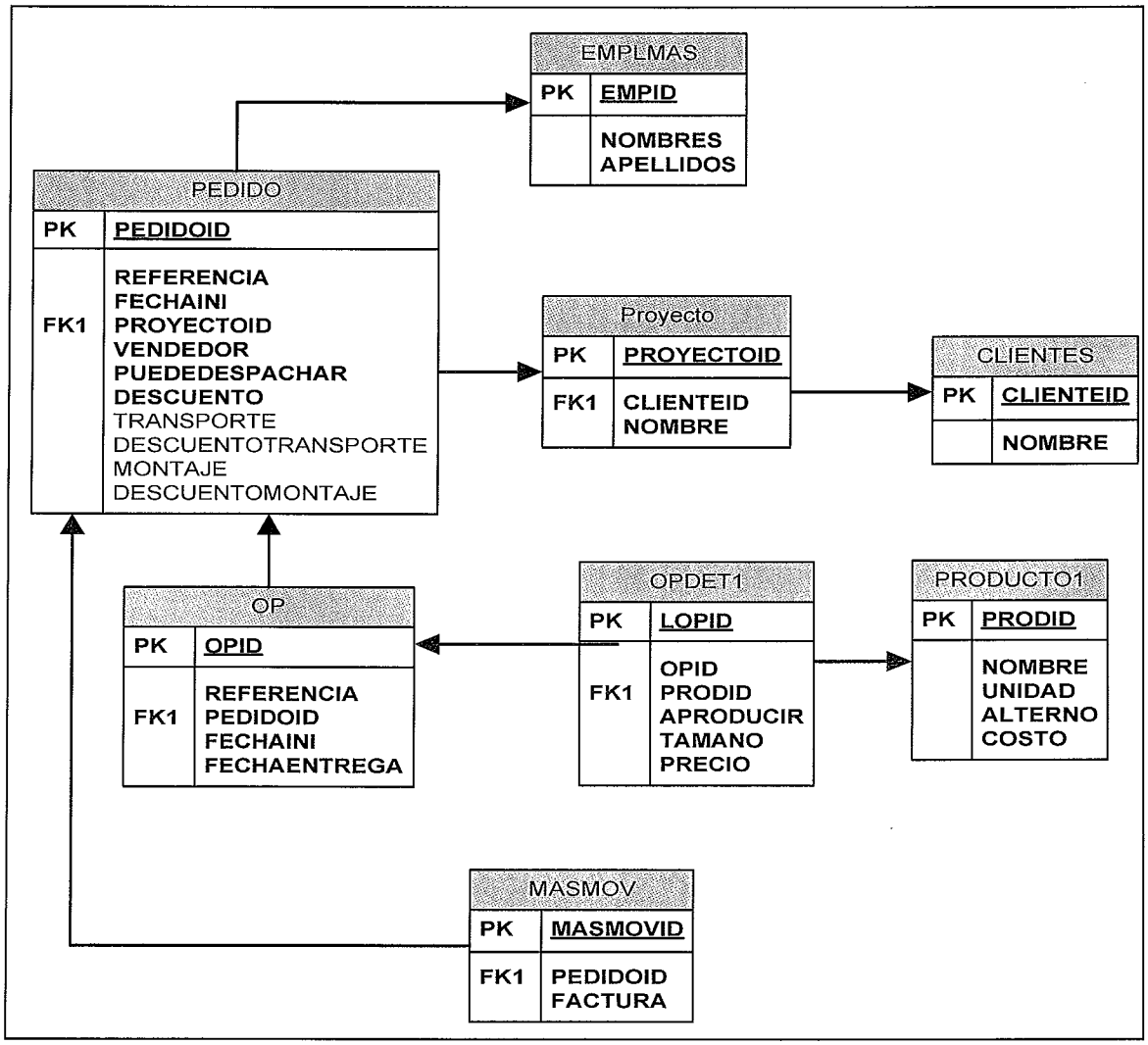
Las columnas que la base de datos de Excel debe tener son:

- Día.
- Mes.
- Año.
- Identificador de pedido.
- Cliente.
- Vendedor.
- Descuento.
- Transporte.
- Montaje.
- Descuento transporte.
- Descuento montaje.
- Producto.
- Cantidad.
- Costo.

- Precio venta.
- Factura.

Estas columnas se obtienen de las tablas y campos de la base de datos actual que se muestran en la siguiente figura:

Figura 72: Tablas y campos para alimentar las columnas de la base de datos en Excel



El campo FECHAINI del pedido debe estar dentro del rango de fechas ingresado por el asistente y PUEDEDESPACHAR debe ser '1'.

Figura 73: Diagrama de secuencia para alimentar base de datos secundaria

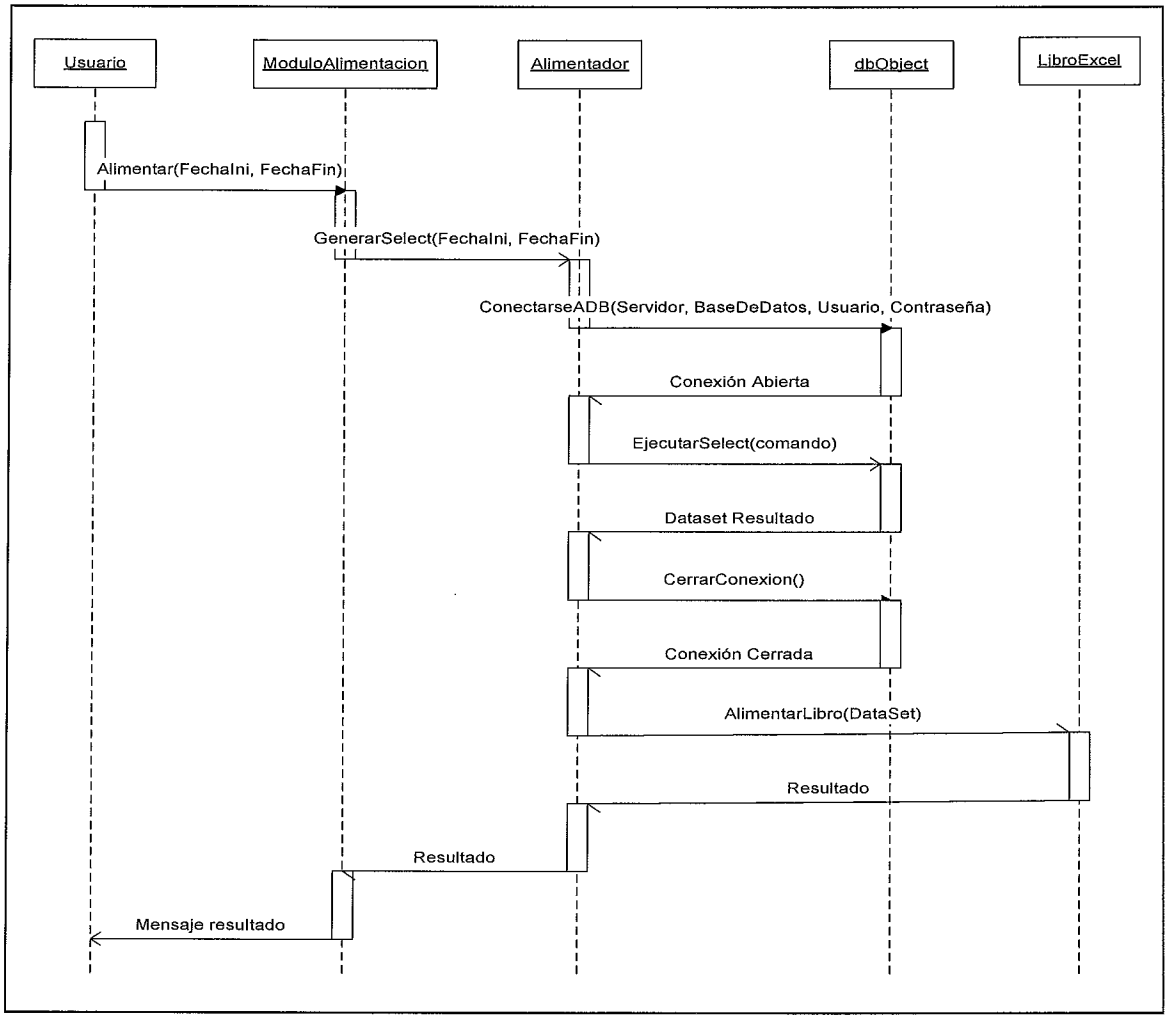


Figura 74: Clase alimentador

Alimentador
-dbObjects dbObject
+GenerarSelect(in FechaIni: int, in FechaFin: int):DataSet +AlimentarLibro(in datos: DataSet): int

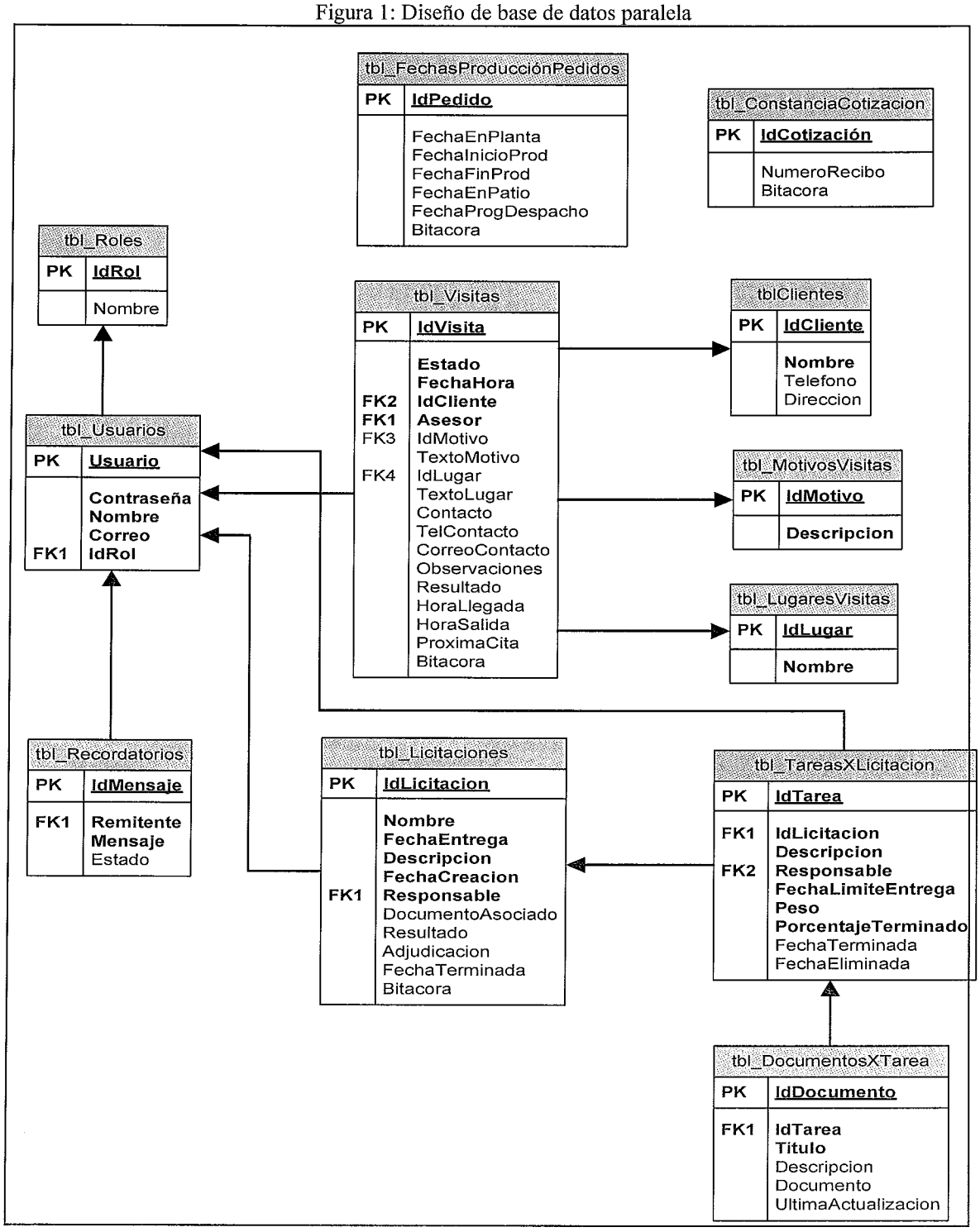
El usuario ingresa en la ventana de la aplicación, la fecha de inicio y fin para indicar el rango de fechas en el que deben encontrarse los pedidos. La capa de presentación envía estos datos al método GeneraSelect, del objeto *Alimentador,* que genera la sentencia *Select* para obtener los datos de la vista en la figura 72, que cumplen las condiciones de fecha indicadas por los parámetros FechaIni y FechaFin. *GeneraSelect* envía la sentencia a *dbObject* para que a través de EjecutarSelect devuelva el resultado de la ejecución en un dataset. El dataset es enviado al método AlimentarLibro para que escriba cada fila del dataset en una

hoja de un libro de Excel. En el apéndice E se encuentra un ejemplo del código para exportar un dataset a un libro de Excel.

XIII. DISEÑO DE LA BASE DE DATOS PARALELA Y REPLICACIÓN

A. Diseño de la base de datos

Figura 1: Diseño de base de datos paralela



1. Triggers

- En el *update* de una visita se ejecuta un trigger que inserta en la bitácora, en la misma tabla, la fecha y hora del cambio, los datos guardados y el usuario que lo realizó.

- En el *update* de una licitación se ejecuta un trigger que inserta en la bitácora, de la misma tabla, la fecha y hora del cambio, los datos guardados y el usuario que lo realizó.

B. Replicación

En la base de datos paralela la tabla *tbl_FechasProduccionPedidos* debe contener los mismos pedidos que la tabla *PEDIDO* de la base de datos actual. La tabla *tbl_ConstanciaCotización* debe tener las mismas cotizaciones que la tabla *COTIZA* de la base de datos actual. La tabla *tbl_Clientes* debe tener los mismos clientes que la base de datos CLIENTES del sistema actual. Esto significa que estas tres tablas deben mantenerse sincronizadas con las tablas de la base de datos actual.

La solución a este problema es que se repliquen los campos de estas tablas desde la base actual hacía la base paralela. Debido a que Firebird no cuenta con un motor para replicación. La replicación se realizará utilizando la herramienta *Interbase replication suite.*

El *Interbase replication suite* trabaja en base a triggers que son activados y ejecutados cuando se realizan cambios sobre las tablas especificadas. Estos triggers almacenan en la bitácora, los cambios realizados y luego son replicados a la base de datos destino. Lo que significa que, únicamente los cambios son transferidos, siguiendo siempre la misma secuencia que en la base de datos en la que se realizaron los cambios. La bitácora de replicación se elimina luego que se realiza la replicación.

La herramienta permite configurar para qué campos se deben guardar los cambios en la bitácora y en este caso serán los siguientes campos:

- PEDIDOID: de la tabla PEDIDO
- COTIZAID: de la tabla COTIZA
- CLIENTEID: de la tabla CLIENTES
- NOMBRE: de la tabla CLIENTES
- TELEFONO1: de la tabla CLIENTES
- DIRECCION: de la tabla CLIENTES

Debido a que los primeros tres campos son llave primaria en las tablas fuente, los únicos cambios que existirán siempre en la bitácora serán las inserciones.

Luego de que *replicator* se conecta a la base de datos fuente, busca los cambios en la bitácora y si hay registros, se conecta a la base de datos destino (en este caso la base de datos paralela en el mismo servidor) y transfiere los cambios. Debido a que replicator se conectará a las dos bases de datos simultáneamente la replicación se denomina en línea.

En el apéndice F adjunto el manual para la instalación y configuración de *Interbase replication suite.*

XIV. CONCLUSIONES Y RECOMENDACIONES

1. Para maximizar la utilidad de la información, una empresa debe manejarla correctamente tal como maneja los demás recursos. Los líderes deben comprender que hay costos asociados con la producción, distribución, seguridad, almacenamiento y recuperación de toda la información. En la empresa Precon es necesario realizar un análisis sobre el manejo de la información en los demás departamentos para determinar las necesidades y los procesos que deben abarcarse, e iniciar una lenta modernización y reestructuración en la administración de este recurso.

2. Se determinó que el sistema actual únicamente realiza el almacenamiento de datos relacionados a las cotizaciones y pedidos y emite documentos como órdenes de producción, órdenes de despacho, envíos y facturas. Sin embargo, los datos almacenados no son utilizados para informar a los empleados sobre el estado de los pedidos de sus clientes o para informar al gerente sobre los tiempos de despacho, las cotizaciones ofrecidas y rechazadas por un vendedor y los montos perdidos en cotizaciones rechazadas, entre otros. La información es almacenada en la base de datos a través de las operaciones realizadas en los módulos *Producc, Ventas, y Despachos* del sistema, utilizados por la secretaría y el asistente administrativo del departamento. Ningún otro empleado del departamento utiliza el sistema.

3. La base de datos actual cuenta con doscientas noventa y cinco tablas, de éstas las que almacenan información relacionada con el departamento de ventas son:

    a. CLIENTES
    b. CACC0TCC
    c. PROYECTO
    d. PRODUCTO
    e. GRUPO
    f. SUBGRUPO
    g. PEDIDO
    h. OP
    i. OPDET
    j. COTIZA
    k. COTIZADET
    l. CAOD0DES
    m. CAOD0DETDES
    n. MASMOV y
    o. TIPMOV

A pesar de que no todos los campos de las tablas son utilizados, los datos indispensables pueden ser recuperados y complementados para presentar información útil a los asesores y al gerente de ventas sobre todo.

4. En la solución propuesta, se presenta el análisis y diseño de cinco aplicaciones que beneficiarían en gran manera al departamento de ventas de la empresa. Las aplicaciones tienen como objetivo informar a los involucrados en el proceso de ventas sobre el estado de pedidos y cotizaciones; abarcar procesos que no se tomaron en cuenta en el sistema actual, como el control de visitas y de licitaciones con sus respectivas tareas; y alimentar la base de datos utilizada para generar los reportes de ventas mensuales, evitando realizar el mismo trabajo dos veces. El diseño y análisis propuesto presenta una arquitectura de tres capas, la capa de datos, aplicación y presentación, es importante que las aplicaciones se desarrollen siguiendo esta arquitectura, logrando una independencia entre las capas, con el fin de que futuros cambios a una de las capas no afecten a las demás. Por ejemplo, podría migrarse la base de datos a un servidor SQL y esto sólo debería ocasionar cambios en la capa de datos.

5. Dentro de las aplicaciones los usuarios poseerán un *Rol* igual al puesto que desempeñan en la empresa. Según el Rol, se determina qué módulos puede usar y las operaciones que el usuario puede realizar dentro del módulo. El identificador del usuario dentro del sistema será el mismo código de empleado en la empresa y la contraseña debe ser única y confidencial. El usuario deberá ser autenticado para entrar al sistema, en caso contrario no podrá utilizar ningún módulo del mismo.

6. La plataforma de desarrollo, el servidor de base de datos y las herramientas recomendadas en la presente solución se definieron en base a los recursos que la empresa posee a la fecha, con el fin de minimizar los costos, pero siempre pensando en el mejor rendimiento y la modernización de los sistemas. Por ello se recomienda utilizar Microsoft Visual Studio .NET como plataforma de desarrollo y Firebird como servidor de base de datos, utilizando Firebird ADO.NET Data Provider para la comunicación entre las aplicaciones en .NET y las bases de datos en el servidor de Firebird.

7. Además de implementar las aplicaciones propuestas, es importante informar a los miembros del departamento sobre su existencia y funcionamiento y capacitar a los usuarios para obtener el máximo beneficio de las aplicaciones. De nada servirá implementarlas si nadie las utiliza y si no se crea la cultura apropiada para administrar el recurso de información entre los empleados del departamento.

8. Para futuras inversiones en sistemas de información, se recomienda a la empresa considerar utilizar sistemas estándar probados y verificados. En caso de que se desee comprar un sistema a la medida se recomienda comprar el sistema con su código fuente y permisos para cambios en la base de datos y en el sistema, con el fin de evitar relaciones forzadas con alguna empresa. Además se recomienda validar

un análisis y diseño previo a la implementación del sistema para evitar comprar un sistema que no cumpla con los requisitos mínimos para la correcta gestión de la información.

XV. BIBLIOGRAFÍA

Documentación SDK de Firebird ADO.NET Provider v1.7

Fraude, Eric. 2003. *Ingeniería de Software*. México D.F., Alfaomega. 539

Kendall,Edward; J. Kendall. 1997. *Análisis y Diseño de Sistemas* 3ª ed. México,D.F., Prentice-Hall Inc. 913 págs.

Rumbaugh, James et al. 1996. *Modelado y diseño orientados a objetos*. Madrid, Prentice Hall, 1996.

Yourdon, Edward. 1989. *Modern Structured Análisis*. E.U.A.., Prentice-Hall Inc. 672 págs.

XVI. REFERENCIAS DE INTERNET

Bhagwat, A. *Architecture and Design: Unified Modeling Language (UML)*. 2003
http://www.cetus-links.org/oo_uml.html

Castro, D., Pérez, R. *Replicación de datos en SQL Server*
http://www.monografias.com/trabajos15/replicacion-datos/replicacion-datos.shtml

Concepción, P. *Análisis y diseño de sistemas*. 2003
http://www.ilustrados.com/publicaciones/EpypppIVpEbjHkcuHD.php

DAEDALUS. *Sistemas Complejos*. 2005
http://www.daedalus.es/AreasISAnalisis-E.php

Firebird. http://www.firebirdsql.org/index.php

Gándara, J. *Desarrollo de aplicaciones con tecnología Internet*. 2005.
http://www.soluziona.es/htdocs/areas/consultoria/interes/articulos/tecnologia_internet.shtml

Glossary. http://members.tripod.com/~rvbelzen/c128sg/glossary.htm

Gracia, J. *Diseño de Software Orientado a Objetos*. 2003
http://www.ingenierosoftware.com/analisisydiseno/uml.php

Grupo Precon. http://www.precon.com.gt/index1.htm

Guzmán, C. *About the Firebird ADO.NET Data Provider*. 2005
http://firebird.sourceforge.net/index.php?op=devel&sub=netprovider

Letelier, P. *Desarrollo de Software Orientado a Objetos Usando UML*. UPV España.
http://www.dsic.upv.es/~uml/

Lycos, Inc. Glosario. 2003 http://webmaster.lycos.es/glossary/

Peralta, M. *Sistema de Información*. 1997
http://www.monografias.com/trabajos7/sisinf/sisinf.shtml#intro

Replication Suite Description. 2002 http://www.2p.cz/en/interbase_replicator/index.html

Jiménez, Alexandra; M. Leal, C. González. *Qué es UML?* http://www.creangel.com/uml/home.php

Sánchez, Susana; J. Rodríguez. *La Información Como Recurso en el Desarrollo de las Organizaciones de las Administraciones Públicas*. 2000 http://www.um.es/fccd/anales/ad03/AD10-2000.PDF

Sparks, G. *Una Introducción al UML*. El Modelo de Casos de Uso. http://www.sparxsystems.cl/WhitePapers/El_Modelo_de_Casos_de_Uso.pdf

SQL Manager .NET. EMS IB/FB Manager On-Line documentation. http://www.sqlmanager.net/products/ibfb/manager/documentation/hs190.html

Universidad Nacional de Colombia. *Datos e Información*. 2005 http://www.virtual.unal.edu.co/cursos/economicas/2006882/capitulos_/capitulo1/cap13.html

Yank, K. *ASP Language Basis*. 2001http://www.sitepoint.com/article/asp-language-basics

XVII. APÉNDICES

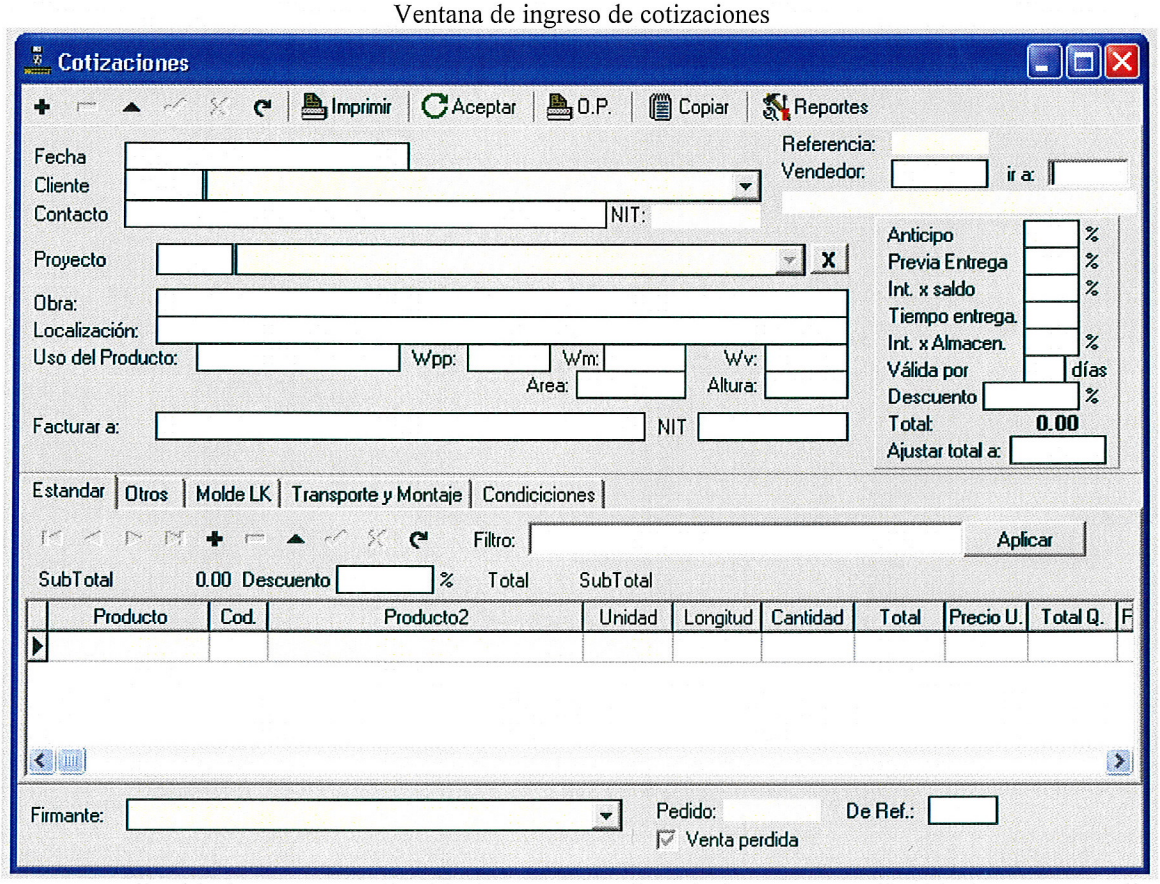
Apéndice A

Ejemplos de los tres documentos utilizados en el proceso de venta de la empresa Precon: Cotización, Orden de Producción y Orden de Despacho respectivamente.

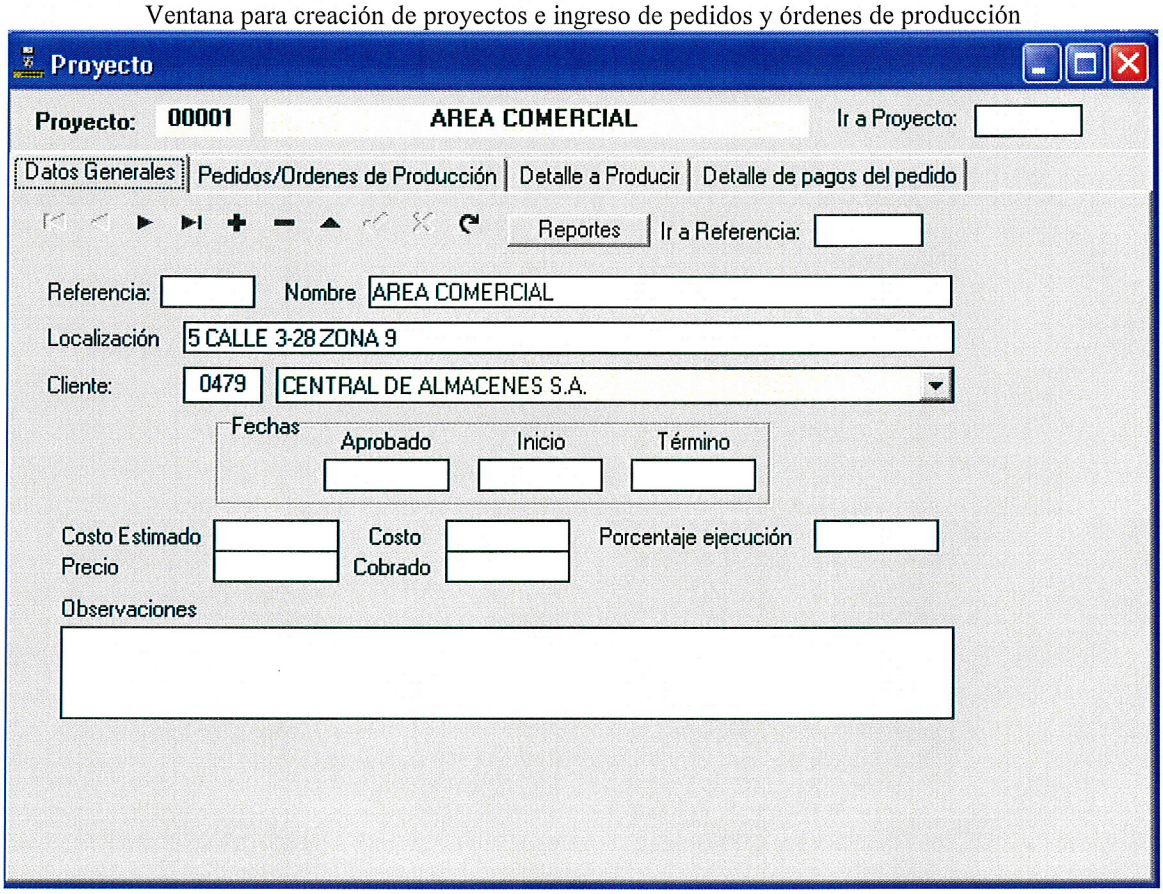
Apéndice B

A continuación se presentan imágenes de las ventanas utilizadas para el ingreso de cotizaciones, pedidos, órdenes de producción y órdenes de despacho.

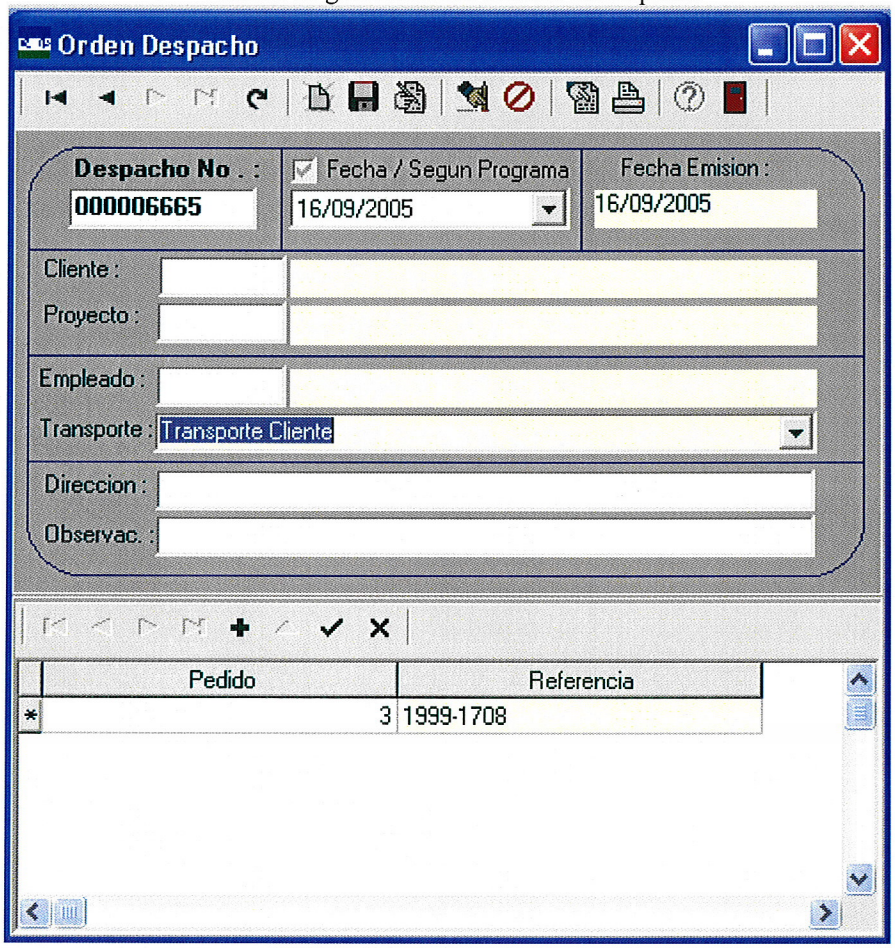
Ventana de ingreso de cotizaciones



Ventana para creación de proyectos e ingreso de pedidos y órdenes de producción



Ventana de generación de órdenes de despacho



Apéndice C

La información proveída en el portal *msdn* sobre autenticación en ASP.Net se puede encontrar en la página:

http://msdn.microsoft.com/library/spa/default.asp?url=/library/SPA/cpguide/html/cpconaspnetauthentication.asp

Y se presenta a continuación:

Autenticación de ASP.NET

La autenticación es un proceso que consiste en obtener las credenciales de identificación, como nombre y contraseña, de un usuario y validarlas consultando a una autoridad determinada. Si las credenciales son válidas, se considera a la entidad que ha enviado las credenciales como una entidad autenticada. Una vez autenticada la identidad, el proceso de autorización determina si esa identidad tiene acceso a un recurso específico.

ASP.NET implementa este proceso a través de proveedores de autenticación, que son módulos que contienen el código necesario para autenticar las credenciales del solicitante. ASP.NET admite los proveedores de autenticación que aparecen en la tabla siguiente.

Proveedor de autenticación de ASP.NET	Descripción
Autenticación mediante formularios	Sistema que redirige las solicitudes no autenticadas a un formulario HTML mediante la redirección del cliente HTTP. El usuario proporciona las credenciales y envía el formulario. Si la aplicación autentica la solicitud, el sistema emite una cookie que contiene las credenciales o una clave para readquirir la identidad. Las solicitudes posteriores se emiten con la cookie en los encabezados de la solicitud; se autentican y se autorizan en un controlador de eventos ASP.NET mediante el método de validación que especifique el programador de la aplicación.
Autenticación mediante Passport	Servicio de autenticación centralizado proporcionado por Microsoft que ofrece a los sitios Web suscritos servicios de perfil básico y un inicio de sesión único.
Autenticación de Windows	ASP.NET utiliza la autenticación de Windows junto con la autenticación de Servicios de Microsoft Internet Information Server (IIS). ISS realiza la autenticación en una de estas tres formas: básica, implícita o integrada de Windows. Cuando IIS finaliza este proceso, ASP.NET utiliza la identidad autenticada para autorizar el acceso.

Para habilitar un proveedor de autenticación de una aplicación ASP.NET, sólo es necesario crear una entrada para el archivo de configuración de la aplicación de la forma siguiente:

```
// Web.config file
```

<authentication mode= "[Windows|Forms|Passport|None]"/>

Se establece el modo en uno de los modos de autenticación: Windows, Forms, Passport o None. El valor predeterminado es Windows. Si el modo está establecido en None, ASP.NET no aplicará ninguna autenticación adicional a la solicitud; esto puede resultar útil cuando se desea implementar un esquema de autenticación personalizado, o bien cuando sólo se utiliza la autenticación anónima y se desea obtener el mayor nivel de rendimiento posible.

En el presente trabajo se propuso utilizar autenticación de formularios y la información proveída en el portal de msdn sobre el tema encuentra en la página:

http://msdn.microsoft.com/library/spa/default.asp?url=/library/SPA/cpguide/html/cpconaspnetauthentication.asp

Y se presenta a continuación.

Proveedor de autenticación mediante formularios

Normalmente, la autenticación mediante formularios hace referencia a un sistema que redirige las solicitudes no autenticadas a un formulario HTML mediante la redirección del cliente HTTP. La autenticación mediante formularios es la elección correcta si la aplicación necesita recopilar sus propias credenciales del usuario en el inicio de sesión a través de formularios HTML. El usuario proporciona las credenciales y envía el formulario. Si la aplicación autentica la solicitud, el sistema emite una cookie que contiene las credenciales o una clave para readquirir la identidad. Las solicitudes posteriores se emiten con la cookie en los encabezados de la solicitud. Las solicitudes se autentican y autorizan mediante un controlador de eventos ASP.NET que utiliza el método de validación especificado en la aplicación.

La autenticación mediante formularios se utiliza a menudo para personalizar, cuando el contenido se ha personalizado para un usuario conocido. En algunos casos, el problema está en la identificación y no en la autenticación, por lo que basta con almacenar el nombre de usuario en una cookie duradera y utilizar ésta para tener acceso a la información de personalización del usuario.

Utilizar la clase FormsAuthenticationModule

La clase FormsAuthenticationModule expone servicios de autenticación basados en cookies a las aplicaciones ASP.NET.

Se debe proporcionar una dirección URL de inicio de sesión que recopile y autentique las credenciales. Si las credenciales son válidas, se puede confiar en las utilidades auxiliares suministradas para redirigir la solicitud al recurso solicitado originalmente con un vale de autenticación adecuado. De forma alternativa y si no desea redirigir la solicitud, puede simplemente obtener la cookie y establecer sus valores.

En el caso más sencillo, puede configurar una dirección URL de inicio de sesión para redirigir las solicitudes no autenticadas a una página, suministrar una implementación mínima de ese archivo personalizado desde una página de ejemplo, y suministrar pares de credenciales válidos, en un archivo Web.config o en otro archivo. En el siguiente ejemplo de código se muestra cómo se puede controlar esto en un archivo de configuración de ASP.NET. Las contraseñas están codificadas.

```
// Web.config file
<authentication mode="Forms">
  <forms name="SavingsPlan" loginUrl="/logon.aspx">
    <credentials passwordFormat="SHA1">
      <user name="Kim"
```

```
        password="07B7F3EE06F278DB966BE960E7CBBD103DF30CA6"/>
    <user name="John"
        password="BA56E5E0366D003E98EA1C7F04ABF8FCB3753889"/>
  </credentials>
 </forms>
</authentication>
```

FormsAuthenticationModule se configura mediante el elemento <forms> de un archivo de configuración Machine.config o Web.config.

Se puede leer mediante programación la identidad del usuario autenticado mediante formularios, tal y como se muestra en el siguiente ejemplo.

```
[Visual Basic]
Dim authUser As String = Request.ServerVariables("AUTH_USER")
Dim authUser2 As String = User.Identity.Name
[C#]
String authUser = Request.ServerVariables["AUTH_USER"];
String authUser2 = User.Identity.Name;
```

Apéndice D

En la página:

http://msdn.microsoft.com/library/spa/default.asp?url=/library/SPA/cpguide/html/cpconoverviewofadonet.asp

Se encuentra la siguiente información sobre DataTables:

DataTable

Un objeto DataTable, que representa una tabla de datos relacionales en la memoria, se puede crear y usar de manera independiente o lo pueden usar otros objetos de .NET Framework, normalmente como miembro de un objeto DataSet.

Un objeto DataTable se puede crear con el constructor de objetos DataTable o pasando argumentos de constructor al método Add de la propiedad Tables del DataSet, que es una DataTableCollection.

También se pueden crear objetos DataTable dentro de un DataSet con los métodos Fill o FillSchema del objeto DataAdapter, o desde un esquema XML predefinido o inferido, con los métodos ReadXml, ReadXmlSchema o InferXmlSchema del DataSet. Tenga en cuenta que una vez que se ha agregado DataTable como miembro de la colección Tables de un DataSet, no se puede agregar a la colección de tablas de ningún otro DataSet.

La primera vez que se crea un DataTable, no tiene esquema (estructura). Para definir el esquema de la tabla, es necesario crear objetos DataColumn y agregarlos a la colección Columns de la tabla. También se puede definir una columna de claves principales para la tabla y crear objetos Constraint y agregarlos a la colección Constraints de la tabla. Una vez que se ha definido el esquema de DataTable, se pueden agregar filas de datos a la tabla, agregando objetos DataRow a su colección Rows.

No es necesario proporcionar un valor para la propiedad TableName cuando se crea una DataTable: dicha propiedad se puede especificar en otro momento, o se puede dejar vacía. Sin embargo, cuando se agrega una tabla sin valor TableName a un DataSet, la tabla recibirá un nombre predeterminado incremental con el formato Table*N* y comenzando con "Table" para Table0.

Nota Se recomienda evitar la convención de nomenclatura "Table" o "Table*N*" si se proporciona un valor para TableName, ya que el nombre que se proporcione podría entrar en conflicto con un nombre de tabla predeterminado existente en el DataSet. Si el nombre proporcionado ya existe, se iniciará una excepción.

En el ejemplo siguiente se crea una instancia de un objeto DataTable, a la que se asigna el nombre "Customers".

[Visual Basic]

```
Dim workTable as DataTable = New DataTable("Customers")
```

[C#]

```
DataTable workTable = new DataTable("Customers");
```

En el siguiente ejemplo se crea una instancia de una DataTable agregándola a la colección Tables de un DataSet.

[Visual Basic]

```
Dim custDS As DataSet = New DataSet
Dim custTable As DataTable = custDS.Tables.Add("CustTable")
```

[C#]

```
DataSet custDS = new DataSet();
```

```
DataTable custTable = custDS.Tables.Add("CustTable");
```

Después de crear una DataTable en un DataSet, se pueden realizar las mismas actividades que al usar una tabla de una base de datos. Se puede agregar, ver, modificar y eliminar datos en la tabla, se puede supervisar los errores y eventos y se puede consultar los datos de la tabla. Al modificar los datos de una DataTable, también se puede comprobar si los cambios son precisos y determinar si aceptarlos o rechazarlos mediante programación.

La información proporcionada en el portal de msdn sobre el dataset de ADO.NET obtenida de la página

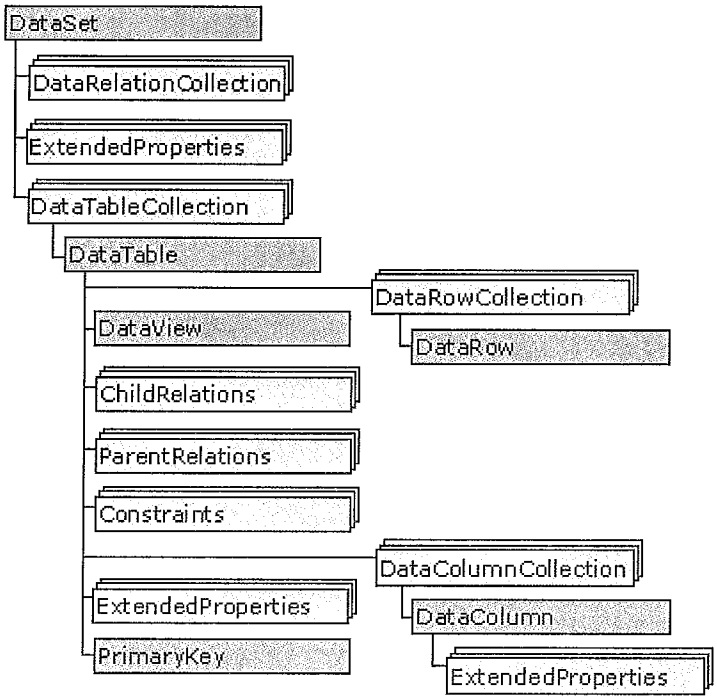
http://msdn.microsoft.com/library/spa/default.asp?url=/library/SPA/cpguide/html/cpconoverviewofadonet.asp

Es presentada a continuación:

DataSet de Ado.net

El objeto DataSet es esencial para admitir escenarios de datos distribuidos de ADO.NET sin mantener una conexión. El DataSet es una representación residente en memoria de datos que proporciona un modelo de programación relacional coherente independientemente del origen de datos. Se puede utilizar con múltiples y distintos orígenes de datos, con datos XML o para administrar datos locales de la aplicación. El DataSet representa un conjunto completo de datos entre los que se incluyen tablas relacionadas, restricciones y relaciones entre las tablas. En la siguiente ilustración se muestra el modelo de objeto DataSet.

Modelo de objeto DataSet



Los métodos y objetos contenidos en un DataSet son coherentes con los del modelo de base de datos relacional.

El DataSet también puede persistir y volver a cargar su contenido como XML y su esquema como esquema XSD (Lenguaje de definición de esquemas XML).

La DataTableCollection

Un DataSet de ADO.NET contiene una colección de cero o más tablas representadas por objetos DataTable. La DataTableCollection contiene todos los objetos DataTable de un DataSet.

Un DataTable se define en el espacio de nombres System.Data y representa una única tabla de datos residentes en memoria. Contiene una colección de columnas representadas por una DataColumnCollection y restricciones representadas por una ConstraintCollection que, juntas, definen el esquema de la tabla. Un DataTable también contiene una colección de filas representadas por la DataRowCollection, que contiene los datos de la tabla. Junto con su estado actual, un DataRow conserva tanto la versión original como la actual para identificar los cambios realizados en los valores almacenados en la fila.

La DataRelationCollection

Un DataSet contiene relaciones en su objeto DataRelationCollection. Una relación, representada por el objeto DataRelation, asocia las filas de un DataTable con las filas de otro DataTable. Es análogo a una ruta de unión que podría existir entre las columnas de claves externas y principales en una base de datos relacional. Un DataRelation identifica columnas coincidentes en dos tablas de un DataSet.

Las relaciones permiten pasar de una tabla a otra dentro de un mismo DataSet. Los elementos esenciales de un DataRelation son el nombre de la relación, el nombre de las tablas relacionadas y las columnas relacionadas de cada tabla. Se pueden establecer relaciones con más de una columna por tabla, para lo que debe especificar una selección de objetos DataColumn como columnas clave. Cuando se agrega una relación al DataRelationCollection, se puede agregar también un UniqueKeyConstraint y un ForeignKeyConstraint para imponer restricciones de integridad cuando se realicen cambios en los valores de las columnas relacionadas.

Apéndice E

A continuación presento el ejemplo de código en Visual Basic .NET para exportar un dataset a un archivo de Excel. El código fue obtenido de la página con dirección:

http://www.dotnetspider.com/technology/kbpages/950.aspx

```
'Now create a windows appliaction in vb.net
'Now in the COM Reference section refer to the MIcrosoft Excel 10.0 Object library.
'Now in the code view state of the form paste the following code

Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles My-
Base.Load
    fncExcelExport("select emp_id,fname from pubs..employee")
End Sub

Private Function fncExcelExport(ByVal strSql As String)
    Dim dsExcelExport As New System.Data.DataSet
    Dim daExcelExport As System.Data.SqlClient.SqlDataAdapter
    Dim Excel As New Excel.Application
    Dim intColumn, intRow, intColumnValue As Integer
    Dim strExcelFile As String
    Dim strFileName As String
    Dim conCurrent As New SqlClient.SqlConnection
    conCurrent.ConnectionString = "data source=;initial catalog=pubs;User ID=sa;Password="
    daExcelExport = New System.Data.SqlClient.SqlDataAdapter(strSql, conCurrent)
    daExcelExport.Fill(dsExcelExport)
    Dim strAppPath = System.Reflection.Assembly.GetExecutingAssembly.Location.Substring(0, Sys-
                     tem.Reflection.Assembly.GetExecutingAssembly.Location.LastIndexOf("\"
                     ) + 1)
    With Excel
      .SheetsInNewWorkbook = 1
      .Workbooks.Add()
      .Worksheets(1).Select()
      'For displaying the column name in the the excel file.
      For intColumn = 0 To dsExcelExport.Tables(0).Columns.Count - 1
        .Cells(1, intColumn + 1).Value = dsExcelEx-
                     port.Tables(0).Columns(intColumn).ColumnName.ToString
      Next
      'For displaying the column value row-by-row in the the excel file.
      For intRow = 0 To dsExcelExport.Tables(0).Rows.Count - 1
        For intColumnValue = 0 To dsExcelExport.Tables(0).Columns.Count - 1
          .Cells(intRow + 1, intColumnValue + 1).Value = dsExcelEx-
           port.Tables(0).Rows(intRow).ItemArray(intColumnValue).ToString
        Next
      Next
      'strFileName = InputBox("Please enter the file name.", "Swapnil")
      strExcelFile = strAppPath & "employee"
      .ActiveWorkbook().SaveAs(strExcelFile)
      .ActiveWorkbook.Close()
    End With

MessageBox.Show("File exported sucessfully.", "Exporting done", MessageBoxButtons.OK, MessageBox-
Icon.Information)
NormalExit:
```

```
    Excel.Quit()
    Excel = Nothing
    GC.Collect()
    Exit Function
End Function
```

Apéndice F
Glosario

1. Algoritmo: Es un secuencia de pasos que conforman un procedimiento para solucionar un problema dado.

2. ASP: Sus siglas significan *Active Server Pages – Páginas Activas de Servidor.* Es una especificación que permite crear dinámicamente páginas Web mediante HTML, scripts, y componentes de servidor ActiveX reutilizables.

3. Autenticación: Un esquema de autenticación permite determinar si el mensaje realmente es enviado por el emisor.

4. Base de datos: Colección de datos relacionados lógicamente almacenada de forma organizada y ordenada. Las bases de datos están estructuradas físicamente usando las técnicas de organización de índices.

5. Campo: Los campos son los distintos tipos de datos que componen la tabla.

6. CASE: Sus siglas en inglés significan *Computer Aided Software Engineering,* en español *Ingeniería de Software Asistida por Computadora.* Son instrumentos o sistemas automatizados que brindan soporte a las actividades de producción de software.

7. Clase: Norma o patrón que permite comprender y usar de forma ordenada un conjunto de objetos que comparten características comunes.

8. Cliente: Un ordenador o una aplicación que recurre a los servicios de un servidor.

9. Constraint: Es una regla o restricción que debe cumplirse dentro de una base de datos para las columnas de sus tablas.

10. Cookie: Es un fragmento de información que se almacena en el disco duro del visitante de una página web a través de su navegador, a petición del servidor de la página. Esta información puede ser luego recuperada por el servidor en posteriores visitas.

11. Diagrama: Es una representación gráfica de las relaciones de una colección de elementos de modelado, a menudo dibujada como un grafo con vértices conectados por arcos.

12. Encriptación: Es el proceso por medio del cual un texto plano es cifrado de forma que el resultado es ilegible a menos que se conozcan los datos necesarios para su interpretación. Al texto resultante se le denomina criptograma y constituye un conjunto de caracteres que a simple vista no tiene ningún sentido para el lector.

13. Firebird: Es un sistema de administración de base de datos (o RDBMS) de código abierto, basada en la versión 6 de Interbase, cuyo código fue liberado por Borland en 1999, y de la cual corrige múltiples errores. Su código fue totalmente reescrito en C++.

14. Hash: también llamada función resumen, se refiere a una función o método para generar claves o llaves que representen de manera unívoca un elemento como un registro por ejemplo.

15. Interbase: Es un motor para transacciones SQL que implementa características avanzadas como vistas, triggers, stored procedures entre otras. Interbase fue desarrollado por Borland como una base de datos commercial pero en la actualidad se considera open-source.

16. Llave Primaria: Atributo o conjunto de atributos que identifica de manera única a un registro de una tabla. Esta llave no debe aceptar nulos, no debe cambiar de valor y no debe repetirse.

17. Llave Foránea: Permite la relación entre tablas. Es un atributo o conjunto de atributos que es llave primaria en alguna otra tabla.

18. MD5: (acrónimo de *Message-Digest Algorithm 5 - Algoritmo de Resumen del Mensaje 5*) es un algoritmo de reducción criptográfico de 128 bits ampliamente usado.

19. Modelo: Es una abstracción o vista de un sistema del mundo real. El modelo describe completamente aquellos aspectos del sistema que son relevantes al propósito del modelo, y a un apropiado nivel de detalle.

20. Nulo: El término null o nulo es a menudo utilizado en la computación, haciendo referencia a la nada.

21. Objeto: Unidad atómica que encapsula estado y comportamiento. Es una entidad compleja provista de datos (propiedades, atributos) y comportamiento (funcionalidad, programas, métodos).

22. Replicación: La replicación de datos consiste en el transporte de datos entre servidores, permitiendo que ciertos datos de la base de datos estén almacenados en más de un sitio, y así aumentar la disponibilidad de los datos y mejorar el rendimiento de las consultas globales.

23. Servidor: Un ordenador o software que ofrece servicios a máquinas de cliente distantes o a aplicaciones, como el suministro de contenidos de páginas (textos u otros recursos) o el retorno de los resultados de consultas.

24. Sistema: Un grupo de elementos independientes pero interrelacionados que componen un todo unificado.

25. Stored Procedure: En español procedimiento almacenado, es un programa o procedimiento que es almacenado físicamente en una base de datos. Generalmente son escritos en un lenguaje de bases de datos propietario. La ventaja de un procedimiento almacenado es que al ser ejecutado, en respuesta a una petición de usuario, es ejecutado directamente en el motor de bases de datos, el cual usualmente corre en un servidor separado. Como tal, posee acceso directo a los datos que necesita manipular y solo necesita enviar sus resultados de regreso al usuario, deshaciéndose de la sobrecarga resultante de comunicar grandes cantidades de datos salientes y entrantes.

26. SQL: *Structured Query Language* o *Lenguaje Estructurado de Consultas*, es un estándar implementado por los principales motores o sistemas de gestión de bases de datos relacionales.

27. Trigger: Es un programa auto contenido asociado a una tabla o vista de una base de datos que automáticamente ejecuta una acción cuando una fila de la tabla o vista es insertada, actualizada o eliminada.

28. UML: Sus siglas significan *Unified Modelling language – Lenguaje unificado de Modelado,* es el lenguaje de modelado de sistemas de software más conocido en la actualidad.