PROYECTO DE GRADO

ALVARO LUGO C.
COD. 79.538.985

UNIVERSIDAD NACIONAL ABIERTA Y A DISTANCIA UNAD
FACULTAD DE CIENCIAS BASICAS E INGENIERIA
BOGOTA
2005

SISTEMA PARA MANEJAR LAS COMPRAS Y VENTAS DE LABORATORIO CLÍNICO

ALVARO LUGO C.

PROYECTO DE GRADO PARA OPTAR POR EL TITULO DE TECNOLOGO EN SISTEMAS

UNIVERSIDAD NACIONAL ABIERTA Y A DISTANCIA "UNAD"
FACULTAD DE CIENCIAS BASICAS E INGENIERIA
BOGOTA
2005

NOTA DE ACEPTACION

_____

_____

_____

_____

JURADO

_____

JURADO

Viernes, 29 de Abril de 2005

CONTENIDO

LISTA ESPECIAL

LISTA DE TABLAS

GLOSARIO

ACTOR: Conjunto coherente de roles que juegan los ususarios de los casos de uso cuando interactúan con estos.

ARQUITECTURA: Conjunto de decisiones significativa acerca de la organización de un sistema software, la selección de los elementos estructurales (y sus interfaces) de las que se compone el sistema, junto con su comportamiento tal y como se especifica en las colaboraciones entre esos elementos, la composición de esos elementos estructurales y de comportamiento en subsistemas cada vez mayores y el estilo arquitectónico que orienta esta organización. La arquitectura software no solo tiene que ver con la estructura y el comportamiento, sino también con las restricciones y los compromisos entre uso, funcionalidad, rendimiento, flexibilidad, reutilización, comprensibilidad, economía y tecnología, y con intereses estéticos.

ARTEFACTO: Pieza de información que es utilizada o producida por un proceso de desarrollo de software.

CASOS DE USO: Descripción de un conjunto de secuencias de acciones, incluyendo variantes, que ejecuta un sistema para producir un resultado observable, de valor para un actor.

CLASE: Descripción de un conjunto de objetos que comparten los mismos atributos, operaciones, relaciones y semántica.

CONSTRUCCION: Tercera fase del ciclo de vida del desarrollo de software, en la cual el software se lleva desde una base arquitectónica ejecutable básica hasta su disponibilidad para la comunidad de usuarios.

DIAGRAMA: Representación gráfica de un conjunto de elementos, representado la mayoría de las veces como un grafo conexo de nodos (elementos) y arcos (relaciones).

DIAGRAMA DE ACTIVIDADES: Diagrama que muestra el flujo de control entre actividades.

DIAGRAMAS DE CASOS DE USO: Diagrama que muestra un conjunto de casos de uso y actores y sus relaciones.

DIAGRAMA DE CLASES: Diagrama que muestra un conjunto de clases, interfaces y colaboraciones y sus relaciones. Cubre la vista estática de un sistema.

DIAGRAMA DE COLABORACION: Diagrama de interacción que resalta la organización estructural entre los objetos que envían y reciben señales. Muestra interacciones organizadas alrededor de instancias y los enlaces de unas a otras

DIAGRAMA DE COMPONENTES: Diagrama que muesta la organización y las dependencias entre un conjunto de componentes.

DIAGRAMA DE DESPLIEGUE: Diagrama que muestra la configuración en tiempo de ejecución de los nodos de procesamiento y los componentes que residen en ellos.

DIAGRAMA DE INTERACCION: Diagrama que muestra una interacción, que consta de un conjunto de objetos y sus relaciones, incluyendo los mensajes que pueden enviarse entre ellos. Cubren la vista dinámica de un sistema.

DIAGRAMAS DE OBJETOS: Diagrama que muestra un conjunto de objetos y sus relaciones en un momento dado.

DIAGRAMA DE SECUENCIA: Diagrama de interacción que resalta la ordenación temporal de los mensajes.

DIRIGIDO POR CASOS DE USO: En el contexto del ciclo de vida del desarrollo de software, proceso en el que se utilizan los casos de uso como artefactos principales para estalecer el comportamiento deseado del sistema, para verificar y validar la arquitectura del sistema, para las pruebas y para comunicación entre los usuarios del proyecto.

DIRIGIDO POR EL RIESGO: En el contexo del ciclo de vida del desarrollo de software, proceso en el que cada nueva versión se ocupa principalmente de acometer y reducir los riesgos más significativos para el éxito del proyecto.

DISCIPLINAS: Es un conjunto de actividades (y artefactos relacionados) en un área determinada, como las actividades en el análisis de requisitos.

ELABORACION: Segunda fase del ciclo de vida del desarrollo de software, en la cual se definen la visión del producto y su arquitectura.

ESCENARIO: Secuencia específica de acciones que ilustra un comportamiento.

INCREMENTAL: En el contexto del ciclo de vida del desarrollo de software, proceso que implica la continua integración de la arquitectura del sistema para producir versiones, donde cada nueva versión incluye mejoras incrementales sobre las otras.

INICIO: Primera fase del ciclo de vida del desarrollo, en el cual se lleva la idea inicial del desarrollo hasta el punto de estar suficientemente bien fundada para justificar la entrada en la fase de elaboración.

ITERATIVO: En el contexto del ciclo de vida del desarrollo de software, proceso que implica la gestión de un flujo de versiones ejecutables.

MARCO DE DESARROLLO: Permite seleccionar los artefactos a desarrollar en un proyecto

REQUISITO: Característica, propiedad o comportamiento deseado de un sistema.

TRANSICION: Cuarta fase del ciclo de vida del desarrollo de software, en la que sel software se pone en manos de la comunidad de usuarios.

UML: Lenguaje unificado de modelado, un lenguaje para visualizar, especificar, construir y documentar los artefactos de un sistema con grana cantidad de software

VISTA DE CASOS DE USO: Vista de la arquitctura de un sistema que incluye los casos de uso que describen el comportamiento del sistema tal y como es visto por sus usuarios finales, los analistas y los encargados de pruebas.

VISTA DE DESPLIEGUE: Vista de la arquitectura de un sistema que incluye los nodos que forman la topología hardware sobre la cual se ejecuta el sistema Cubre la distribución, entrega e instalación de las partes que configuran el sistema físico.

VISTA DE DISEÑO: Vista de la arquitectura de un sistema que incluye las clases, interfases y colaboraciones que forman el vocabulario del problema y su solución; se ocupa de los requisitos funcionales de un sistema.

VISTA DE IMPLEMENTACION: Vista de la arquitectura de un sistema que incluye los componentes que se utilzan para ensamblar y hacer disponible el sistema físico; cubre la gestión de configuraciones de las versiones del sistema, formada por componentes relativamente independientes que se pueden ensamblar en varias formas para producir un sistema ejecutable.

INTRODUCCION

El presente trabajo propone el desarrollo de un software que permita la gestión de las compras y las ventas, la generación de la factura y reportes para una empresa que comercializa productos a laboratorios clínicos ubicados en Bogotá.

La empresa suministra reactivos para química sanguínea (Ácido úrico, Albúmina, bilirrubinas total y directa, creatinina, etc), enzimas (Amilasa, CK / NK, CK / NAC, CK / MB, Colinesterasa, etc), Lípidos (Colesterol, HDL Colesterol, Fosfolípidos, Triglicéridos, etc), Electrolitos (Calcio, Fósforo, Hierro, etc), Soluciones estándares (Estándar de Hemoglobina, Estándar de Ácido úrico, Estándar de Cloro, etc), productos para control de calidad, serología, serodiagnóstico, productos varios (Agujas desechables, agujas múltiples vacutainer, aplicadores con o sin algodón, etc), colorantes, productos para urianálisis, etc.

El proyecto se plantea para resolver la problemática de las compras y las ventas de los productos, ya que en la actualidad se llevan de forma manual, haciendo difícil conocer el número de productos comprados y vendidos en un período de tiempo determinado, así como los proveedores y clientes de la empresa.

CAPITULO 1. ASPECTOS GENERALES

1.1. PLANTEAMIENTO DEL PROBLEMA Desde su fundación, la empresa ha venido haciendo compras y ventas de reactivos de laboratorio clínico, registrando tal información de forma manual, lo cual ha llevado a dificultades relacionadas con la utilidad en las ventas, los costos asociados con las compras, la información redundante de proveedores y clientes, y en últimas con la generación de la factura.

Esta problemática se manifiesta:

- Cuando se necesita conocer la cantidad de productos vendidos, y solo después de algún tiempo es posible conocer su cantidad y su precio
- Los productos, los proveedores, los clientes cambian de nombre con el paso del tiempo, ya que no existe un método estandarizado para llamarlos de la misma manera, y un sistema de codificación, si bien ayuda a solucionar este problema, no es intuitivo para el usuario.
- Es difícil reconocer a los clientes que habiendo comprado, no han pagado la factura.
- La empresa no cuenta con un medio eficiente, rápido y confiable para determinar el costo de las compras y las ventas mensuales
- La información de los clientes, los proveedores y los productos se encuentra dispersa

Todos estos problemas son causados por la falta de un sistema de estandarización y centralización de la información, que permita controlar los productos, las compras, las ventas, los proveedores y los clientes de la empresa.

.1.1. FORMULACION

¿Es posible crear un sistema software para la captura, modificación, eliminación de productos, proveedores, clientes, compras, ventas, y que asista a la empresa en la generación de la factura de venta?

.1.2. DELIMITACIÓN

El proyecto creará un sistema para la captura, modificación, eliminación de productos y/o reactivos de laboratorio clínico, proveedores, clientes, compras y ventas. El sistema estará en capacidad de asistir la generación de la factura, por cuanto este mantendrá un registro interno de control del inventario, con el cual podrá determinar las existencias de productos.

El sistema capturará información única y específicamente para la ciudad de Bogotá, ya que en esta se suceden todos los eventos anteriormente mencionados.

.2. ANALISIS DE VARIABLES

- Lenguajes de Programación: Visual Basic 6.0. Deplhi 5.0. Java. Se selecciona Visual Basic. Si bien soporta la herencia a través de delegación, y no es por completo un lenguaje orientado a objetos, soporta la implementación de interfaces, polimorfismo y encapsulación, permite además, la creación de controles, librerías, etc. Delphi y java, si bien son lenguajes orientados completamente a objetos, no se utilizan, por cuanto la experiencia en desarrollo bajo estos lenguajes es escasa, lo cual dificulta la creación de la solución, e impide entrever futuras dificultades en el uso del lenguaje.
- Base de Datos: Access 2000. SQL Server. MSDE. Oracle: Se utiliza access por cuanto existía en esta base cierta información almacenada. SQL Server no se utiliza por el costo de las licencias, MSDE si bien es gratuito, molesta a la instalación en Windows 98. Oracle el costo de sus licencias lo hacen prohibitivo
- Herramienta para la Generación de Reportes: Cristal Reports. Datareport. Se utilizará el DataReport, ya que este viene incluído en Visual Basic, y se cuenta con experiencia en el uso del mismo
- Modelado: Rational Rose, Enterprise Architect. Se utilizará el último, puesto que su demo descargado por internet tiene altísima funcionalidad, y su costo esta al alcance. Rational Rose no se utiliza por su elevado precio.
- **Compras de Productos:** Se refiere a la compra de productos a proveedores. En este punto es necesario capturar la información del proveedor, el número de factura, la fecha de la compra, los productos comprados, el total de unidades, su costo unitario, descuentos e IVA.

  Tal información debe estar disponible para su búsqueda por factura, proveedor, facturas relacionadas con el producto, y año y mes de compra.
- **Ventas de Productos:** Este punto se refiere a la venta de los reactivos de laboratorio clínico comprados. Debe permitir la búsqueda de las ventas, de acuerdo al número de la factura, la fecha de la orden de pedido, ventas canceladas, nombre del comprador, producto vendido, y facturas vendidas en un mes determinado.

  También debe permitir el registro de nuevas ventas. En tal sentido debe capturar información del número de la factura, la fecha en la cual se ordena la compra, la fecha solicitada de entrega, el comprador, el vendedor, la fecha del embarque, el transportador, el flete, cancelado, forma de pago y productos vendidos, cantidad y precio de los mismos
- **Productos:** Se refiere a la información relativa a los productos de laboratorio clínico que la empresa va a vender, su nombre, categoría, descripción, precio de compra, valor del IVA, descuentos, e información sobre si el producto esta o no descontinuado.
- **Proveedores – Clientes :** Se refiere a las personas naturales o jurídicas que tienen relaciones comerciales con la empresa. De estas personas es necesario conocer la razón social, el NIT, el código, el nombre del contacto, la dirección, el código postal, el teléfono, el fax, y la información de la página web.

- **Reportes de Compras y Ventas: deben**Permiten determinar:
  - Valor de las compras y las ventas del mes
  - Listado de Productos
  - Cantidad de Productos comprados y vendido en el mes

.3. OBJETIVOS

.3.1. GENERAL

Analizar, diseñar e implementar un sistema software para el manejo de la información de los proveedores, clientes, productos comprados y vendidos a laboratorios clínicos de la ciudad de Bogotá, y la generación de la factura, para AMI Diagnostica, empresa comercializadora de productos para laboratorio clínico, ubicada en Bogotá

.3.2. ESPECIFICOS

- Llevar el control de ventas, compras, proveedores y clientes.
- Generar la factura de venta.
- Capturar la información de los productos comprados
- Auxiliar al usuario en el procesos de generación de la factura
- Crear la base de datos que mantenga información de proveedores, clientes, productos comprados y vendidos
- Generar reportes de productos mas comprados y más vendidos

.4. JUSTIFICACION

La sistematización de los datos relativos a las compras y ventas de la empresa permitirá la toma de decisiones rápida y oportuna. La información generada por el sistema, no solo será fiable y confiable, sino que además será instantánea y la toma de decisiones a nivel gerencial ágil

El proyecto tiene una aplicación directa en el campo de las ventas de reactivos de Laboratorio Clínico. Es capaz de mostrar resultados concretos al mejorar los procesos de la empresa, siendo una solución a un problema administrativo

.5. HIPOTESIS

.5.1. GENERAL

El sistema permitirá la centralización de la información capturada. Permitirá la adición, actualización y eliminación de registros de productos, proveedores, clientes, compras, ventas y facturación, manteniendo la integridad de la misma.

.5.2. DE TRABAJO

El sistema capturará la información de los productos comprados y vendidos, Generará la factura para los productos vendidos, en el formato establecido por el solicitante del proyecto, capturará la información de los productos, su precio de compra, IVA, descuentos, y precio recomendado de venta, y generará los reportes necesarios para las Compras y Ventas, Valores y Consolidados, Listado de productos, con precios sugeridos de venta para obtener ganancias de 10%, 20%, 30%, 40% y reportes de productos mas comprados y más vendidos

CAPITULO 2. MARCOS DE REFERENCIA

2.1. MARCO TEORICO

El presente proyecto será desarrollado utilizando Visual Basic 6.0. ADODB (y sus objetos connection, command, recordset y parameter.), clases datasource, controles de usuario (OCX) y librerías DLL. Para el almacenamiento persistente de la información, se utilizará Microsoft Access.

Para la planificación del mismo, se utilizará el Proceso Unificado, en el cual, el desarrollo se organiza en una serie de mini – proyectos cortos, de duración fija llamadas iteraciones (Inicio, Elaboración, Construcción y Transición.), cuyo resultado es un un sistema que puede ser probado, integrado y ejecutado. Cada iteración incluye sus propias actividades de análisis de requisitos, diseño, implementación y pruebas.

Los artefactos a elaborar en el presente proyecto se pueden observar en el marco de desarrollo.

Disciplina	**Artefacto**	**Inicio I1**	**Elab E1..En**	**Cons C1..Cn**	**Trans T1..Tn**
Modelado del Negocio	Modelo del Dominio		C		
Requisitos	Modelo de Casos de Uso	C	R		
	Especificación Complementaria	C	R		
	Glosario	C	R		
Diseño	Modelo de Diseño de Software		C	R	
	Modelo de Datos		C	R	
Implementación	Modelo de Implementación de Software		C	R	R
Entorno	Marco de Desarrollo	C	R		
Gestión del Proyecto	Plan de Desarrollo de Software	C	R	R	R
Pruebas	Modelo de Pruebas de Software		C	R	R

Elab = Fase Elaboración. Cons = Fase Construcción. Trans = Fase de Transición C = Comenzar. R = Refinar

Tabla 1. Marco de Desarrollo

El presente proyecto será desarrollado de la siguiente manera:

a. Creación del marco de desarrollo (Selección de los artefactos a desarrollar)
b. Creación del modelo en UML
c. Determinación del tiempo para el desarrollo del producto, teniendo como base los casos de uso
d. Creación del Cronograma de Actividades
e. Desarrollo del sistema

2.2. ANTECEDENTES

Ante la alternativa de desarrollar un nuevo producto, con los consabidos riesgos, costos y dificultad en la estimación del tiempo, se plantea la posibilidad de compra de software ya desarrollado y disponible en el mercado.

Sin embargo, se considera que el software para la solución del problema, antes de ser muy sofisticado y costoso, debe aportar unos elementos mínimos que permitan por un lado ayudar en la gestión de la empresa, y por el otro ayudar a establecer si la empresa es o no viable.

Por ello, se opto por la construcción de un software sencillo, capaz de llevar la información básica de los productos, su costo, los clientes y proveedores de la empresa, generar la factura de venta y los reportes necesarios para establecer pérdidas o ganancias en el período de tiempo estimado.

2.3. MARCO CONCEPTUAL

UML es un lenguaje gráfico para visualizar, especificar, construir y documentar los artefactos de un sistema con gran cantidad de software. UML proporciona una forma estándar de escribir los planos de un sistema, cubriendo tanto las cosas conceptuales, tales como procesos del negocio y funciones del sistema, como cosas concretas, tales como las clase escritas en un lenguaje de programación específico, esquemas de bases de datos y componentes software re-utilizables.

El objetivo de uml es ayudar a producir de una manera consistente software de calidad que satisface las necesidades de sus usuarios, desarrollando software de forma predecible y puntual, con un uso eficiente y efectivo de los recursos, tanto humanos como materiales.

Para desarrollar software rápida, eficiente y efectivamente con el mínimo de desechos software y de trabajo repetido, hay que tener la gente apropiada, las herramientas apropiadas y el enfoque apropiado. Para hacer todo esto de forma consistente y predecible, con una estimación de los costes del sistema en cada etapa de su vida, hay que disponer de un proceso de desarrollo sólido que pueda adaptarse a las necesidades cambiantes del problema en cuestión y de la tecnología.

El modelado es una parte central de todas las actividades que conducen a la producción de buen software. Construimos modelos para comunicar la estructura deseada y el comportamiento de nuestro sistema. Construímos modelos para visualizar y controlar la arquitectura del sistema. Construimos modelos para comprender mejor el sistema que estamos construyendo, muchas veces descubriendo oportunidades para la simplificación y la reutilización. Construímos modelos para controlar el riesgo.

A través del modelado, se consiguen cuatro objetivos:

1. Los modelos nos ayudan a visualizar cómo es o queremos que sea un sistema
2. Los modelos nos permiten especificar la estructura del comportamiento de un sistema
3. Los modelos nos proporcionan plantillas que nos guían en la construcción de un sistema
4. Los modelos documentan las decisiones que hemos adoptado.

Cuanto más grande y complejo es el sistema, el modelado se hace más importante, por la simple razón que construímos modelos de sistemas complejos porque no podemos comprender el sistema en su totalidad.

El uso del modelado tiene una historia en todas las disciplinas de la ingeniería. Esa experiencia sugiere cuatro principios básicos de modelado:

Primero: La elección de qué modelos crear tiene una profunda influencia sobre como se acomete el un problema, y cómo se forma la solución.

Segundo: Todo modelo puede ser expresado a diferentes niveles de precisión.

Tercero: Los mejores modelos están ligados a la realidad.

Cuarto: Un único modelo no es suficiente. Cualquier sistema no trivial se aborda mejor a través de un pequeño conjunto de modelos casi independientes.

Lo mismo es cierto para los sistemas software orientados a objetos. Para comprender la arquitectura de tales sistemas, se necesitan varias vistas complementarias y entrelazadas: una vista de casos de uso(que muestre los requisitos del sistema), una vista de diseño (que capture el vocabulario del espacio del problema y del espacio de la solución), una vista de procesos (que modele la distribución de los procesos e hilos del sistema) y una vista de despliegue (que se centre en las cuestiones de ingeniería del sistema. Cada una de estas vistas puede tener aspecto tanto estructurales como de comportamiento. En conjunto, estas vistas representan los planos del software.

Según la naturaleza del sistema, algunos modelos pueden ser más importantes que otros. Por ejemplo, en sistema con grandes cantidades de datos, dominarán los modelos centrados en las vistas de diseño estáticas. En sistemas con uso intensivo de interfaces gráficas del usuario, las vistas de casos de uso estáticas y dinámicas sob bastante importantes. En sistemas de tiempo real muy exigentes, las vistas de procesos dinámicas tienden a ser más importantes. Finalmente, en los sistemas distribuidos, como los encontrados en aplicaciones de uso intensivo de la web, los modelos de implementación y despliegue son los más importantes.

UML es bastante independiente del proceso. Para obtener el máximo beneficio de UML, se debería considerar un proceso que fuese:

- Dirigido por los casos de Uso: Significa que los casos de uso se utilizan como un artefacto básico para establecer el comportamiento deseado del sistema, para verificar y validar la arquitectura del sistema, para las pruebas y para la comunicación entre las personas involucradas en el proyecto.
- Centrado en la arquitectura: Significa que la arquitectura del sistema se utiliza como un artefacto básico para conceptuar, construir, gestionar y hacer evolucionar el sistema en desarrollo.
- Iterativo e incremental: Es aquel que involucra la gestión de un flujo de ejecutables del sistema. Un proceso incremental es aquel que involucra la continua integración de la arquitectura del sistema para producir esos ejecutables, donde cada nuevo ejecutable incorpora mejoras incrementales sobre los otros. En conjunto un proceso iterativo es incremental está dirigido por el riesgo, lo que significa que cada nueva versión se encarga de atacar y reducir los riesgos más significativos para el éxito del proyecto

Este proceso dirigido por los casos de uso, centrado en la arquitectura, iterativo e incremental puede descomponerse en fases. Una fase es el intervalo de tiempo entre dos

hitos importantes del proceso, cuando se cumplen un conjunto de objetivos bien definidos, se completan los artefactos y se toman las decisiones sobre si pasar o no a la siguiente fase.

Existen cuatro fases en el ciclo de vida del desarrollo del software: Inicio, elaboración, construcción y transición.

En el inicio, se clarifican y fundamentan las ideas iniciales, de tal forma que se garantice la entrada a la fase de elaboración.

En la elaboración, se define la visión del producto y su arquitectura. Se expresan los requisitos del sistema, se priorizan y se utilizan para crear una base arquitectónica fuerte.

En la construcción, el software se lleva desde una base arquitectónica ejecutable hasta su disponibilidad para la comunidad de usuarios. Los requisitos del sistema, y los criterios de evaluación son reexaminados constantemente frente a las necesidades del proyecto, y los recursos se asignan al proyecto de forma apropiada para atacar los riesgos.

En la transición, el software es puesto en las manos de la comunidad de usuarios. En esta fase se erradican errores, se añaden características y se mejora continuamente el sistema.

Un elemento que distingue a este proceso y que afecta a las cuatro fases es una iteración. Una iteración es un conjunto bien definido de actividades, con un plan y unos criterios de evaluación bien establecidos, que acaba en una versión, bien interna o externa. Esto significa que el ciclo de vida del desarrollo de software puede caracterizarse por involucrar un flujo continuo de versiones ejecutables de la arquitectura del sistema. Este énfasis en la arquitectura vomo un artefacto importante es el que conduce a UML a centrarse en el modelado de las diferentes vistas de la arquitectura de un sistema.

CAPITULO 3. METODOLOGÍA

3.1. Tipo de Investigación: Ingeniería del Software.

La ingeniería de software es una disciplina o área de la informática o ciencias de la se ha convertido en el elemento clave de la evolución de los sistemas y productos informáticos. En las pasadas cuatro décadas, el software ha pasado de ser una resolución de problemas especializada y una herramienta de análisis de información, a ser una industria por si misma. Pero la temprana cultura e historia de la "Programación" ha creado un conjunto de problemas que persisten todavía hoy. El software se ha convertido en un factor que limita la evolución de los sistemas informáticos. El software se compone de programas, datos y documentos. Cada uno de estos elementos componen una configuración que se crea como parte del proceso de la ingeniería del software. El intento de la ingeniería del software es proporcionar un marco de trabajo para construir software de mayor calidad.

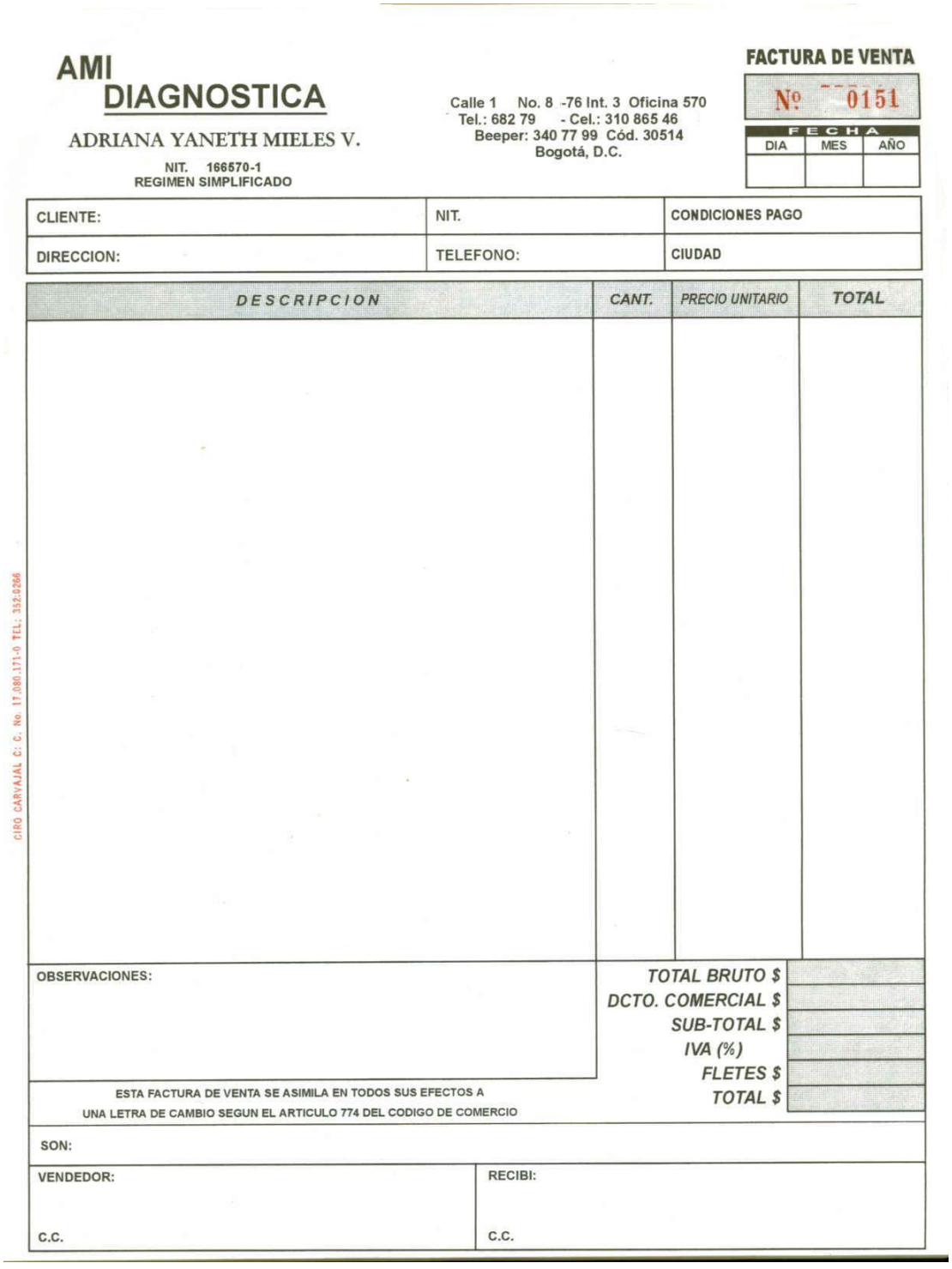
3.2. ALTERNATIVA DE TRABAJO DE GRADO

Proyecto de desarrollo empresarial y tecnológico: Es un proyecto empresarial, puesto que será aplicado en un empresa real que comercializar productos para laboratorio Clínico.

CAPITULO 4. ETAPAS O FASES

4.1. FASE DE EXPLORACIÓN

4.1.1. Observación Directa: La información del presente documento ha sido obtenida del análisis de facturas de la empresa (fig. 1), información de proveedores (fig. 2), documentos relacionados con productos de laboratorio clínico, y la entrevista directa con las personas responsables de la administración de la misma. Se utilizaron estas técnicas para la exploración del problema, ya que la empresa cuenta con información detallada de cada una de las mismas (facturas y compras a proveedores), y la persona encargada cuenta con amplia experiencia en la comercialización de esta clase de productos. Todos los hallazgos han sido recogidos en los casos de uso, modelos del dominio, diagramas de secuencia y modelos de despliegue del sistema, descritos posteriormente en este mismo documento.



AMI
DIAGNOSTICA

ADRIANA YANETH MIELES V.

NIT. 166570-1
REGIMEN SIMPLIFICADO

Calle 1 No. 8 -76 Int. 3 Oficina 570
Tel.: 682 79 - Cel.: 310 865 46
Beeper: 340 77 99 Cód. 30514
Bogotá, D.C.

FACTURA DE VENTA

Nº 0151

FECHA		
DIA	MES	AÑO

CLIENTE:	NIT.	CONDICIONES PAGO
DIRECCION:	TELEFONO:	CIUDAD

DESCRIPCION	CANT.	PRECIO UNITARIO	TOTAL

OBSERVACIONES:

TOTAL BRUTO $	
DCTO. COMERCIAL $	
SUB-TOTAL $	
IVA (%)	
FLETES $	
TOTAL $	

ESTA FACTURA DE VENTA SE ASIMILA EN TODOS SUS EFECTOS A
UNA LETRA DE CAMBIO SEGUN EL ARTICULO 774 DEL CODIGO DE COMERCIO

SON:

VENDEDOR:	RECIBI:
C.C.	C.C.

CIRO CARVAJAL C.: C. No. 17.080.171-0 TEL.: 352.0266

Fig. 1. Facturas de AMI Diagnóstica

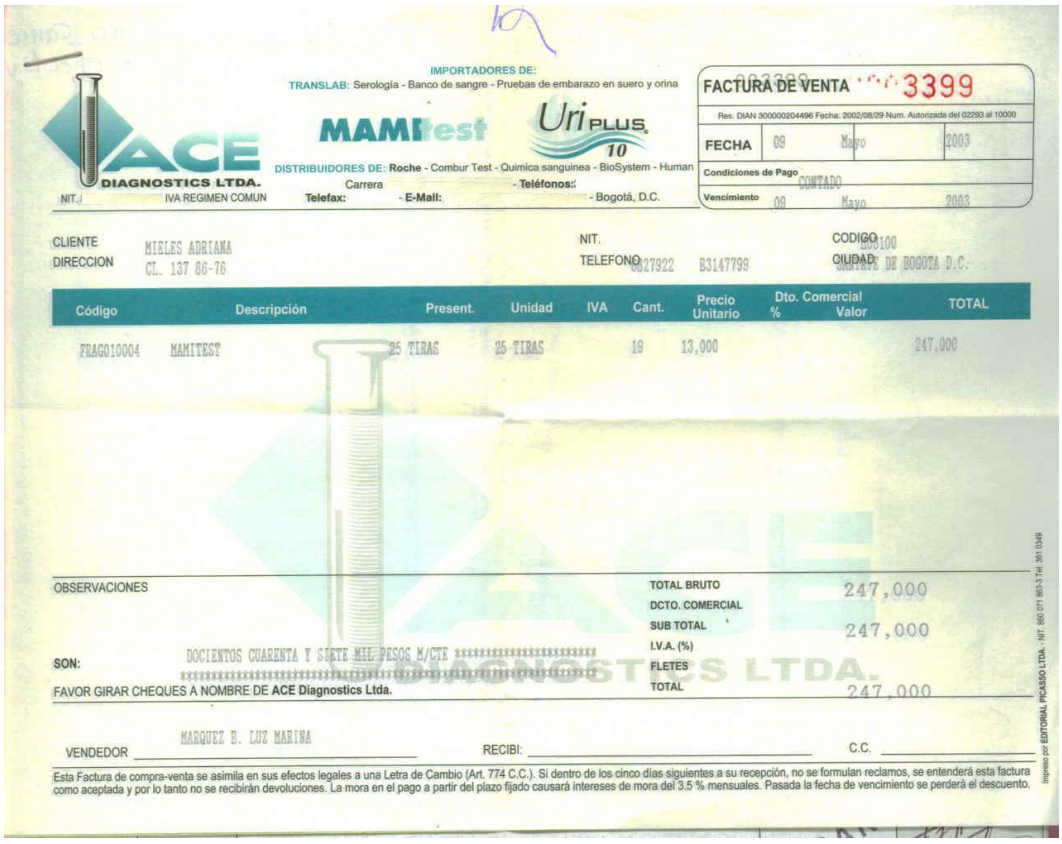


Fig. 2. Proveedores de AMI – Diagnóstica

4.1.2. Identificación y Descripción de los procesos actuales de la empresa

- Compra de Productos de Laboratorio Clínico: Se refiere básicamente a la compra a distribuidores especializados, de reactivos y otros productos de laboratorio clínico.
- Venta de Productos: Se refiere a la venta y distribución de los productos comprados
- Gestión de Productos: Se refiere a la captura de la información del producto como tal (Nombre, categoría, precio, etc.)
- Gestión de Clientes y Proveedores: Información del nombre o razón social, NIT, teléfono, además de otra información relevante.
- Gestión de las Compras y las Ventas en conjunto: Se refiere a los procesos relacionados con la obtención del estado de las ventas y las compras

Los procesos y procedimientos se definen claramente en el numeral 5.3 Fase de Análisis y Diseño, en donde se determinan los casos de uso que se desarrollarán y la correspondiente implementación de los mismos

4.1.3. Técnicas de levantamiento de Información:

- Entrevista: Se utilizó la entrevista para reunir información de la persona conocedora del sistema en estudio, información que fue plasmada en los casos de uso del sistema
- Observación: Documentos, Facturas, productos de laboratorio, permitieron conocer la forma y el manejo de los documentos, y la forma como estos fluyen en los procesos de la ermpresa

4.1.4. Deficiencias

- Compra de Productos de Laboratorio Clínico: No exite identificación única para los productos, proveedores. Difícil encontrar la información de una factura específica, si no es a través de los archivos.
- Venta de Productos: No existe identificación única de productos vendidos, clientes, transportadores de mercancía, vendedor del producto. Se desconoce el número de productos en stock
- Gestión de Productos: Duplicidad en el nombre de los productos
- Gestión de Clientes y Proveedores: Duplicidad en el nombre de clientes y proveedores. Información poco confiable.
- Gestión de las Compras y las Ventas en conjunto: Difícil determinar el total de compras y ventas en un mes determinado, o en un intervalo de tiempo

4.1.5. Estudio de Factibilidad y Análisis Costo - Beneficio

El estudio de factibilidad permite determinar la viabilidad del proyecto, y la posibilidad de que el sistema sea de utilidad para la empresa. El presenta proyecto es factible, ya que ha pasado los estudios correspondientes.

4.1.5.1. Factibilidad Económica

Se considera que los beneficios financieros del sistema son mayores que los costos de su desarrollo.

Factor	**Tiempo (Semanas)**	**Costo**
Diseño	1	50.000
Desarrollo	8	1.000.000
Implementación	1	100.000
Subtotal	**10**	**1.150.000**
Software		**Costo**
Sistema Operativo Win 98		300.000
Visual Basic 6.0		1.600.000
Office		500.000
Subtotal		2.400.000
Otros Gastos e Imprevistos		1.000.000
Subtotal		1.000.000
Total		4.550.000

Tabla 2. Factibilidad Económica

Del estudio financiero se concluye que los costos del desarrollo del sistema son menores que los costos de las licencias de software. Sin embargo, la compra del mismo representa una inversión a corto, mediano y largo plazo, ya que permitirá, por un lado a la empresa trabajar con software legal y por el otro, efectuar modificaciones al software desarrollado, crear uno nuevo, o quizá, explorar un mercado de soluciones software.

4.1.5.2. Factibilidad Técnica

Software: El sistema será construido en Visual Basic 6.0, MS Access 97. Windows 98. Para su análisis y diseño, se empleará a Enterprise Architect

Hardware: Pentium II de 400 Mhz, DD 10 Gigas, 128 RAM

El sistema no necesita ningún tipo de Software o Hardware de difícil consecución. Así mismo el desarrollo puede diseñarse para conseguir la función y el rendimiento necesarios dentro de las restricciones descubiertas en el análisis.

4.1.5.3. Factibilidad Operacional

- Existe apoyo suficiente para el proyecto por parte de la Administración? Si, la administración considera que los procesos actuales son un riesgo para la viabilidad de la empresa a mediano y largo plazo
- Los métodos que actualmente emplea la empresa son aceptados por los usuarios? No. Los métodos actuales son pesados, toman tiempo y son difíciles. La información redunda cuando no se ha extraviado.
- Los usuarios han participado en la planeación y desarrollo del Proyecto? Si. Son precisamente los usuarios quienes han visto la necesidad de un sistema que permita agilizar los procesos de la empresa.
- El sistema propuesto causará perjuicios? El sistema se plantea como un medio básico para la toma de decisiones. No se pretende, ni se necesita un sistema elaborado, sino por el contrario liviano y ágil.
- Se perderá facilidad de acceso a la información?: No. Por el contrario, la información estará centralizada y disponible.
- La productividad de los empleados será menor después que antes de la implantación?: No. Un sistema de bases de datos es suficientemente ágil, confiable, fiable y económico como para permitir afirmar que los procesos no se ralentizarán sino que por le contrario fluirán.

- El sistema planteado trabajará cuando este terminado e instalado, ya que cuenta con el apoyo de la parte administrativa, no existen barreras importantes para la implantación del mismo y existe la técnica, la metodología y los medios para crearlo.

4.1.6. Metas

- Registrar la Compra de Productos de Laboratorio Clínico, identificar a los productos, proveedores. Permitir búsqueda extendida de compras y ventas
- Registrar la Venta de Productos, identificar los productos vendidos, clientes, transportadores de mercancía y vendedores del producto. Permitir la determinación de existencias en stock
- Registrar los productos, identificándolo adecuadamente
- Registrar Clientes y Proveedores,
- Determinación de las Compras y las Ventas, permitiendo totalizar las compras y ventas en intervalos de tiempo determinados

4.1.7. DETERMINACIÓN DE REQUERIMIENTOS

- Adicionar, Actualizar, Eliminar y Buscar Productos de Laboratorio Clínico

- Adicionar, Actualizar, Eliminar y Buscar proveedores - Clientes. Permitir búsqueda extendida de compras y ventas
- Adicionar, Actualizar, Eliminar y Buscar Ventas de Productos,
- Adicionar, Actualizar, Eliminar y Buscar transportadores de mercancía
- Adicionar, Actualizar, Eliminar y Buscar vendedores del producto
- Registrar existencias en stock
- Registrar los productos, identificándolo adecuadamente
- Totalizar las compras y ventas en intervalos de tiempo determinados
- Generar factura de venta
- Asegurar la información
- Evitar duplicidad en la información

4.2. FASE DE DISEÑO

Vista de Casos de Uso

Los casos de uso representan una descripción de los procesos del dominio relacionados.

Modelo de Casos de Uso

Apolo, esta construído en Visual Basic 6.0. La base de datos es Access 2000. Internamente, esta dividido en 4 componentes diferentes:

a. Librería de Compresión (CGZipLibrary.dll)
b. Librería de objetos de Negocios (NorthDLL.dll)
c. Librería de controles ActiveX (NorthUct.ocx)
d. Ejecutable Principal (Apolo)

Estas librerías colaboran de tal forma que en el sistema en conjunto produce la salida de la figura 3. En ella se pueden observar:

- Subsistema de productos: Se encargan de manejar la información relacionada con los diferentes productos que la empresa comercializa (Nombre, categoría, descripción, Precio de compra, precios de venta etc)
- Subsistema de Categorías: Permite agrupar a los productos en categorías específicas (Antígenos febriles, banco de sangre, etc)
- Subsistema de Empleados: Permite capturar la información la información de los empleados de la empresa (No. De Identificación, Nombres, Apellidos, Título de cortesía, etc)
- Subsistema de transportadores: Captura la información de los transportadores (Aquellos que transportan la mercancía, cuando esta tiene que ser enviada a otro lugar del país)
- Subsistema de Proveedores - clientes: Permite almacenar la información de la persona que provee (Natural o jurídica) o las personas que compran un producto
- Ingresar: Permite ingresar al sistema de un modo seguro. Contiene los objetos relacionados con el uso protegido del sistema

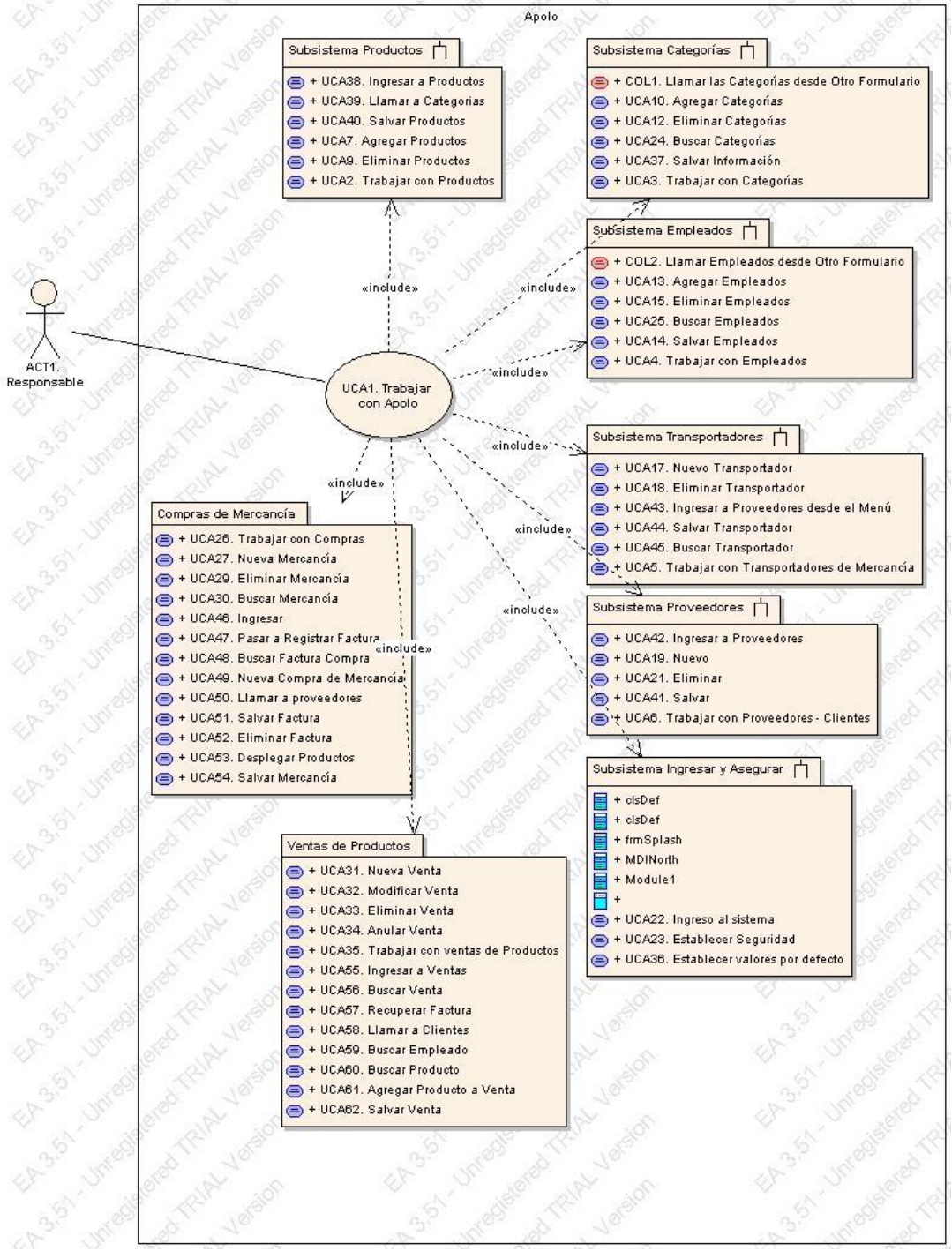


Fig. 3. Modelo de Casos de Uso

- Compras de Mercancía: Permite trabajar con la información al momento de la compra de los productos.
- Ventas de Productos: Permite trabajar con la información al momento de la venta de los productos

La siguiente es una descripción de cada uno de estos subsistemas, y la forma como el sistema los implementa y realiza.

CASOS DE USO

UCA1. Trabajar con Apolo

Tipo: *public* **Use Case**
Estado: Terminado. Versión 1.0. Fase 1.0.
Paquete: Modelo de Casos de Uso

Escenarios

Trabajar con Apolo {Basic Path}.

1. El Actor accede al sistema
2. El sistema despliega la pantalla de inicio. Pregunta el usuario y la contraseña
3. El actor ingresa el usuario y la contraseña
4. El actor ingresa a las pantallas del sistema
5. El actor finaliza la sesión

Subsistema Ingresar y Asegurar

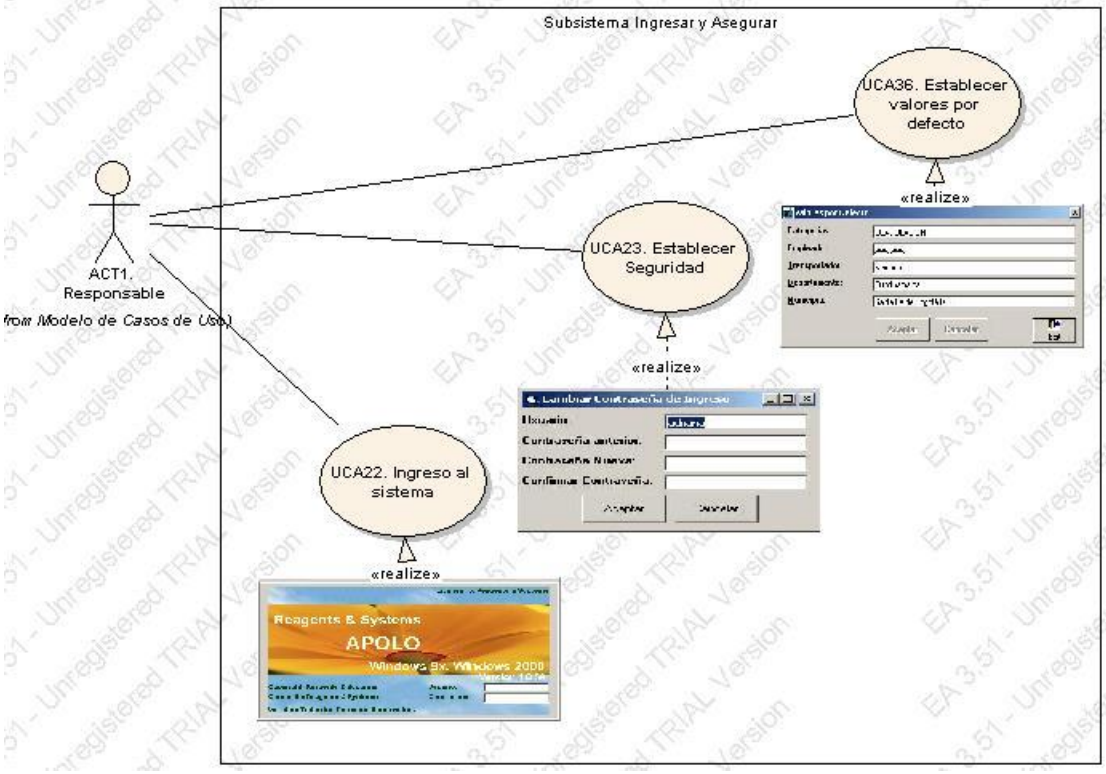


Fig. 4. Ingresar y Asegurar

UCA22. Ingreso al sistema

Tipo: *public* **Use Case**
Estado: Terminado. Versión 1.0. Fase 1.0.
Paquete: Subsistema Ingresar y Asegurar

Objetivo:
- Permitir el ingreso de los usuarios al sistema

Escenarios

Ingresar al Sistema {Basic Path}.

1. El actor inicia Apolo
2. El sistema despliega la pantalla de bienvenida
3. El actor ingresa el Usuario y la contraseña y procede a ingresar (Enter)
4. El sistema establece la conexión con la persistencia
5. El sistema valida el usuario y la contraseña.
6. El sistema despliega el formulario Principal
7. El sistema crea el objeto que maneja la información por defecto del sistema
8. El sistema recupera la información por defecto almacenada en la persistencia

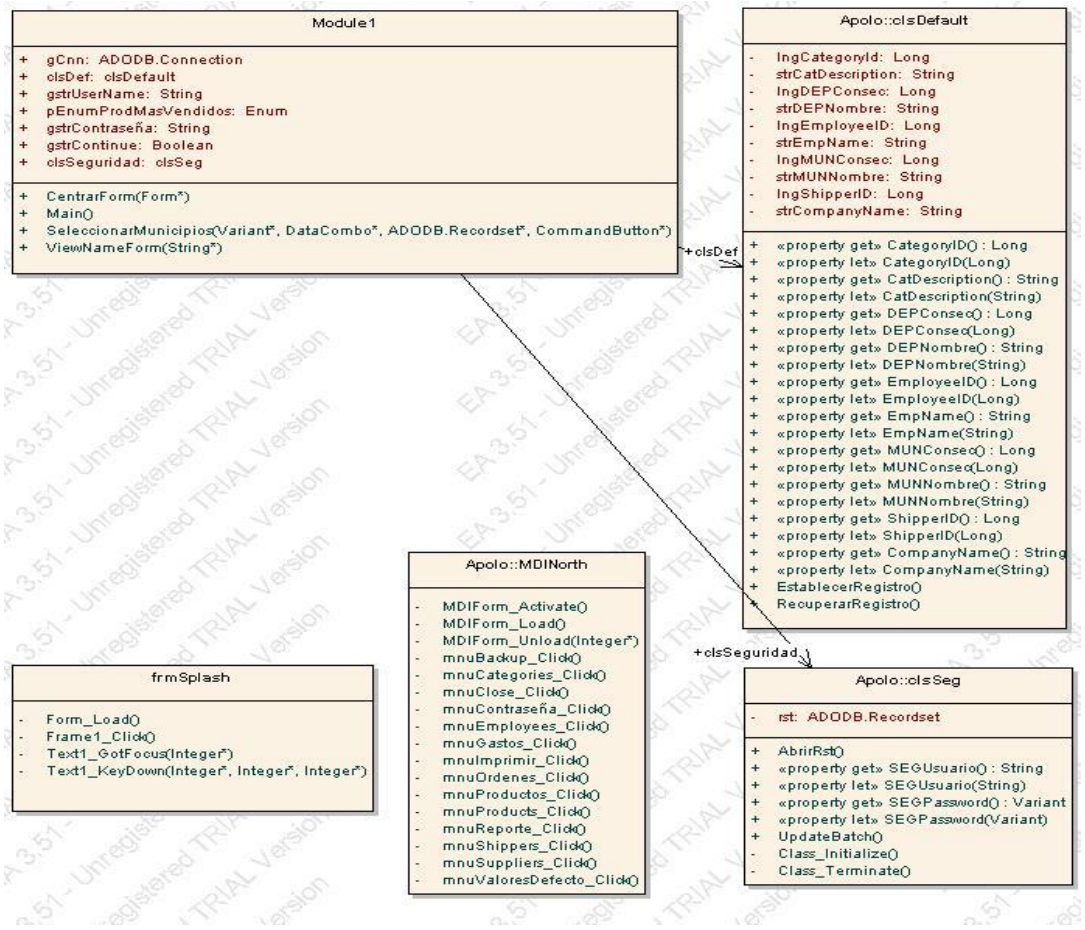


Fig. 5. Ingreso al sistema

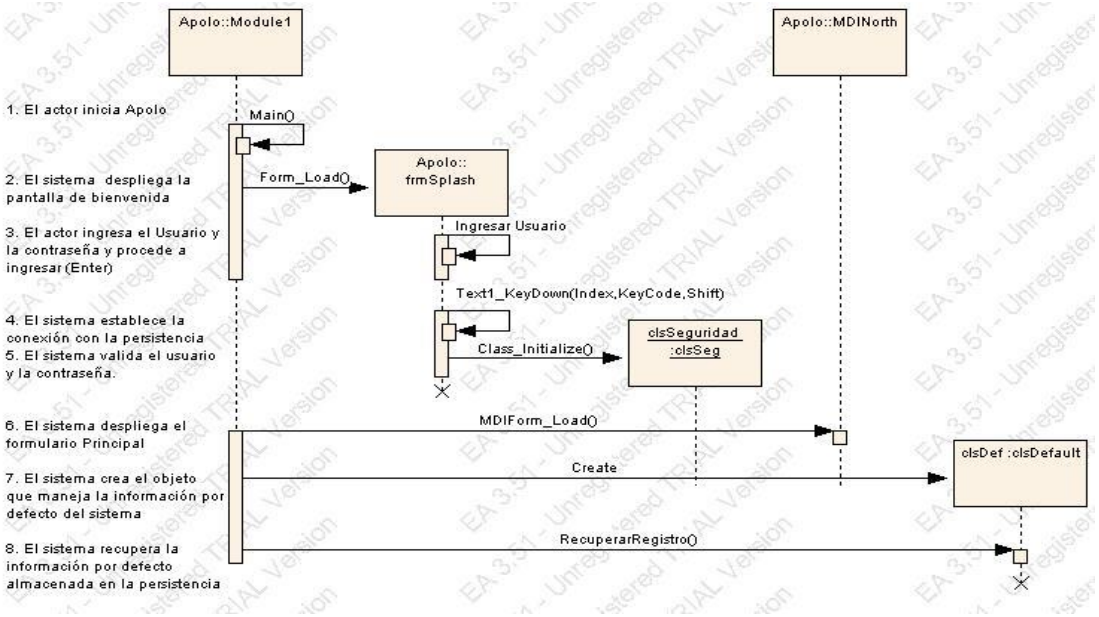


Fig. 6. Secuencia Ingreso al Sistema

Tabla 3. Secuencia Ingreso al Sistema Mensajes

ID	Mensaje	Del Objeto	Al Objeto	Notas
1	Main()	Module1	Module1	Función de inicio del sistema
2	Form_Load()	Module1	frmSplash	Carga el formulario
3	Ingresar Usuario	frmSplash	frmSplash	El actor ingresa el usuario
4	Text1_KeyDown(Integer*, Integer*, Integer*)	frmSplash	frmSplash	Crea la conexión con la base de datos y determina si el usuario que ha ingresado es o no el registrado
5	Class_Initialize()	frmSplash	clsSeguridad	Inicializa la clase. Crea el recordset
6	MDIForm_Load()	Module1	MDINorth	Carga el formulario
7	Create	Module1	clsDef	Crea la clase que almacena los valores por defecto
8	RecuperarRegistro()	Module1	clsDef	Recupera la información en el registro

UCA23. Establecer Seguridad

Tipo: *public* **Use Case**
Estado: Terminado. Versión 1.0. Fase 1.0.
Paquete: Subsistema Ingresar y Asegurar

Objetivo:
- Establecer la seguridad de los usuarios, para impedir violaciones de la misma

Escenarios

Establer seguridad {Basic Path}.
1. El actor ingresa a la pantalla de Cambiar Password
2. El sistema crea la clase controladora
3. El sistema despliega la información del usuario
4. El sistema despliega la pantalla de Cambiar Password
5. El actor procede a cambiar:
 - El usuario,
 - Ingresa la contraseña anterior (Sirve para Verificar que el que hace el cambio, si es el usuario que ingresó a la aplicación)
 - Ingresa la contraseña nueva
 - Confirma la contraseña
6. El actor procede a efectuar los cambios (Click en aceptar)
7. El sistema valida los cambios
8. El sistema almacena la información
9. El sistema descarga la pantalla

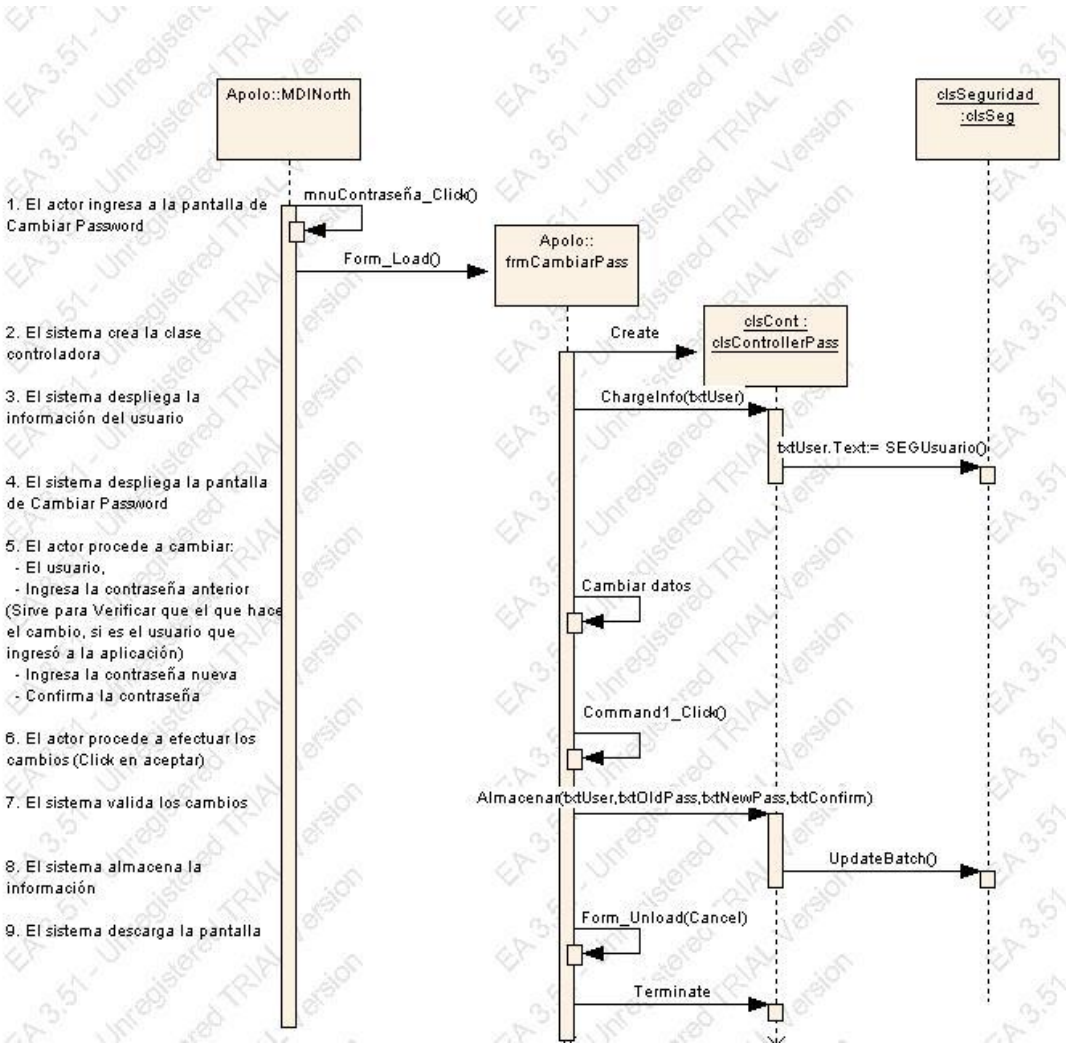


Fig. 7. Establecer Seguridad

Tabla 4. Establecer Seguridad Mensajes

ID	Mensaje	Del Objeto	Al Objeto	Notas
1	mnuContraseña_Clic()	MDINorth	MDINorth	Llama al formulario de cambio de password
2	Form_Load()	MDINorth	frmCambiarPass	Evento load del formulario. Crea la clase controladora
3	Create	frmCambiarPass	clsCont	Crea la clase controladora de la información del password y la contraseña
4	ChargeInfo(TextBox*)	frmCambiarPass	clsCont	Devuelve el usuario desde la clase clsSeguridad
5	SEGUsuario()	clsCont	clsSeguridad	Devuelve el usuario desde la clase clsSeguridad
6	Cambiar datos	frmCambiarPass	frmCambiarPass	El actor procede a cambiar: - El usuario, - Ingresa la contraseña anterior (Sirve para Verificar que el que hace el cambio, si es el usuario que ingresó a la aplicación) - Ingresa la contraseña nueva - Confirma la contraseña
7	Command1_Click()	frmCambiarPass	frmCambiarPass	El actor procede a efectuar los cambios (Click en aceptar)
8	Almacenar(TextBox*, TextBox*, TextBox*, TextBox*)	frmCambiarPass	clsCont	El sistema valida los cambios
9	UpdateBatch()	clsCont	clsSeguridad	El sistema almacena la información
10	Form_Unload(Integer*)	frmCambiarPass	frmCambiarPass	El sistema descarga la pantalla
11	Terminate	frmCambiarPass	clsCont	El Sistema termina la clase controladora

UCA36. Establecer valores por defecto

Tipo: *public* **Use Case**
Estado: Terminado. Versión 1.0. Fase 1.0.
Paquete: Subsistema Ingresar y Asegurar

Objetivo:
- Permitir al usuario ingresar los valores por defecto que se desplegarán en las diferentes pantallas de la aplicación

Escenarios

1. El actor desea cambiar las categorías, empleados, transportadores, Departamentos y municipios
2. El sistem adespliega el formulario de Categorías, empleados, transportadores,

Departamentos y municipios adecuado a la solicitud del usuario.

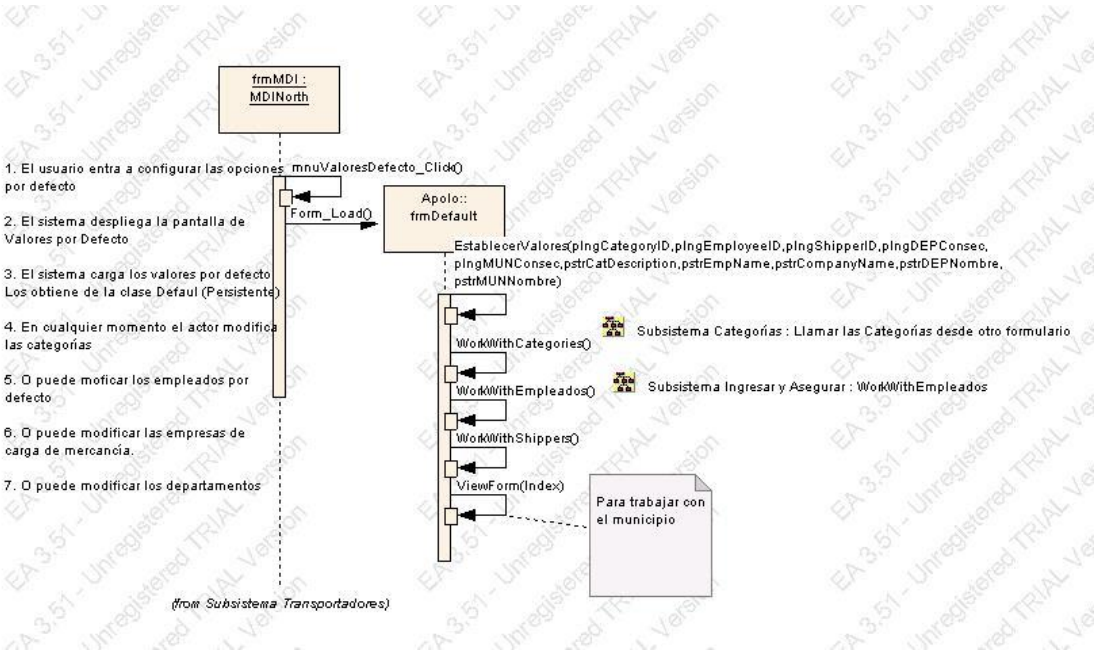


Fig. 8. Establecer Valores por Defecto

Tabla 5. Establecer Valores por Defecto Mensajes

ID	Mensaje	Del Objeto	Al Objeto	Notas
1	mnuValoresDefecto_Click()	frmMDI	frmMDI	El usuario entra a configurar las opciones por defecto
2	Form_Load()	frmMDI	frmDefault	El sistema despliega la pantalla de Valores por Defecto
3	EstablecerValores(Long*, Long*, Long*, Long*, Long*, String*, String*, String*, String*, String*)	frmDefault	frmDefault	El sistema carga los valores por defecto. Los obtiene de la clase Defaul (Persistente)
4	WorkWithCategories()	frmDefault	frmDefault	En cualquier momento el actor modifica las categorías
5	WorkWithEmpleados()	frmDefault	frmDefault	O puede moficar los empleados por defecto
6	WorkWithShippers()	frmDefault	frmDefault	O puede modificar las empresas de carga de mercancía.

7	ViewForm(Integer*)	frmDefault	frmDefault	O puede modificar los departamentos

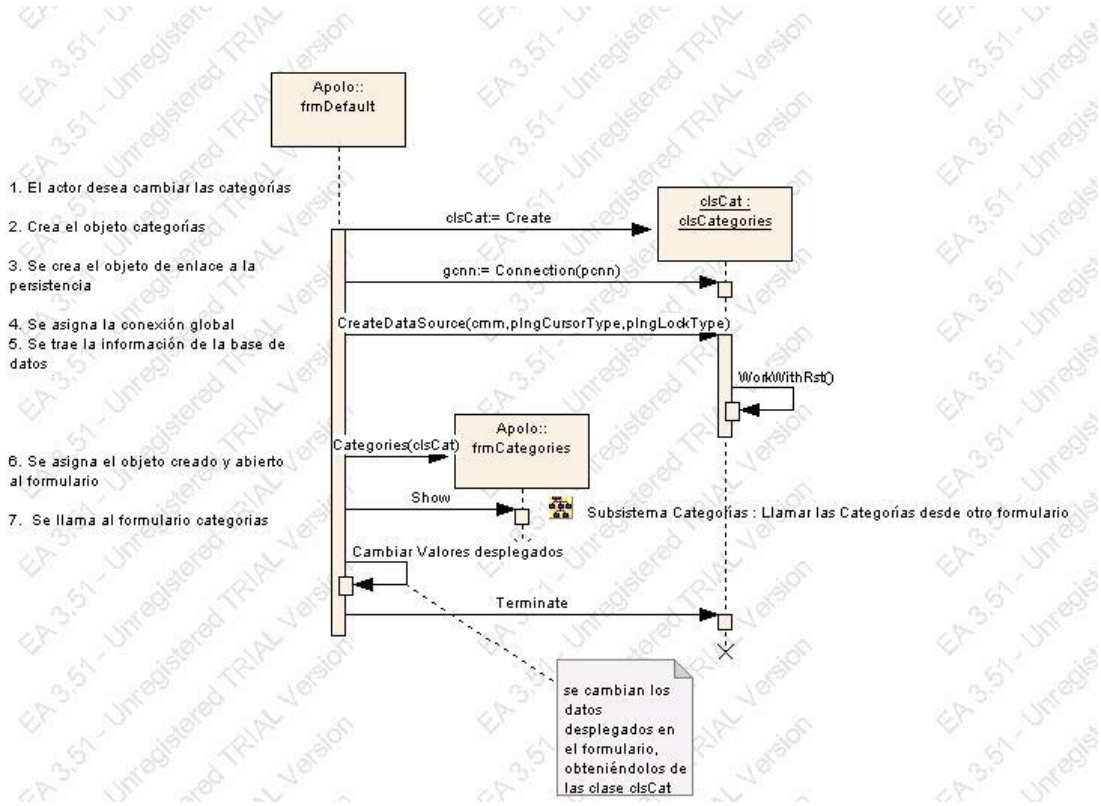


Fig. 9. WorkWithCategories

Tabla 6. WorkWithCategories Mensajes

ID	Mensaje	Del Objeto	Al Objeto	Notas
1	Create	frmDefault	clsCat	Crea el objeto categorías
2	Connection(ADODB.Connection*)	frmDefault	clsCat	Se crea el objeto de enlace a la persistencia
3	CreateDataSource(ADODB.Command*, ADODB.CursorTypeEnum*, ADODB.LockTypeEnum*)	frmDefault	clsCat	Se asigna la conexión global Se trae la información de la base de datos
4	WorkWithRst()	clsCat	clsCat	Crea el recordset

5	Categories(NorthDLL.clsCategories*)	frmDefault	frmCategories	Se asigna el objeto creado y abierto al formulario
6	Show	frmDefault	frmCategories	Se llama al formulario categorías
7	Cambiar Valores desplegados	frmDefault	frmDefault	se cambian los datos desplegados en el formulario, obteniéndolos de las clase clsCat
8	Terminate	frmDefault	clsCat	Se termina la clase

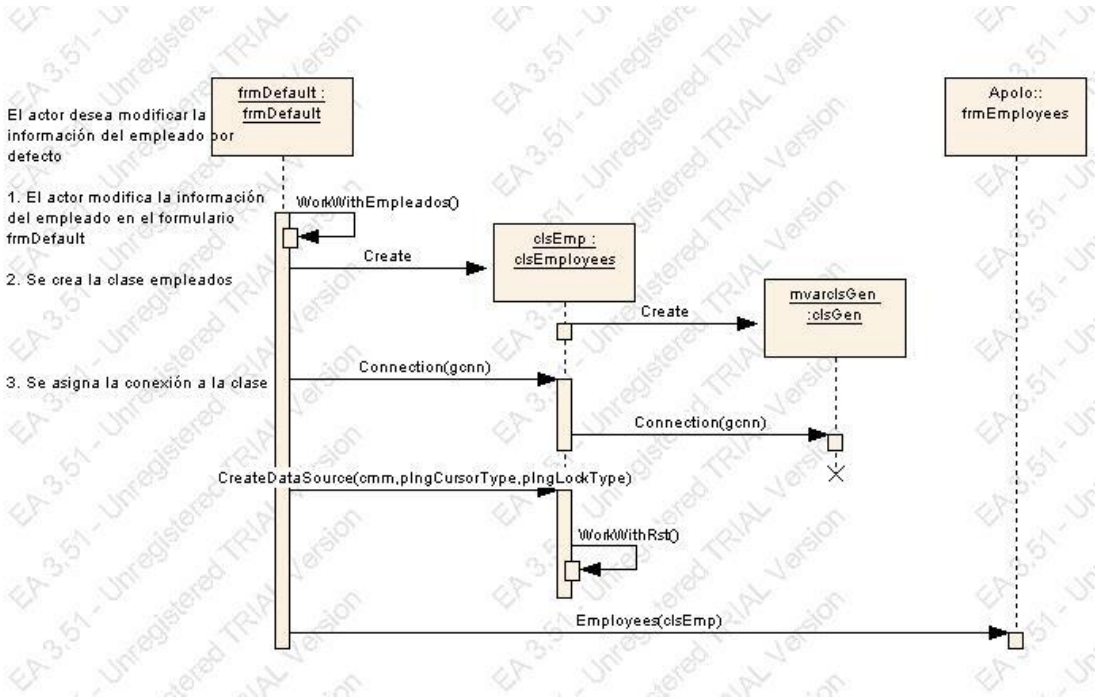


Fig. 10. WorkWithEmpleados

Tabla 7. WorkWithEmpleados Mensajes

ID	Mensaje	Del Objeto	Al Objeto	Notas
1	WorkWithEmpleados()	frmDefault	frmDefault	El actor desea modificar la información del empleado por defecto
2	Create	frmDefault	clsEmp	Se crea el objeto de empleados
3	Create	clsEmp	mvarclsGen	Crea el objeto manejador del recordset
4	Connection(ADODB.Connection*)	frmDefault	clsEmp	Se asigna la conexión a la clase
5	Connection(ADODB.Connection*)	clsEmp	mvarclsGen	Se asigna la conexión a la clase manejadora del recordset

6	CreateDataSource(ADODB.Command*, ADODB.CursorTypeEnum*, ADODB.LockTypeEnum*)	frmDefault	clsEmp	Se crea el origen de los registros
7	WorkWithRst()	clsEmp	clsEmp	Se crea el recordset
8	Employees(NorthDLL)	frmDefault	frmEmployees	Se asigna la clase empleados al formulario de empleados

Subsistema Categorías

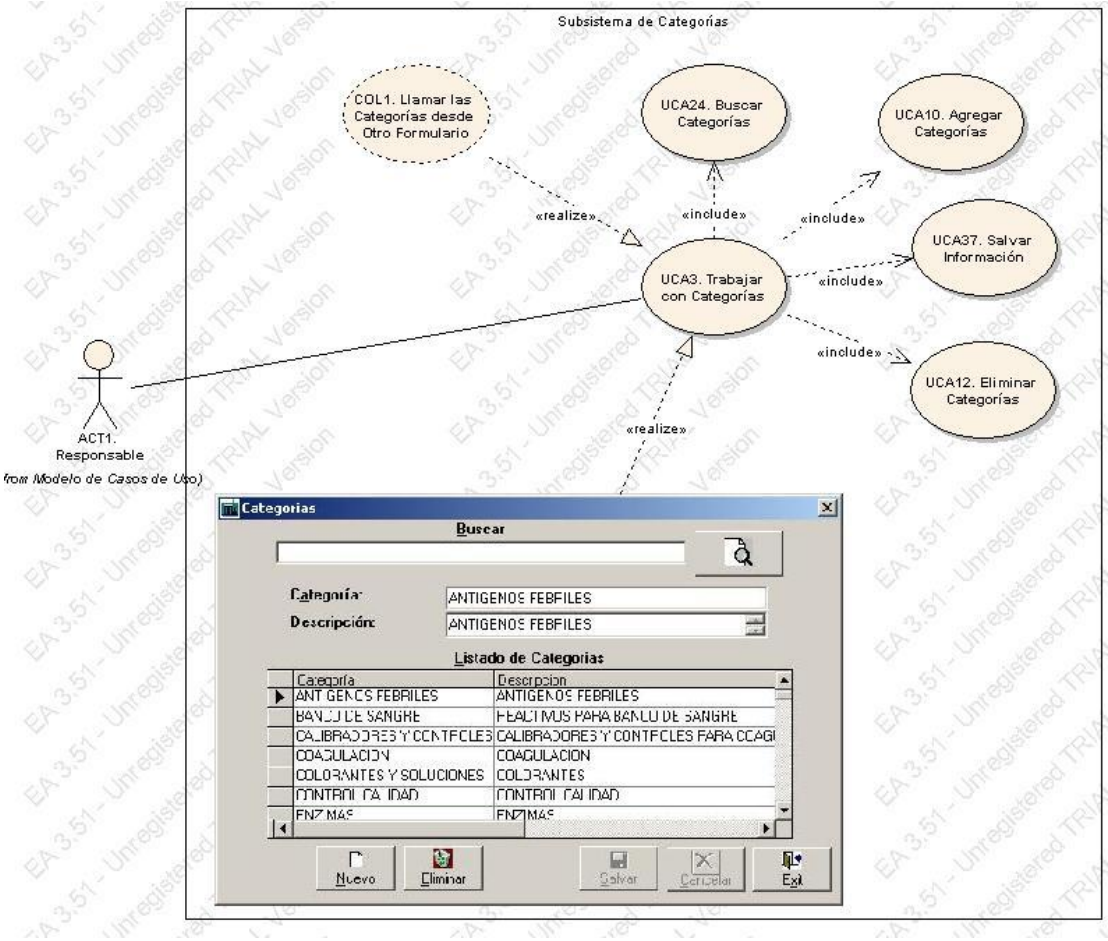


Fig. 11. Categorías

UCA3. Trabajar con Categorías

Tipo: *public* **Use Case**
Estado: Terminado. Versión 1.0. Fase 1.0.
Paquete: Subsistema Categorías

Cuando se trabaja con categorías, se puede:
- Agregar nuevas categorías de productos
- Eliminar categorías de Productos
- Buscar Categorías almacenadas en la persistencia
- Salvar la información nueva o modificada

Escenarios

Trabajar con categorías {Basic Path}.

Al trabaja con categorías de productos, el usuario puede:
1. Agregar nuevas categorías de productos. Incluye UCA10. Agregar Categorías
2. Eliminar categorías de Productos. Incluye UCA12. Eliminar Categorías
3. Buscar Categorías almacenadas en la persistencia. Incluye UCA24. Buscar Categorías
4. Salvar la información nueva o modificada. Incluye UCA 37. Salvar Información

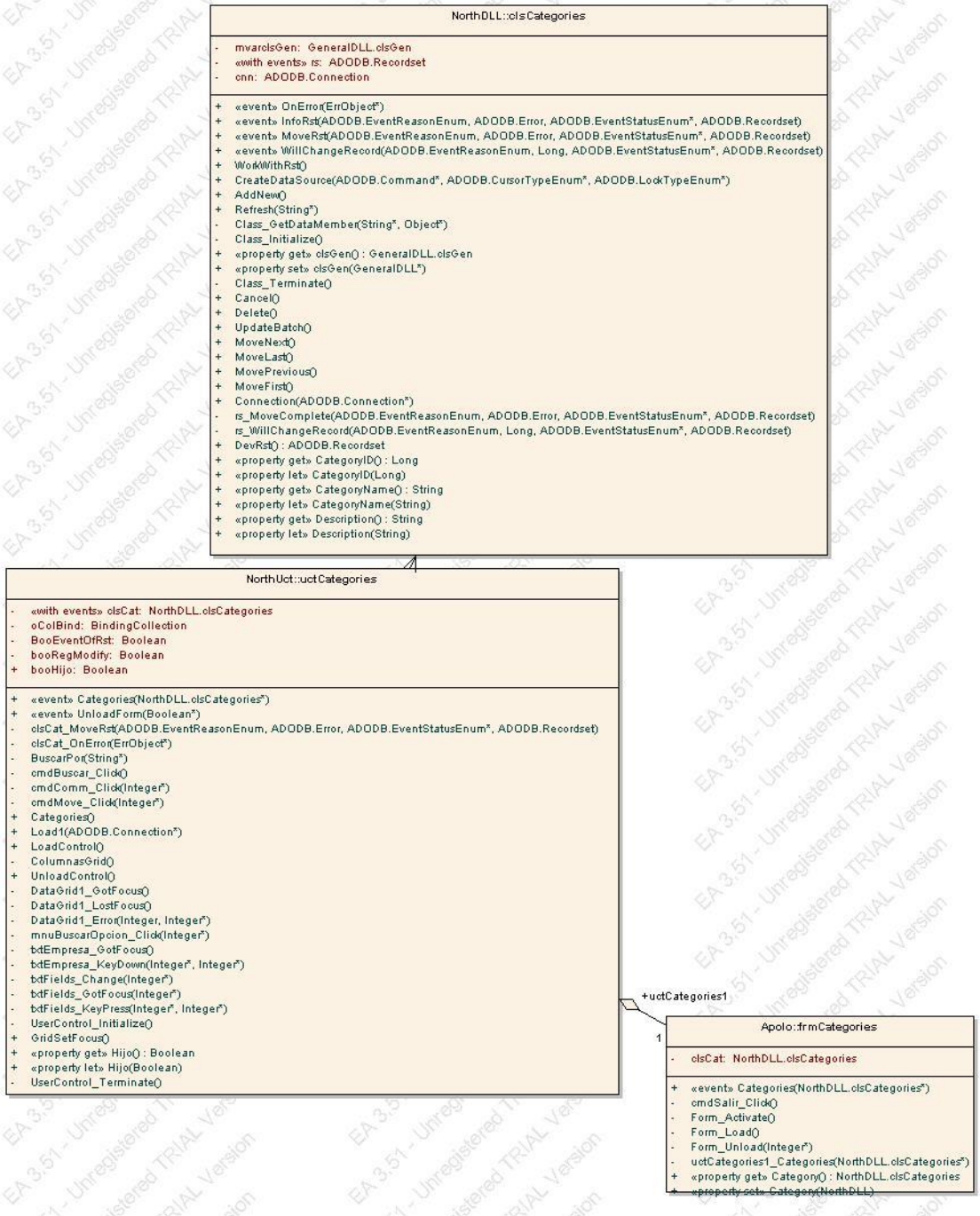


Fig. 12. Subsistema Categorías

COL1. Llamar las Categorías desde Otro Formulario

Tipo: *public* **Collaboration**
Estado: Terminado. Versión 1.0. Fase 1.0.
Paquete: Subsistema Categorías

Objetivo:
- Cooperar con los formularios que requieran información de las categorías

Escenarios

llamar categorías desde otro formulario {Basic Path}.

1. El actor inicia el llamado de las categorías
2. El sistema despliega el formulario completamente funcional de las categorías

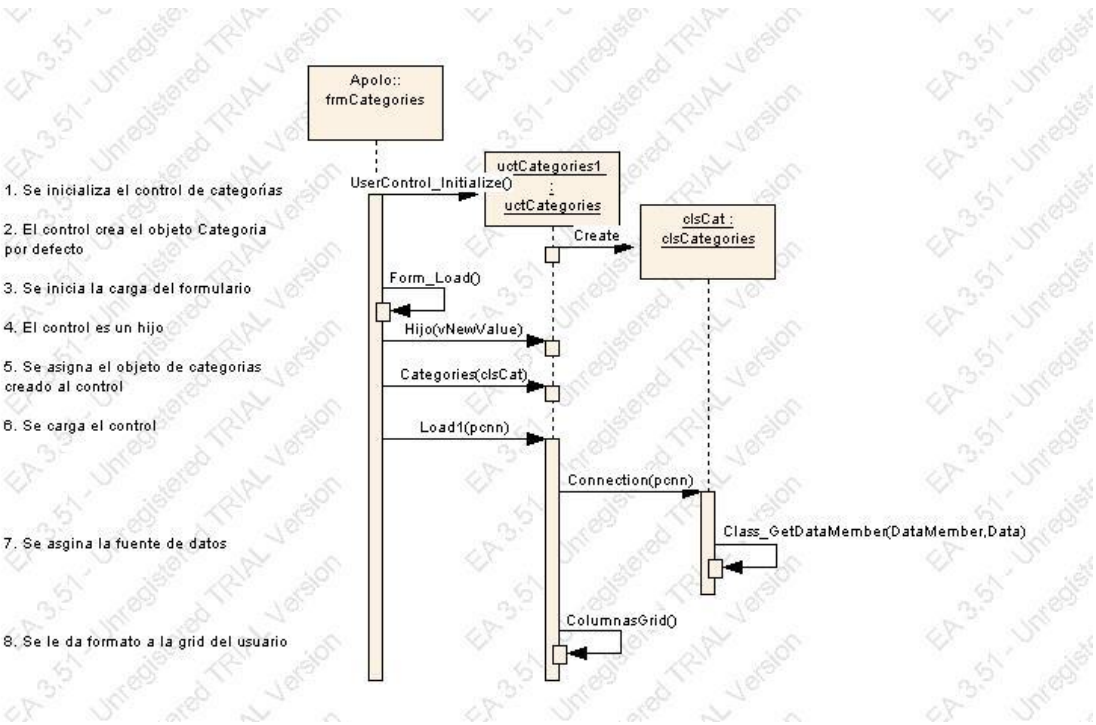


Fig. 13. Llamar las Categorías desde otro formulario

Tabla 8. Llamar las Categorías desde otro formulario Mensajes

ID	Mensaje	Del Objeto	Al Objeto	Notas
1	UserControl_I	frmCategories	uctCategories1	Se inicializa el control de categorías

	nitialize()			
2	Create	uctCategories1	clsCat	El control crea el objeto Categoria por defecto
3	Form_Load()	frmCategories	frmCategories	Se inicia la carga del formulario
4	Hijo(Bolean)	frmCategories	uctCategories1	El control es un hijo
5	Categories(NorthDLL.clsCategories*)	frmCategories	uctCategories1	Se asigna el objeto de categorias creado al control
6	Load1(ADODB.Connection*)	frmCategories	uctCategories1	Se carga el control
7	Connection(ADODB.Connection*)	uctCategories1	clsCat	Se asigna la conexión
8	Class_GetDataMember(String*, Object*)	clsCat	clsCat	Se asgina la fuente de datos
9	ColumnasGrid()	uctCategories1	uctCategories1	Se le da formato a la grid del usuario

UCA10. Agregar Categorías

Tipo: *public* **Use Case**
Estado: Terminado. Versión 1.0. Fase 1.0.
Paquete: Subsistema Categorías

Objetivo:
- Adiciona nuevas categorías

Escenarios

Agregar Una nueva Categoría {Basic Path}.
El actor desea agregar una nueva categoría
1. El actor selecciona el botón nuevo
2. El sistema le dice a la clase categorías que se va a adicionar un registro
3. El recordset genera el evento MoveRst. Se actualizan los campos y los botones
4. El actor ingresa la información solicitada por el sistema
5. El actor Guarda la información

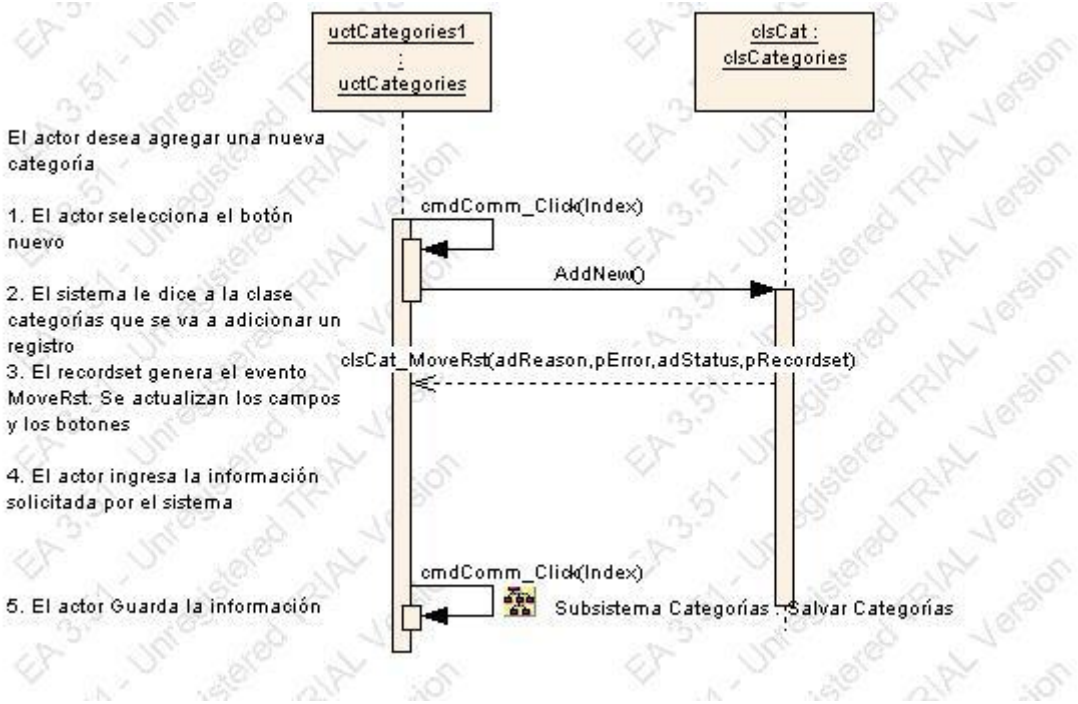


Fig. 14. Salvar Categorías

Tabla 9. Salvar Categorías Mensajes

ID	Mensaje	Del Objeto	Al Objeto	Notas
1	cmdComm_Click(Integer*)	uctCategories1	uctCategories1	El actor desea agregar una nueva categoría. El actor selecciona el botón nuevo
2	AddNew()	uctCategories1	clsCat	El sistema le dice a la clase categorías que se va a adicionar un registro
3	clsCat_MoveRst(ADODB.EventReasonEnum, ADODB.Error, ADODB.EventStatusEnum*, ADODB.Recordset)	clsCat	uctCategories1	El recordset genera el evento MoveRst. Se actualizan los campos y los botones
4	cmdComm_Click(Integer*)	uctCategories1	uctCategories1	El actor ingresa la información solicitada por el sistema. El actor Guarda la información

UCA12. Eliminar Categorías

Tipo: *public* **Use Case**
Estado: Terminado. Versión 1.0. Fase 1.0.
Paquete: Subsistema Categorías

Objetivo:
- Permitir la eliminación de las categorias deseadas

Escenarios

Eliminar Categoría {Basic Path}.
El actor selecciona un registro para eliminar.

1. Continúa, seleccionando el botón eliminar
2. El sistema genera el evento MoveRst. Se actualizan los campos, y los botones

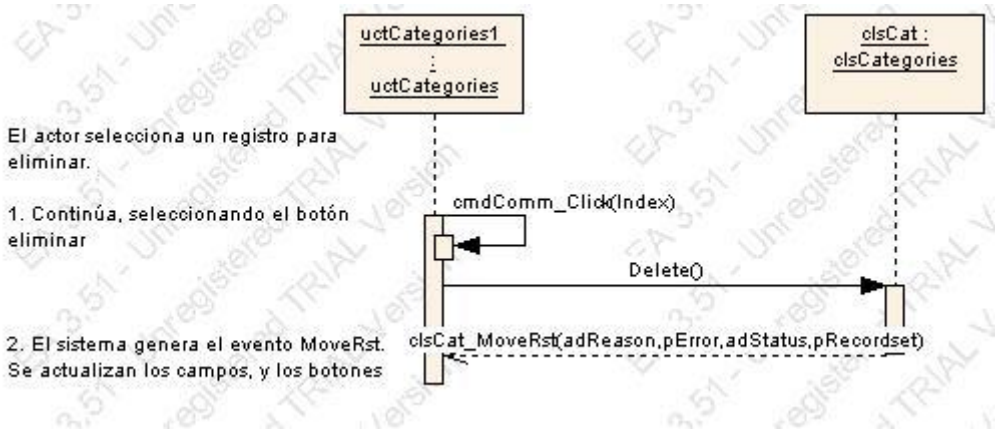


Fig. 15. Eliminar Categorías

Tabla 10. Eliminar Categorías Mensajes

ID	Mensaje	Del Objeto	Al Objeto	Notas
1	cmdComm_Click(Integer*)	uctCategories1	uctCategories1	El actor selecciona un registropara eliminar.
2	Delete()	uctCategories1	clsCat	Continúa, seleccionando el botón eliminar
3	clsCat_MoveRst(ADODB.EventReasonEnum, ADODB.Error, ADODB.EventStatusEnum*, ADODB.Recor	clsCat	uctCategories1	El sistema genera el evento MoveRst. Se actualizan los campos, y los botones

	dset)			

UCA24. Buscar Categorías

Tipo: *public* **Use Case**
Estado: Terminado. Versión 1.0. Fase 1.0.
Paquete: Subsistema Categorías

Objetivo:
- Busca las categorías en la persistencia

Escenarios

Buscar Categorías {Basic Path}.

1. El actor desea buscar información de categorías de productos. El sistema despliega el menú de búsqueda
2. El actor selecciona una de las opciones del menú
3. El sistema comienza la búsqueda de la información
4. Se cierra el recordset
5. Se genera el evento MoveRst del recordset. Se Actualizan los campos y los botones de comando (Nuevo, Agregar, Eliminar, etc)
6. Actualiza la grid de categorías
7. El sistema devuelve control al actor

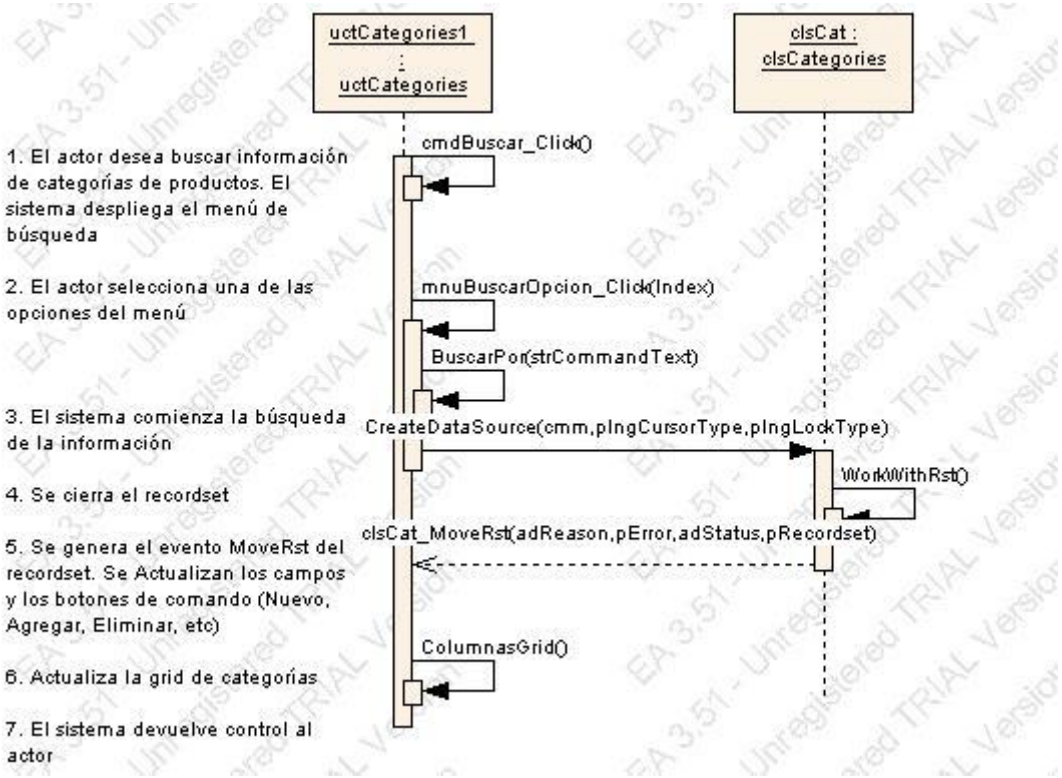


Fig. 16. Buscar Categorías

Tabla 11. Buscar Categorías Mensajes

ID	Mensaje	Del Objeto	Al Objeto	Notas
1	cmdBuscar_Click()	uctCategories1	uctCategories1	El actor desea buscar información de categorías de productos. El sistema despliega el menú de búsqueda
2	mnuBuscarOpcion_Click(Integer*)	uctCategories1	uctCategories1	El actor selecciona una de las opciones del menú
3	BuscarPor(String*)	uctCategories1	uctCategories1	Se crea la cadena de búsqueda
4	CreateDataSource(ADODB.Command*, ADODB.CursorTypeEnum*, ADODB.LockTypeEnum*)	uctCategories1	clsCat	El sistema comienza la búsqueda de la información
5	WorkWithRst()	clsCat	clsCat	Se crea el recordset
6	clsCat_MoveRst(ADODB.EventReasonEnum, ADODB.Error, ADODB.EventStatusEnum*, ADODB.Recordset)	clsCat	uctCategories1	Se genera el evento MoveRst del recordset. Se Actualizan los campos y los botones de comando (Nuevo, Agregar, Eliminar, etc)
7	ColumnasGrid()	uctCategories1	uctCategories1	Actualiza la grid de categorías

UCA37. Salvar Información

Tipo: *public* **Use Case**
Estado: Terminado. Versión 1.0. Fase 1.0.
Paquete: Subsistema Categorías

Objetivo:
- Salva la información, bien sea nueva o modificada por el actor

Escenarios

Salvar Información {Basic Path}.

El actor desea Salvar nueva categoría (Nueva o modificada)
1. El actor Selecciona Guardar
2. El sistema llama a actualizar

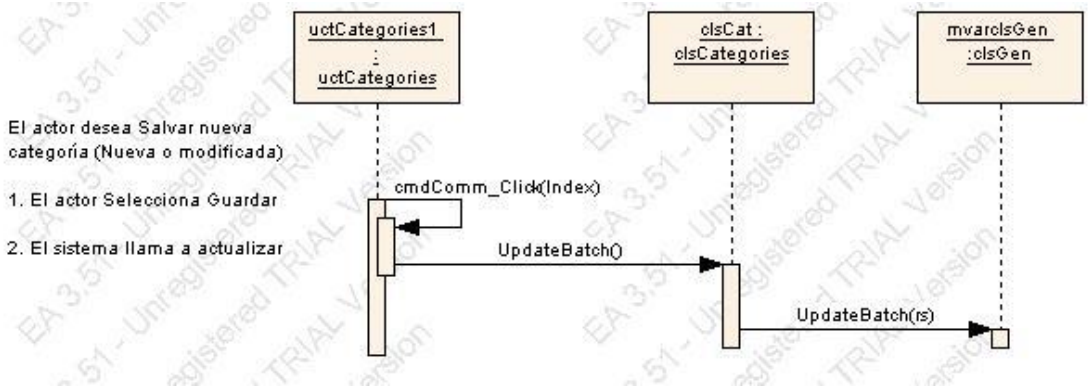


Fig. 17. Salvar Categorías

Tabla 12. Salvar Categorías Mensajes

ID	Mensaje	Del Objeto	Al Objeto	Notas
1	cmdComm_Click(Integer*)	uctCategories1	uctCategories1	El actor desea Salvar nueva categoría (Nueva o modificada). El actor Selecciona Guardar
2	UpdateBatch()	uctCategories1	clsCat	El sistema llama a actualizar
3	UpdateBatch(ADODB.Recordset*)	clsCat	mvarclsGen	Se llama al método que permite actualiza la persistencia

Subsistema Empleados

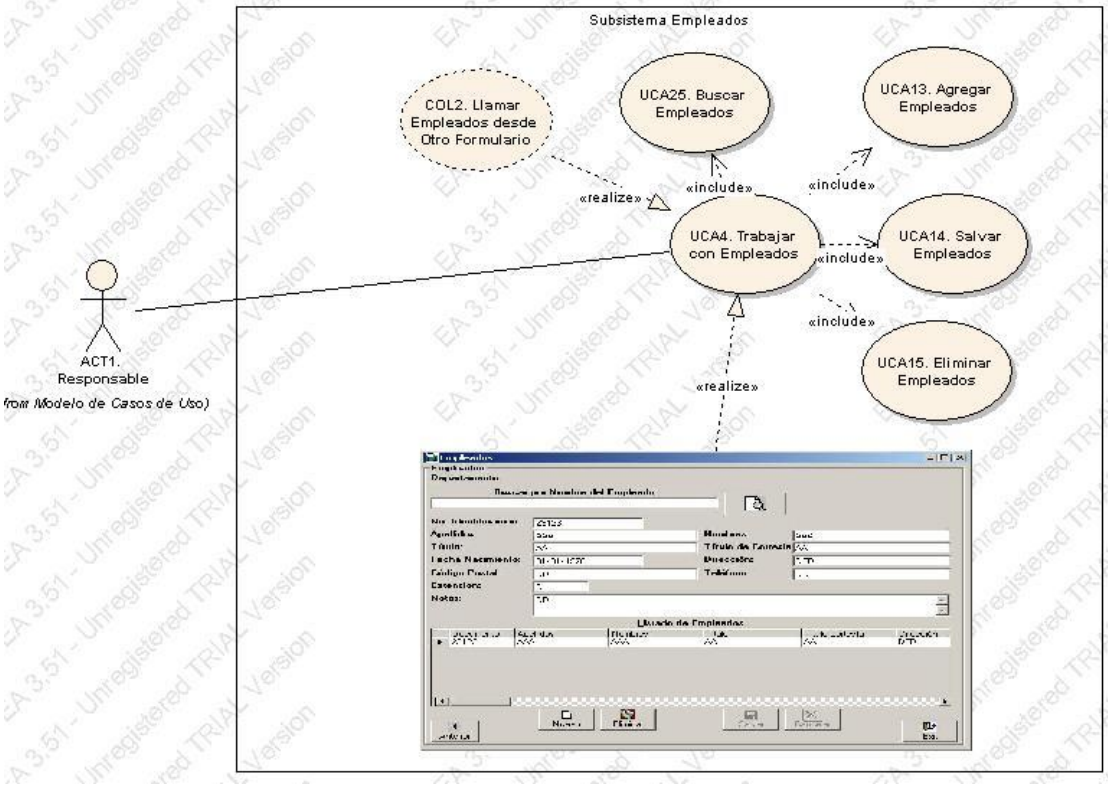


Fig. 18. Empleados

UCA4. Trabajar con Empleados

Tipo: *public* **Use Case**
Estado: Terminado. Versión 1.0. Fase 1.0.
Paquete: Subsistema Empleados

Trabajar con empleados incluye:
- Adicionar y/o modificar nuevos empleados
- Eliminar Empleados
- Buscar Empleados
- Salvar Empleados

Escenarios

Trabajar con empleados {Basic Path}.

1. El actor puede Agregar Empleados. Incluye UCA13. Agregar Empleados
2. El actor puede Eliminar empleados. Incluye UCA15. Eliminar Empleados
3. El actor puede Buscar empleados. Incluye UCA 25. Salvar Empleados
5. El actor puede Salvar la información nueva o modificada. Incluye UCA 14. Salvar Empleados

COL2. Llamar Empleados desde Otro Formulario

Tipo: *public* **Collaboration**
Estado: Terminado. Versión 1.0. Fase 1.0.
Paquete: Subsistema Empleados

Objetivo:
- Permitir al formulario de empleados, cooperar con otras clases que requieran de la información contenida en este para su funcionamiento

Escenarios

Llamar Empleados {Basic Path}.

1. El actor necesita llamar a los empleados desde otro punto de la aplicación
2. El sistema despliega el formulario de empleados

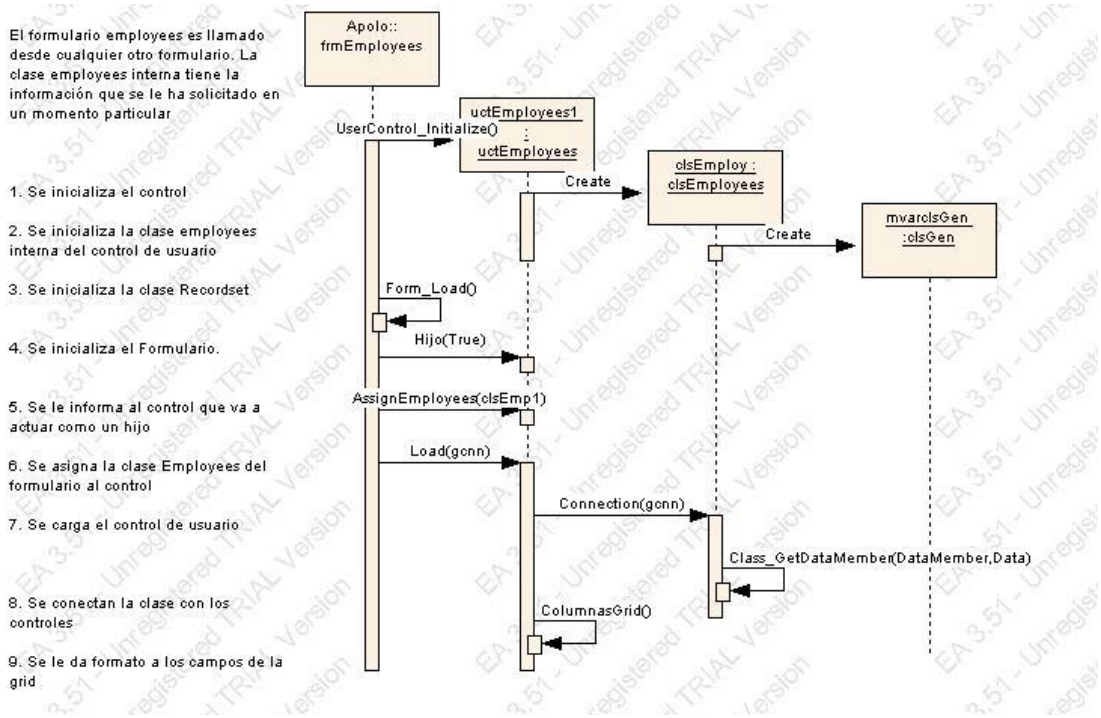


Fig. 19. Llamar Empleados desde otro Formulario

Tabla 13. Llamar Empleados desde otro Formulario Mensajes

ID	Mensaje	Del Objeto	Al Objeto	Notas
1	UserControl_Initialize()	frmEmployees	uctEmployees1	El formulario employees es llamado desde cualquier otro formulario. La clase employees interna tiene la información que se le ha

				solicitado en un momento particular. Se inicializa el control
2	Create	uctEmployees1	clsEmploy	Se inicializa la clase employees interna del control de usuario
3	Create	clsEmploy	mvarclsGen	Se inicializa la clase Recordset
4	Form_Load()	frmEmployees	frmEmployees	Se inicializa el Formulario.
5	Hijo(Boolean)	frmEmployees	uctEmployees1	Se le informa al control que va a actuar como un hijo
6	AssignEmployees(NorthDLL.clsEmployees*)	frmEmployees	uctEmployees1	Se asigna la clase Employees del formulario al control
7	Load(ADODB.Connection*)	frmEmployees	uctEmployees1	Se carga el control de usuario
8	Connection(ADODB.Connection*)	uctEmployees1	clsEmploy	Se asigna la conexión a la clase empleados
9	Class_GetDataMember(String*, Object*)	clsEmploy	clsEmploy	Se conecta la clase con los controles
10	ColumnasGrid()	uctEmployees1	uctEmployees1	Se le da formato a los campos

UCA13. Agregar Empleados

Tipo: *public* **Use Case**
Estado: Terminado. Versión 1.0. Fase 1.0.
Paquete: Subsistema Empleados

Objetivo:
- Agregar nuevos empleados al sistema

Escenarios

Agregar Empleados {Basic Path}.
El actor desea agregar un nuevo registro
1. El actor presiona el botón Nuevo
2. el sistema informa a la clase Empleado que debe iniciar un nuevo registro
3. El sistema dispone los controles y los botones para el ingreso y captura de la información
4. Una vez ingresados los datos, el sistema se dispone a salvar la información

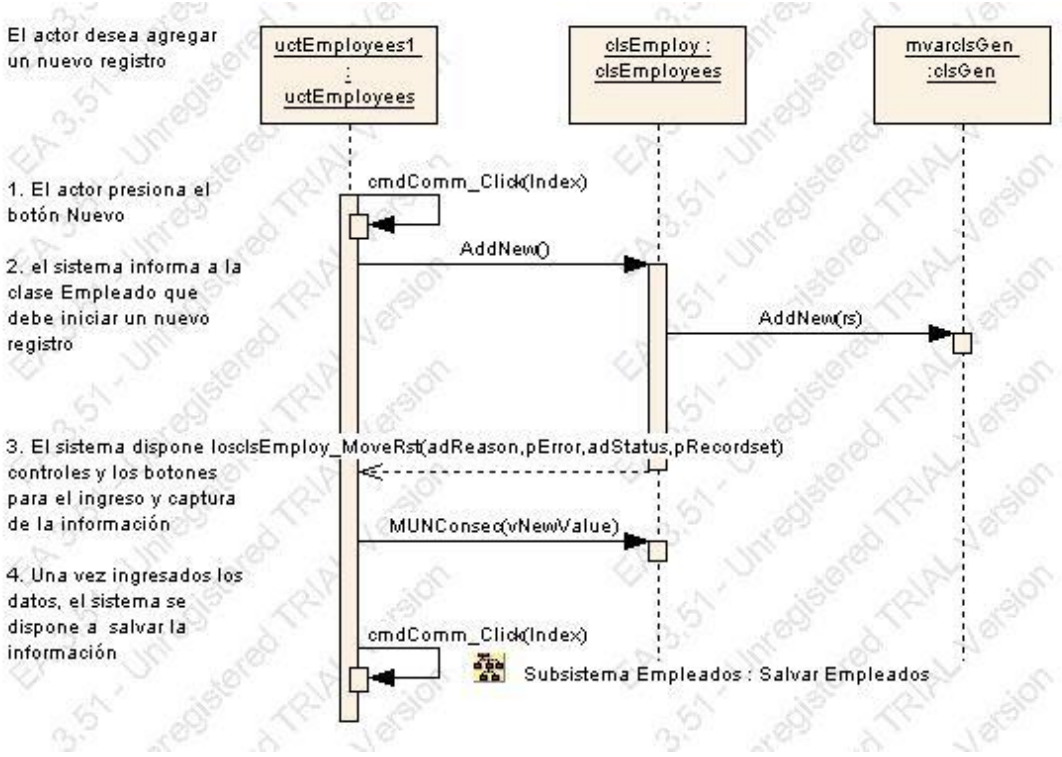


Fig. 20. Agregar Empleados

Tabla 14. Agregar Empleados Mensajes

ID	Mensaje	Del Objeto	Al Objeto	Notas
1	cmdComm_Click(Integer*)	uctEmployees1	uctEmployees1	El actor desea agregar un nuevo registro
2	AddNew()	uctEmployees1	clsEmploy	El actor presiona el botón Nuevo
3	AddNew(ADODB.Recordset*)	clsEmploy	mvarclsGen	el sistema informa a la clase Empleado que debe iniciar un nuevo registro
4	clsEmploy_MoveRst(ADODB.EventReasonEnum, ADODB.Error, ADODB.EventStatusEnum*, ADODB.Recordset)	clsEmploy	uctEmployees1	El sistema dispone los controles y los botones para el ingreso y captura de la información
5	MUNConsec(Long)	uctEmployees1	clsEmploy	Se asigna el municipio
6	cmdComm_Click(Integer*)	uctEmployees1	uctEmployees1	Una vez ingresados los datos, el sistema se dispone a salvar la información

UCA15. Eliminar Empleados

Tipo: *public* **Use Case**
Estado: Terminado. Versión 1.0. Fase 1.0.
Paquete: Subsistema Empleados

Objetivo:
- Eliminar registros de empleados que no se requieran y no estén relacionados con algún otro elemento del sistema

Escenarios

Eliminar Empleados {Basic Path}.
El actor selecciona un registro y procede a su eliminación
1. El actor selecciona el botón de eliminar
2. El sistema informa a la clase empleados interna que debe eliminar un registro
3. El sistema actualiza la información en la GUI

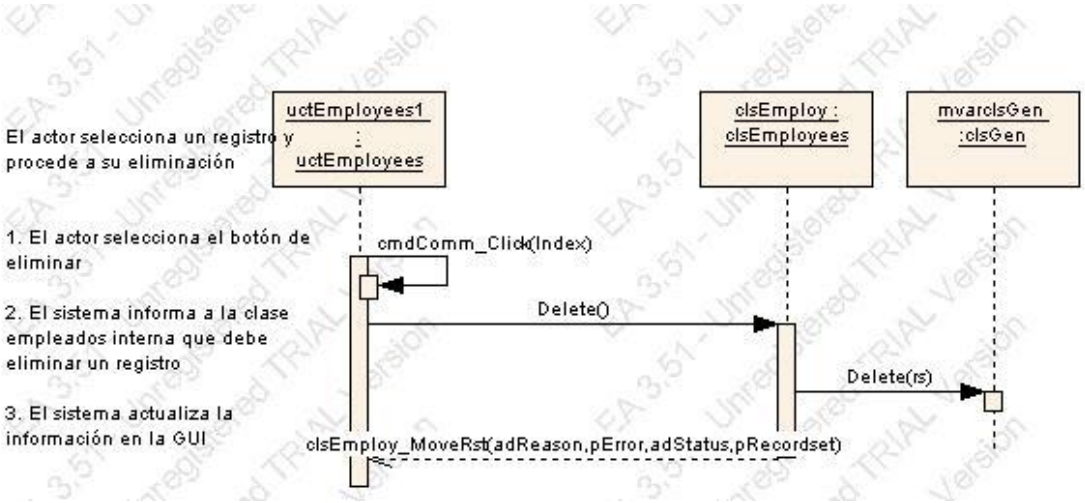


Fig. 21. Eliminar Empleados

Tabla 15. Eliminar Empleados Mensajes

ID	Mensaje	Del Objeto	Al Objeto	Notas
1	cmdComm_Click(Integer*)	uctEmployees1	uctEmployees1	El actor selecciona un registro y procede a su eliminación
2	Delete()	uctEmployees1	clsEmploy	El actor selecciona el botón de eliminar. El sistema informa a la clase empleados interna que debe eliminar un registro
3	Delete(ADODB.Recordset*)	clsEmploy	mvarclsGen	La clase controladora del recordset procede a eliminar el registro

4	clsEmploy_MoveRst(ADODB.EventReasonEnum, ADODB.Error, ADODB.EventStatusEnum*, ADODB.Recordset)	clsEmploy	uctEmployees1	El sistema actualiza la información en la GUI

UCA25. Buscar Empleados

Tipo: *public* **Use Case**
Estado: Terminado. Versión 1.0. Fase 1.0.
Paquete: Subsistema Empleados

Objetivo:
- Buscar a los empleados por algún criterio de búsqueda

Escenarios

Buscar Empleados {Basic Path}.
El actor va a buscar por los apellidos de los empleados
1. El actor ingresa el criterio de búsqueda. Selecciona el boton buscar
2. Se crea la cadena de búsqueda y se le asigna a la clase Empleados
3. La clase busca la información. Se genera el evento MoveRst. Se actualizan los controles y los botones de comando

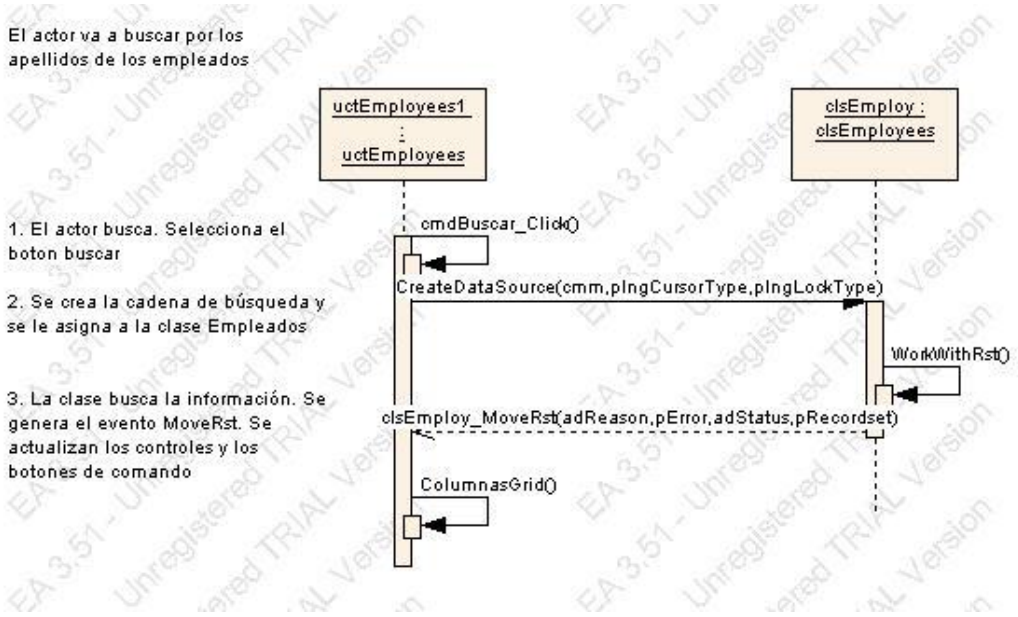


Fig. 22. Buscar Empleados

Tabla 16. Buscar Empleados Mensajes

ID	Mensaje	Del Objeto	Al Objeto	Notas
1	cmdBuscar_Click()	uctEmployees1	uctEmployees1	El actor va a buscar por los apellidos de los empleados. El actor busca. Selecciona el boton buscar
2	CreateDataSource(ADODB.Command*, ADODB.CursorTypeEnum*, ADODB.LockTypeEnum*)	uctEmployees1	clsEmploy	Se crea la cadena de búsqueda y se le asigna a la clase Empleados
3	WorkWithRst()	clsEmploy	clsEmploy	Se crea el recordset
4	clsEmploy_MoveRst(ADODB.EventReasonEnum, ADODB.Error, ADODB.EventStatusEnum*, ADODB.Recordset)	clsEmploy	uctEmployees1	La clase busca la información. Se genera el evento MoveRst. Se actualizan los controles y los botones de comando
5	ColumnasGrid()	uctEmployees1	uctEmployees1	Se actualiza la información de las columnas de la gris

UCA14. Salvar Empleados

Tipo: *public* **Use Case**
Estado: Terminado. Versión 1.0. Fase 1.0.
Paquete: Subsistema Empleados

Objetivo:
- salvar la información de los empleados (Nueva o modificada)

Escenarios

Salvar Empleados {Basic Path}.
El actor procede a salvar la información (nueva o modificada)
1. El actor oprime la tecla salvar
2. El sistema le informa a la clase que proceda a salvar la información
3. El sistema almacena la información
4. El sistema actualiza controles de edición y botones de comando

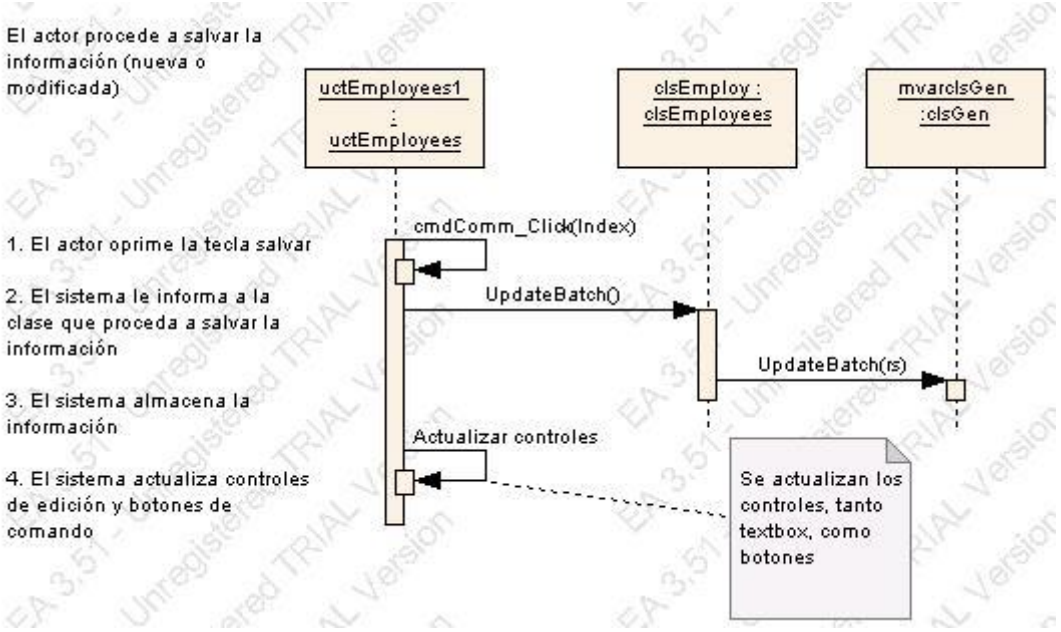


Fig. 23. Salvar Empleados

Tabla 17. Salvar Empleados Mensajes

ID	Mensaje	Del Objeto	Al Objeto	Notas
1	cmdComm_Click(Integer*)	uctEmployees1	uctEmployees1	El actor procede a salvar la información (nueva o modificada) El actor oprime la tecla salvar
2	UpdateBatch()	uctEmployees1	clsEmploy	El sistema le informa a la clase que proceda a salvar la información
3	UpdateBatch(ADODB.Recordset*)	clsEmploy	mvarclsGen	La clase comienza la actualización del lote. Se almacena la información
4	Actualizar controles	uctEmployees1	uctEmployees1	El sistema actualiza controles de edición y botones de comando

Subsistema Productos

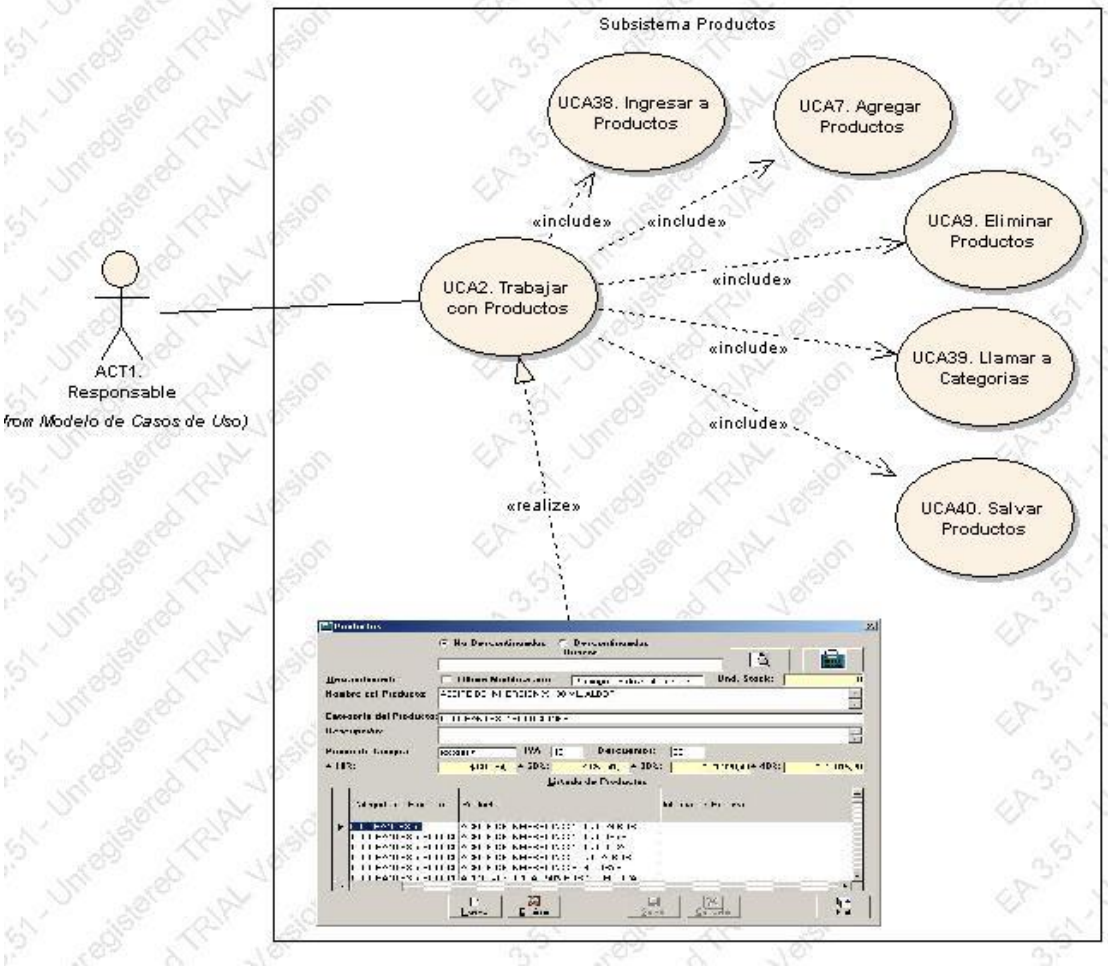


Fig. 24. Productos

UCA2. Trabajar con Productos

Tipo: *public* **Use Case**
Estado: Terminado. Versión 1.0. Fase 1.0.
Paquete: Subsistema Productos

Cuando se Trabaja con los productos es posible:
- Ingresar a la página de productos
- Agregar nuevos productos
- Eliminar productos
- Llamar a la categoria del producto para su modificación o adición

- Salvar los productos

UCA38. Ingresar a Productos

Tipo: *public* **Use Case**
Estado: Terminado. Versión 1.0. Fase 1.0.
Paquete: Subsistema Productos

Objetivo:
- Ingresar al formulario de productos para trabajar con el

Escenarios

Ingresar a Productos {Basic Path}.

1. El actor selecciona el menú de productos
2. Se inicializa el control de Productos
3. Se carga el formulario
6. Se carga la información a partir de la persistencia
7. Se despliega el formulario

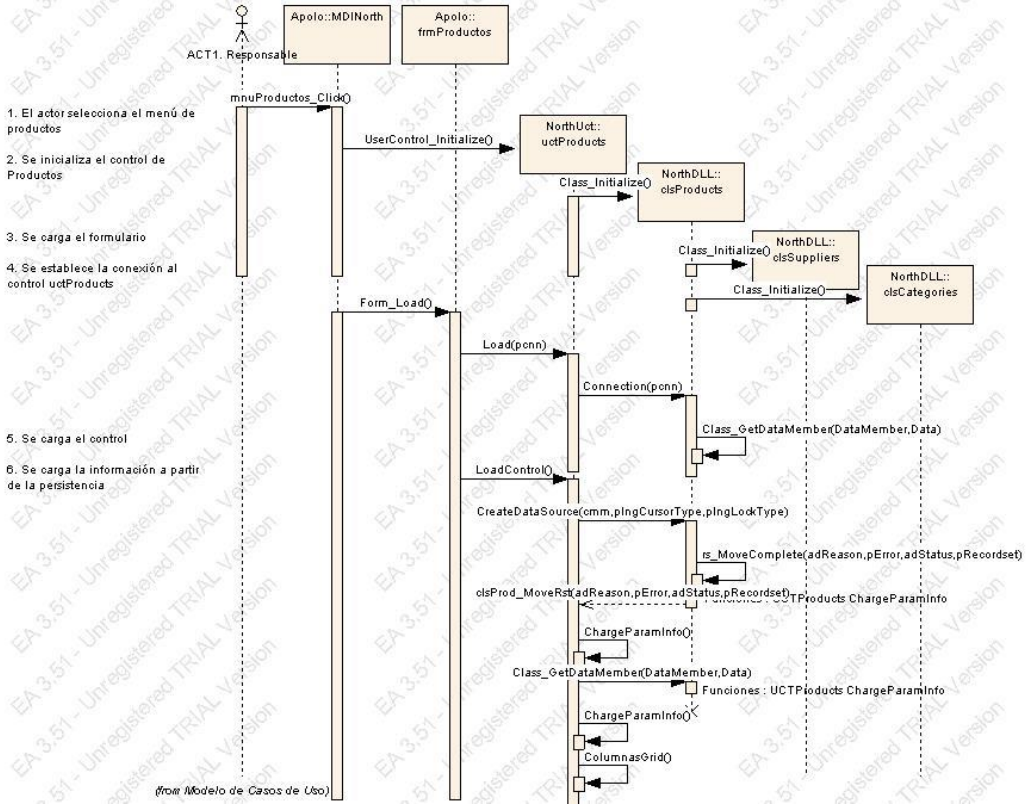


Fig. 25. Ingresar a Productos

Tabla 18. Ingresar a Productos. Mensajes

ID	Mensaje	Del Objeto	Al Objeto	Notas
1	mnuProductos _Click()	ACT1. Responsable	MDINorth	El actor selecciona el menu productos
2	UserControl_Initialize()	MDINorth	uctProducts	Se inicializa el control de productos
3	Class_Initialize()	uctProducts	clsProducts	Se inicializa la clase manejadora del control
4	Class_Initialize()	clsProducts	clsSuppliers	Se inicializa la clase de proveedores de la clase productos
5	Class_Initialize()	clsProducts	clsCategories	Se inicializa la clase categorías
6	Form_Load()	MDINorth	frmProductos	Se carga el formulario de productos
7	Load(ADODB.Connection*)	frmProductos	uctProducts	Se carga el control con la conexión y se inicializa el mismo
8	Connection(ADODB.Connection*)	uctProducts	clsProducts	Se asigna la conexión a la clase de productos
9	Class_GetDataMember(String*, Object*)	clsProducts	clsProducts	Se establece la relación entre la clase y los controles
10	LoadControl()	frmProductos	uctProducts	Se continúa con la carga del control
11	CreateDataSource(ADODB.Command*, ADODB.CursorTypeEnum*, ADODB.LockTypeEnum*)	uctProducts	clsProducts	Se comienza la búsqueda de registros por defecto
12	rs_MoveComplete(ADODB.EventReasonEnum, ADODB.Error, ADODB.EventStatusEnum*, ADODB.Recordset)	clsProducts	clsProducts	El recordset de la clase genera el evento
13	clsProd_MoveRst(ADODB.EventReasonEnum, ADODB.Error, ADODB.EventStatusEnum*, ADODB.Recordset)	clsProducts	uctProducts	El evento es escuchado por el control de productos
14	ChargeParamInfo()	uctProducts	uctProducts	Carga la información desde las tablas relacionadas con los productos
15	Class_GetDataMember(String*, Object*)	uctProducts	clsProducts	Clase dataSource. Ocurre cuando un consumidor de datos requiere una nueva fuente de datos

16	ChargeParamInfo()	uctProducts	uctProducts	Carga la información desde las tablas relacionadas con los productos
17	ColumnasGrid()	uctProducts	uctProducts	Procedimiento que permite el formateo de los títulos de las columnas

UCA39. Llamar a Categorias

Tipo: *public* **Use Case**
Estado: Terminado. Versión 1.0. Fase 1.0.
Paquete: Subsistema Productos

Objetivo:
- Llamar o configurar la categoría relacionada con el producto

Escenarios

Llamar a Categorías {Basic Path}.

1. El actor desea observar las categorías del producto (presiona una tecla en el campo de categorías)
2. Se inicializa la clase categorias con la información de la categoría
3. Se devuelve la clase creada al formulario principal, para que continúe la carga de la clase en el formulario categorías

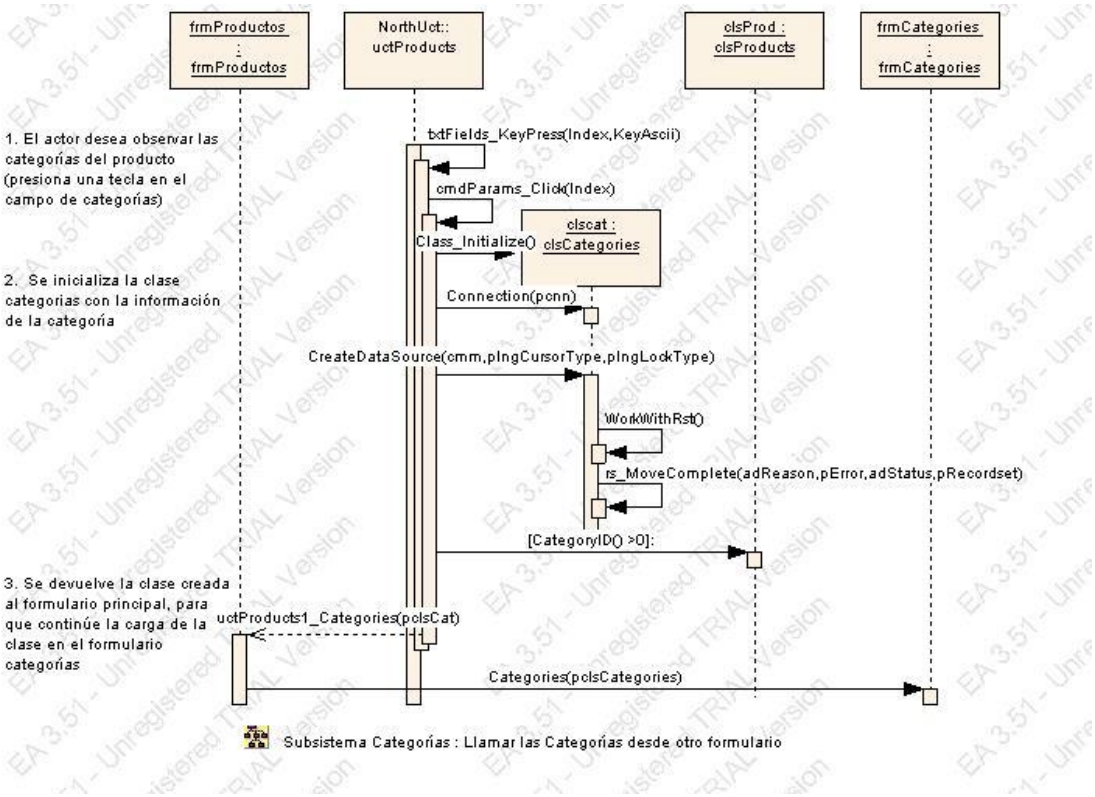


Fig. 26. Llamar a Categorías

Tabla 19. Llamar a Categorías Mensajes

ID	Mensaje	Del Objeto	Al Objeto	Notas
1	txtFields_KeyPress(Integer*, Integer*)	uctProducts	uctProducts	Evento KeyPress de los texbox vinculados con el datasource. Permite determinar la tecla presionada
2	cmdParams_Click(Integer*)	uctProducts	uctProducts	Permite la creación de un objeto datasource específico que debe ser desplegado, y enviado a su correspondiente control para permitir la selección de un elemento vinculado con este control
3	Class_Initialize()	uctProducts	Clscat	Inicializa el apuntador al recordset y crea el datamember de categorías
4	Connection(ADODB.Connection*)	uctProducts	Clscat	Procedimiento que permite assignar la conexión a la clase
5	CreateDataSource(ADODB.Command*, ADODB.CursorTypeEnum*, ADODB.LockTypeEnum*)	uctProducts	Clscat	Abre la conexión
6	WorkWithRst()	Clscat	clscat	Cierra y crea un nuevo recordset
7	rs_MoveComplete(ADODB.EventReasonEnum, ADODB.Error, ADODB.EventStatusEnum*, ADODB.Recordset)	Clscat	clscat	Evento llamado despues del cambio de posición del recordset
8		uctProducts	clsProd	Devuelve el id de la categoría
9	uctProducts1_Categories(NorthDLL.clsCategories*)	uctProducts	frmProductos	Evento que envía la clase clscategories
10	Categories(NorthDLL.clsCategories*)	frmProductos	frmCategories	Permite desplegar el formulario de categorias, asignandole la clase con la cual este trabaja internamente

UCA40. Salvar Productos

Tipo: *public* **Use Case**
Estado: Terminado. Versión 1.0. Fase 1.0.
Paquete: Subsistema Productos

Objetivo:
- Salvar los productos nuevos o modificados

Escenarios

Salvar Productos {Basic Path}.
1. El actor selecciona el botón salvar
2. el control informa a la clase que debe almacenar la información
3. Se calculan los precios del producto

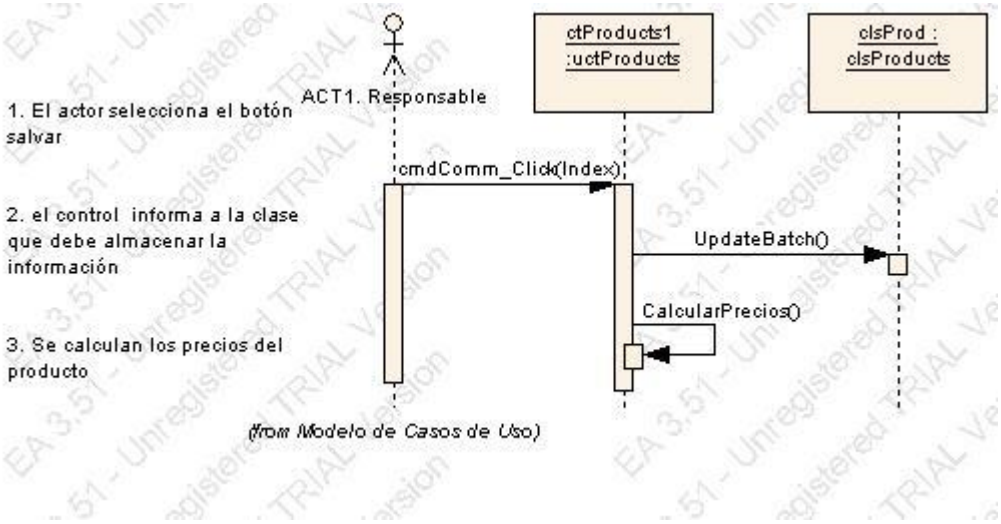


Fig. 27. Salvar Productos

Tabla 20. Salvar Productos Mensajes

ID	Mensaje	Del Objeto	Al Objeto	Notas
1	cmdComm_Click(Integer*)	ACT1. Responsable	ctProducts1	El actor selecciona el botón salvar
2	UpdateBatch()	ctProducts1	clsProd	el control informa a la clase que debe almacenar la información
3	CalcularPrecios()	ctProducts1	ctProducts1	Calcula y despliega los precios del producto

UCA7. Agregar Productos

Tipo: *public* **Use Case**
Estado: Terminado. Versión 1.0. Fase 1.0.
Paquete: Subsistema Productos

Objetivo:
- Agregar nuevos productos al sistema

Escenarios

Agregar Productos {Basic Path}.

1. El actor selecciona el botón nuevo
2. se carga la información necesaria a partir de la persistencia
3. Se calculan los precios

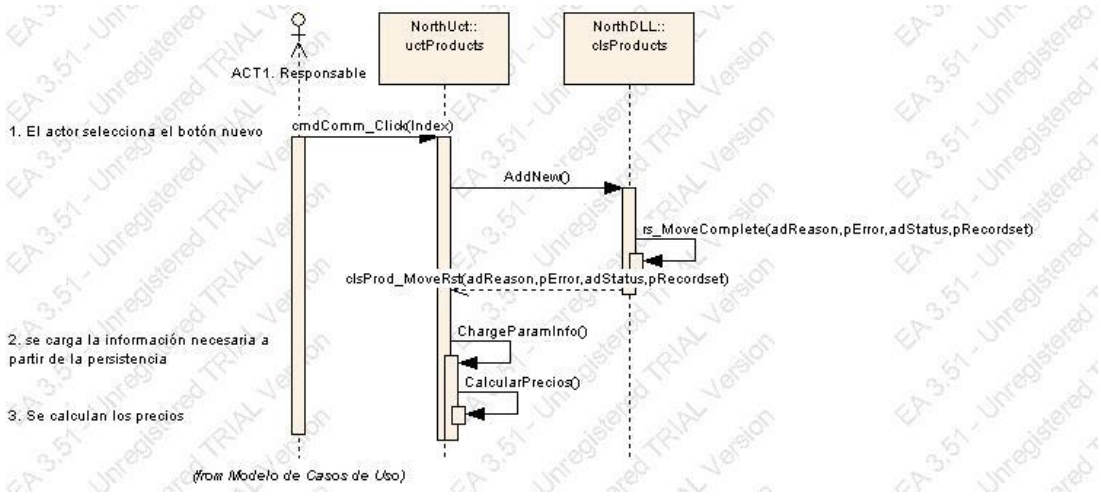


Fig. 28. Nuevo Producto

Tabla 21. Nuevo Producto Mensajes

ID	Mensaje	Del Objeto	Al Objeto	Notas
1	cmdComm_Click(Integer*)	ACT1. Responsable	uctProducts	El actor selecciona el botón nuevo
2	AddNew()	uctProducts	clsProducts	Crea un nuevo registro en el recordset
3	rs_MoveComplete(ADODB.EventReasonEnum, ADODB.Error, ADODB.EventStatusEnum*, ADODB.Recordset)	clsProducts	clsProducts	Evento llamado despues del cambio de posición del recordset
4	clsProd_MoveRst(ADODB.EventReasonEnum, ADODB.Error, ADODB.EventStatusEnum*, ADODB.Recordset)	clsProducts	uctProducts	El evento es llamado después de cambiar la posicion corriente en el recordset

5	ChargeParamInfo()	uctProducts	uctProducts	Carga la información desde las tablas relacionadas con los productos
6	CalcularPrecios()	uctProducts	uctProducts	Calcula y despliega los precios del producto

UCA9. Eliminar Productos

Tipo: *public* **Use Case**
Estado: Terminado. Versión 1.0. Fase 1.0.
Paquete: Subsistema Productos

Objetivo:
- Eliminar los productos que por una u otra razón el usuario así lo desea

Escenarios

Eliminar Producto {Basic Path}.
1. El actor selecciona el botón eliminar
2. Se le informa a la clase que proceda a la eliminación
3. Se carga la información necesaria a partir de la persistencia
4. Se formatean las columnas de la grid para la salida
5. Se calculan los precios

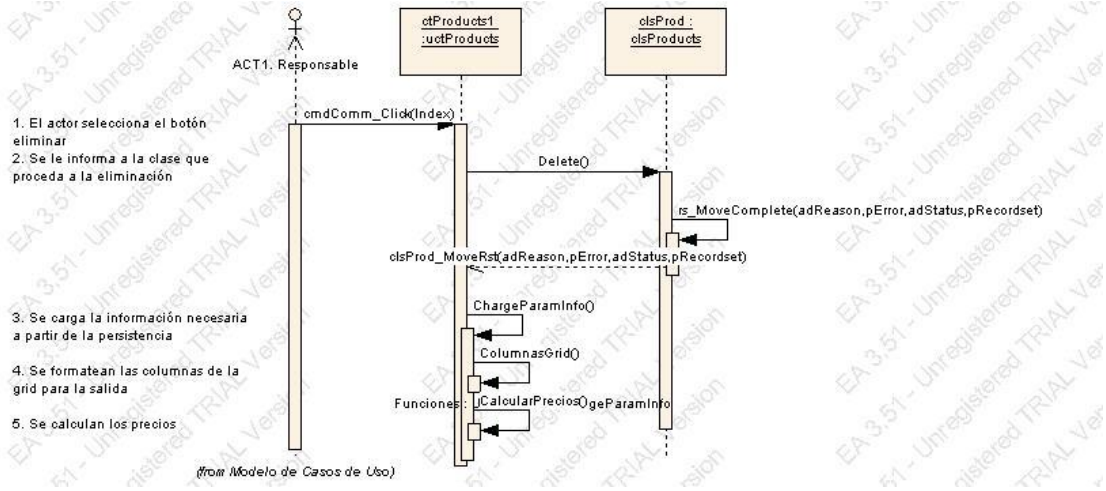


Fig. 29. Eliminar Productos

Tabla 22. Eliminar Productos Mensajes

ID	Mensaje	Del Objeto	Al Objeto	Notas
1	cmdComm_Click(Integer*)	ACT1. Responsable	ctProducts1	El actor selecciona el botón eliminar
2	Delete()	ctProducts1	clsProd	Procedimiento Eliminar del Recordset
3	rs_MoveComp	clsProd	clsProd	Evento llamado despues del cambio de posición

	lete(ADODB.EventReasonEnum, ADODB.Error, ADODB.EventStatusEnum*, ADODB.Recordset)			del recordset
4	clsProd_MoveRst(ADODB.EventReasonEnum, ADODB.Error, ADODB.EventStatusEnum*, ADODB.Recordset)	clsProd	ctProducts1	El evento es llamado después de cambiar la posicion corriente en el recordset
5	ChargeParamInfo()	ctProducts1	ctProducts1	Carga la información desde las tablas relacionadas con los productos
6	ColumnasGrid()	ctProducts1	ctProducts1	Procedimiento que permite el formateo de los títulos de las columnas
7	CalcularPrecios()	ctProducts1	ctProducts1	Calcula y despliega los precios del producto

Subsistema Proveedores

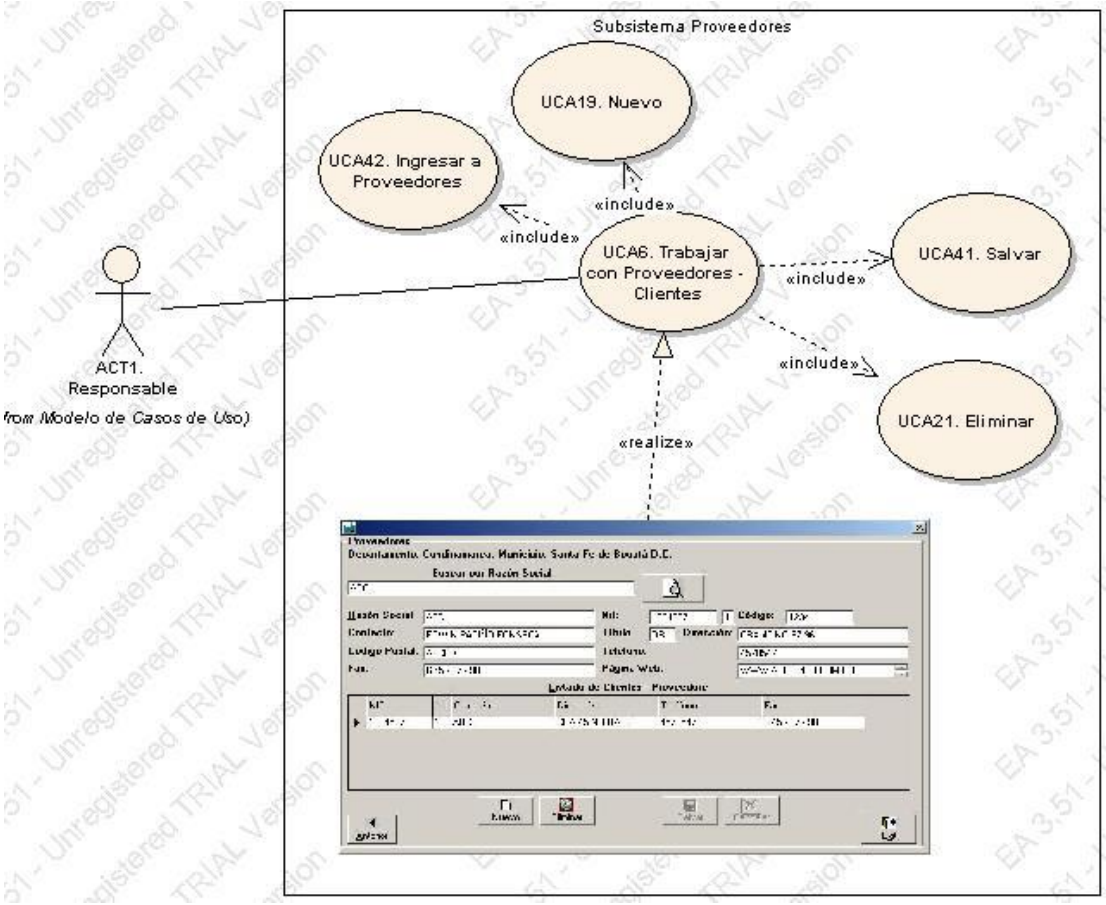


Fig. 30. Proveedores

UCA6. Trabajar con Proveedores - Clientes

Tipo: *public* **Use Case**
Estado: Terminado. Versión 1.0. Fase 1.0.
Paquete: Subsistema Proveedores

Cuando se Trabaja con los proveedores o los clientes, es posible:
- Agregar
- Salvar
- Eliminar

UCA42. Ingresar a Proveedores

Tipo: *public* **Use Case**
Estado: Terminado. Versión 1.0. Fase 1.0.
Paquete: Subsistema Proveedores

Objetivo:
- Describir el ingreso al formulario de proveedores (Clientes)

Escenarios

Ingresar a proveedores {Basic Path}.
1. Se carga el control
2. Se recuperan los datos requeridos
3. Se formatean las columnas de la grid

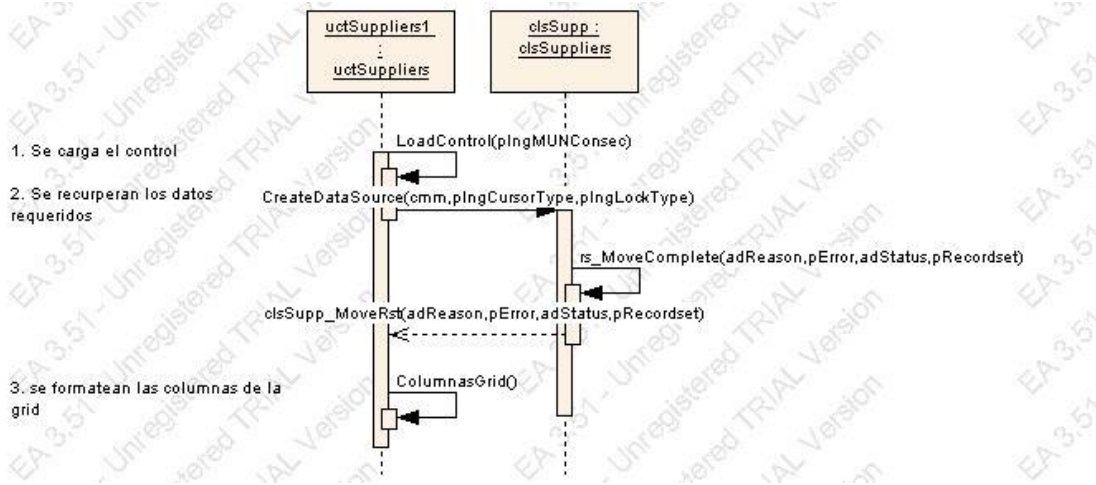


Fig. 31. Ingresar a proveedores

Tabla 23. Ingresar a proveedores Mensajes

ID	Mensaje	Del Objeto	Al Objeto	Notas
1	LoadControl(Long*)	uctSuppliers1	uctSuppliers1	Se carga el control
2	CreateDataSource(ADODB.Command*, ADODB.CursorTypeEnum*, ADODB.LockTypeEnum*)	uctSuppliers1	clsSupp	Abre la conexión
3	rs_MoveComplete(ADODB.EventReasonEnum, ADODB.Error, ADODB.EventStatusEnum*, ADODB.Recor	clsSupp	clsSupp	Evento llamado despues del cambio de posición del recordset

	dset)			
4	clsSupp_MoveRst(ADODB.EventReasonEnum, ADODB.Error, ADODB.EventStatusEnum*, ADODB.Recordset)	clsSupp	uctSuppliers1	El control escucha el Evento llamado despues del cambio de posición del recordset
5	ColumnasGrid()	uctSuppliers1	uctSuppliers1	Procedimiento que permite el formateo de los títulos de las columnas

UCA19. Nuevo

Tipo: *public* **Use Case**
Estado: Terminado. Versión 1.0. Fase 1.0.
Paquete: Subsistema Proveedores

Objetivo:
- Agregar nuevos proveedores (Clientes)

Escenarios

Nuevo Proveedor {Basic Path}.
1. El actor selecciona el botón nuevo
2. Se despliega el formulario listo para la adición del nuevo proveedor

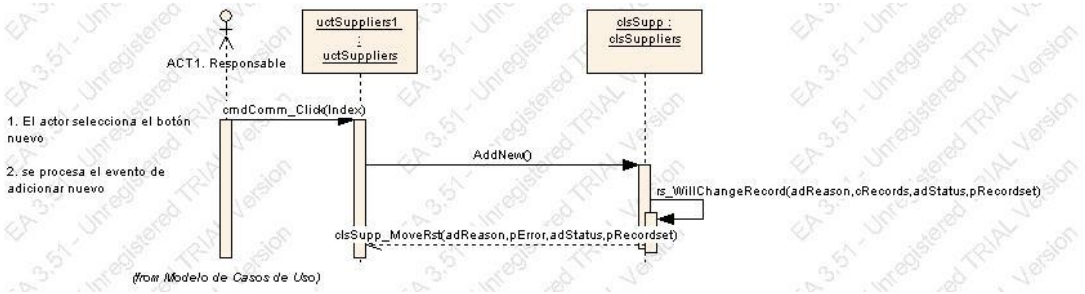


Fig. 32. Nuevo Proveedor

Tabla 24. Nuevo Proveedor Mensajes

ID	Mensaje	Del Objeto	Al Objeto	Notas
1	cmdComm_Click(Integer*)	ACT1. Responsable	uctSuppliers1	El actor selecciona el botón nuevo
2	AddNew()	uctSuppliers1	clsSupp	Crea un nuevo registro en el recordset
3	rs_WillChangeRecord(ADODB.EventReaso	clsSupp	clsSupp	Evento llamado despues del cambio de posición del recordset

	nEnum, Long, ADODB.EventStatusEnum*, ADODB.Recordset)			
4	clsSupp_MoveRst(ADODB.EventReasonEnum, ADODB.Error, ADODB.EventStatusEnum*, ADODB.Recordset)	clsSupp	uctSuppliers1	El control escucha el evento llamado después de cambiar la posicion corriente en el recordset

UCA21. Eliminar

Tipo: *public* **Use Case**
Estado: Terminado. Versión 1.0. Fase 1.0.
Paquete: Subsistema Proveedores

Objetivo:
- Eliminar información de proveedores (Clientes)

Escenarios

Eliminar {Basic Path}.
1. El actor selecciona eliminar proveedor
2. el control informa a la clase que debe eliminar el proveedor

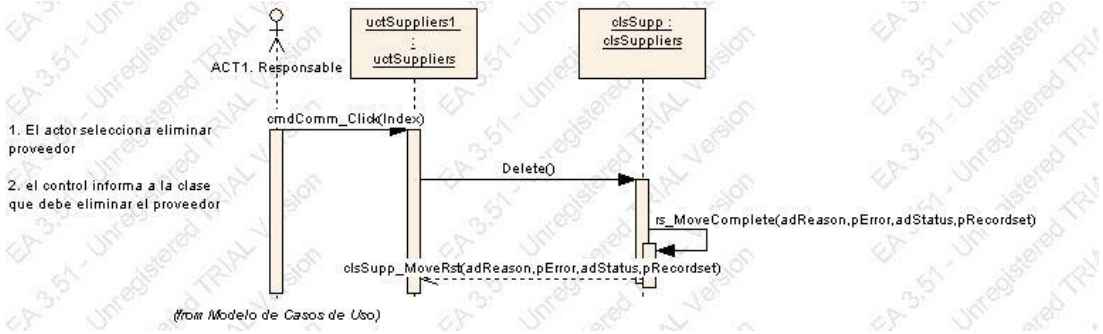


Fig. 33. Eliminar

Tabla 25. Eliminar Mensajes

ID	Mensaje	Del Objeto	Al Objeto	Notas
1	cmdComm_Click(Integer*)	ACT1. Responsable	uctSuppliers1	El actor selecciona eliminar proveedor
2	Delete()	uctSuppliers1	clsSupp	Procedimiento Eliminar del Recordset

3	rs_MoveComplete(ADODB.EventReasonEnum, ADODB.Error, ADODB.EventStatusEnum*, ADODB.Recordset)	clsSupp	clsSupp	Evento llamado despues del cambio de posición del recordset
4	clsSupp_MoveRst(ADODB.EventReasonEnum, ADODB.Error, ADODB.EventStatusEnum*, ADODB.Recordset)	clsSupp	uctSuppliers1	el control escucha el Evento llamado despues del cambio de posición del recordset

UCA41. Salvar

Tipo: *public* **Use Case**
Estado: Terminado. Versión 1.0. Fase 1.0.
Paquete: Subsistema Proveedores

Objetivo:
- Almacenar la información de forma persistente en la base de datos (nueva o cambiada)

Escenarios

Salvar {Basic Path}.
1. El actor selecciona Salvar proveedor
2. el control informa a la clase que debe salvar el proveedor

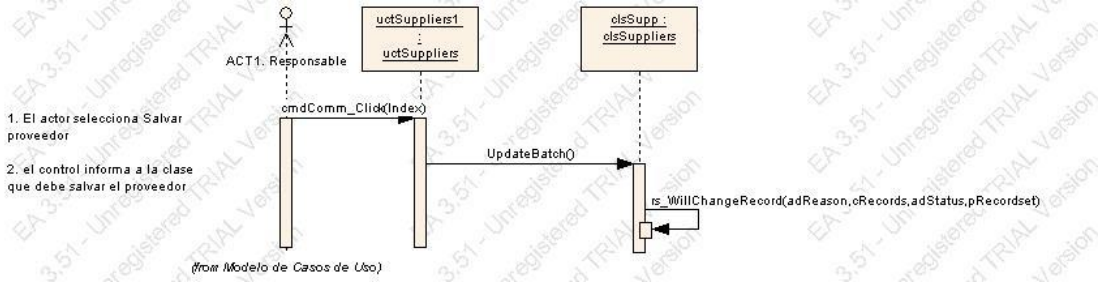


Fig. 34. Salvar Proveedor

Tabla 26. Salvar Proveedor Mensajes

ID	Mensaje	Del Objeto	Al Objeto	Notas
1	cmdComm_Click(Integer*)	ACT1. Responsable	uctSuppliers1	El actor selecciona Salvar proveedor
2	UpdateBatch()	uctSuppliers1	clsSupp	Procedimiento de actualización por lotes
3	rs_WillChangeRecord(ADODB.EventReasonEnum, Long, ADODB.EventStatusEnum*, ADODB.Recordset)	clsSupp	clsSupp	Evento llamado despues del cambio de posición del recordset

Subsistema Transportadores

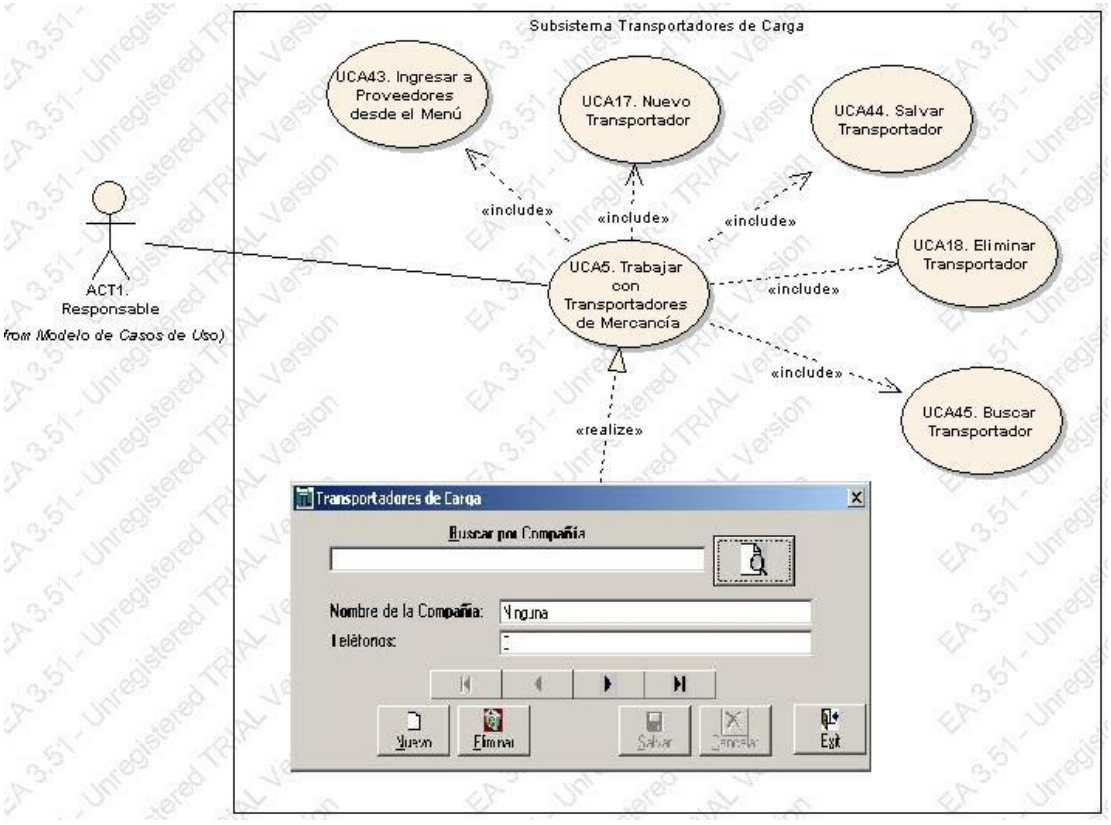


Fig. 35. Transportadores

UCA5. Trabajar con Transportadores de Mercancía

Tipo: *public* **Use Case**
Estado: Terminado. Versión 1.0. Fase 1.0.
Paquete: Subsistema Transportadores

Al trabajar con los transportadores de mercancia es posible:
- Agregar un nuevo transportador
- Salvar la información del transportador (nueva o modificada)
- Elimianr a un transportador
- Buscar información de un transportador

UCA17. Nuevo Transportador

Tipo: *public* **Use Case**
Estado: Terminado. Versión 1.0. Fase 1.0.
Paquete: Subsistema Transportadores

Objetivo:
- Agregar nuevos transportadores al sistema

Escenarios

Nuevo Transportador {Basic Path}.

1. El actor selecciona Nuevo proveedor
2. el control informa a la clase que debe Adicionar un proveedor

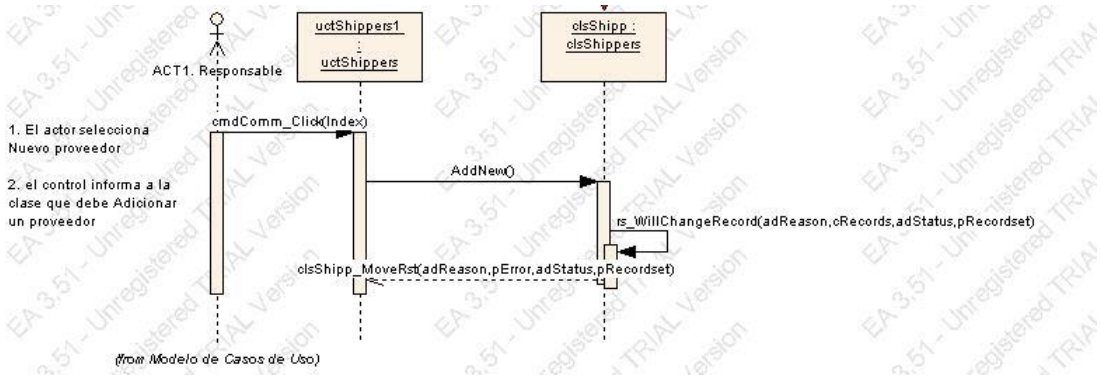


Fig. 36. Nuevo Transportador

Tabla 27. Nuevo Transportador Mensajes

ID	Mensaje	Del Objeto	Al Objeto	Notas
1	cmdComm_Click(Integer*)	ACT1. Responsable	uctShippers1	El actor selecciona Nuevo proveedor
2	AddNew()	uctShippers1	clsShipp	Crea un nuevo registro en el recordset
3	rs_WillChangeRecord(ADODB.EventReasonEnum, Long, ADODB.EventStatusEnum*, ADODB.Recordset)	clsShipp	clsShipp	Evento llamado despues del cambio de posición del recordset
4	clsShipp_MoveRst(ADODB.EventReasonEnum, ADODB.Error, ADODB.EventStatusEnum*, ADODB.Recordset)	clsShipp	uctShippers1	el control escucha el Evento llamado despues del cambio de posición del recordset

UCA18. Eliminar Transportador

Tipo: *public* **Use Case**
Estado: Terminado. Versión 1.0. Fase 1.0.
Paquete: Subsistema Transportadores

Objetivo:
- Eliminar la información de los transportadores

Escenarios

Eliminar Transportador {Basic Path}.
1. El actor selecciona Eliminar proveedor
2. el control informa a la clase que debe Eliminar el proveedor

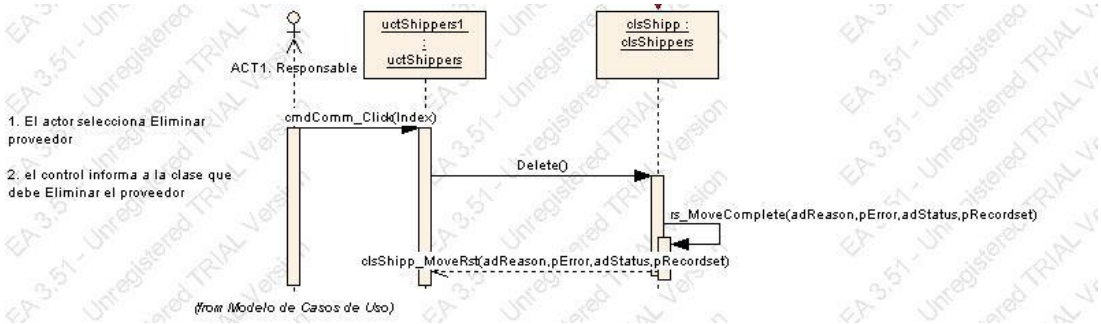


Fig. 37. Eliminar Transportador

Tabla 28. Eliminar Transportador Mensajes

ID	Mensaje	Del Objeto	Al Objeto	Notas
1	cmdComm_Click(Integer*)	ACT1. Responsable	uctShippers1	El actor selecciona Eliminar proveedor
2	Delete()	uctShippers1	clsShipp	Procedimiento Eliminar del Recordset
3	rs_MoveComplete(ADODB.EventReasonEnum, ADODB.Error, ADODB.EventStatusEnum*, ADODB.Recordset)	clsShipp	clsShipp	Evento llamado despues del cambio de posición del recordset
4	clsShipp_MoveRst(ADODB.EventReasonEnum, ADODB.Error, ADODB.EventStatusEnum*, ADODB.Recordset)	clsShipp	uctShippers1	el control escucha el Evento llamado despues del cambio de posición del recordset

UCA43. Ingresar a Transportadores desde el Menú

Tipo: *public* **Use Case**
Estado: Terminado. Versión 1.0. Fase 1.0.
Paquete: Subsistema Transportadores

Objetivo:
- Permitir el acceso del usuario al formulario de proveedores

Escenarios

Ingresar a Proveedores desde el Menú {Basic Path}.
1. El actor selecciona Ingresar a transportadores
2. Se llama al formulario de proveedores
3. se inicializa el control de proveedore, y su clase
4. Se carga el formulario
5. Se carga el control
6. Se carga la información de la persistencia

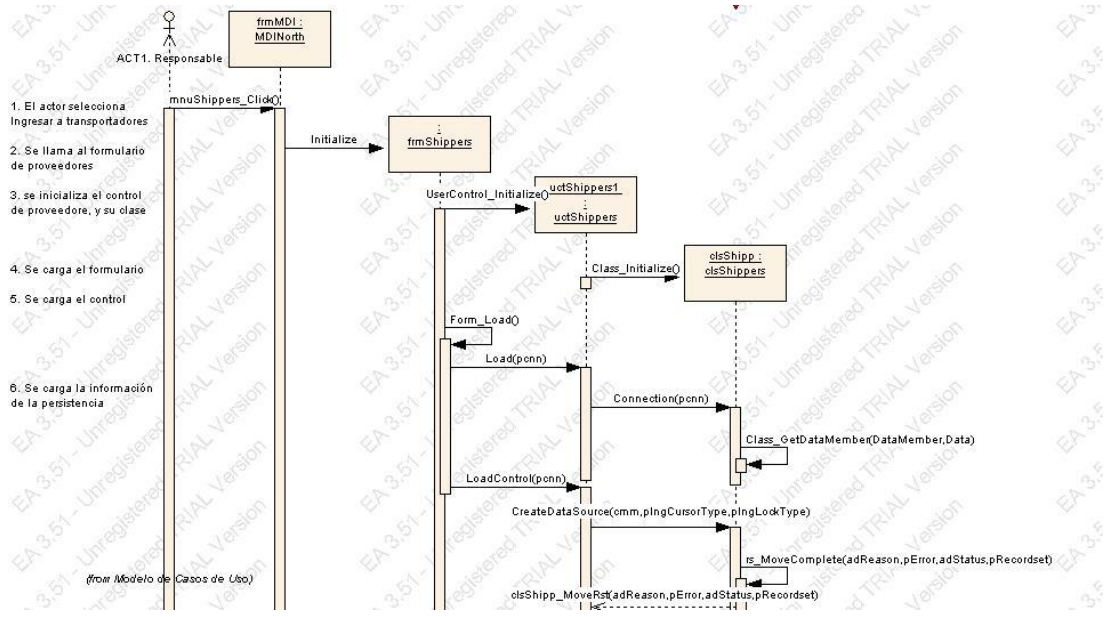


Fig. 38. Ingresar a Transportadores desde el Menú

Tabla 29. Ingresar a Transportadores desde el Menú Mensajes

ID	Mensaje	Del Objeto	Al Objeto	Notas
1	mnuShippers_ Click()	ACT1. Responsable	frmMDI	El actor selecciona Ingresar a transportadores
2	Initialize	frmMDI		Se llama al formulario de proveedores
3	UserControl_I nitialize()		uctShippers1	Se inicializa el control de proveedore, y su clase
4	Class_Initialize	uctShippers1	clsShipp	Se inicializa la clase proveedores

	()			
5	Form_Load()			Se carga el formulario
6	Load(ADODB.Connection*)		uctShippers1	Se carga el control
7	Connection(ADODB.Connection*)	uctShippers1	clsShipp	Se asigna la conexión a la clase datasource
8	Class_GetDataMember(String*, Object*)	clsShipp	clsShipp	Clase dataSource. Ocurre cuando un consumidor de datos requiere una nueva fuente de datos
9	LoadControl(ADODB.Connection*)		uctShippers1	Permite crear el datasource
10	CreateDataSource(ADODB.Command*, ADODB.CursorTypeEnum*, ADODB.LockTypeEnum*)	uctShippers1	clsShipp	Abre la conexión
11	rs_MoveComplete(ADODB.EventReasonEnum, ADODB.Error, ADODB.EventStatusEnum*, ADODB.Recordset)	clsShipp	clsShipp	Evento llamado despues del cambio de posición del recordset
12	clsShipp_MoveRst(ADODB.EventReasonEnum, ADODB.Error, ADODB.EventStatusEnum*, ADODB.Recordset)	clsShipp	uctShippers1	el control escucha el Evento llamado despues del cambio de posición del recordset

UCA44. Salvar Transportador

Tipo: *public* **Use Case**
Estado: Terminado. Versión 1.0. Fase 1.0.
Paquete: Subsistema Transportadores

Objetivo:
- Salvar la información nueva o modificada en la persistencia

Escenarios

Salvar {Basic Path}.

1. El actor selecciona el botón salvar
2. El control llama a la clase para salvar la información

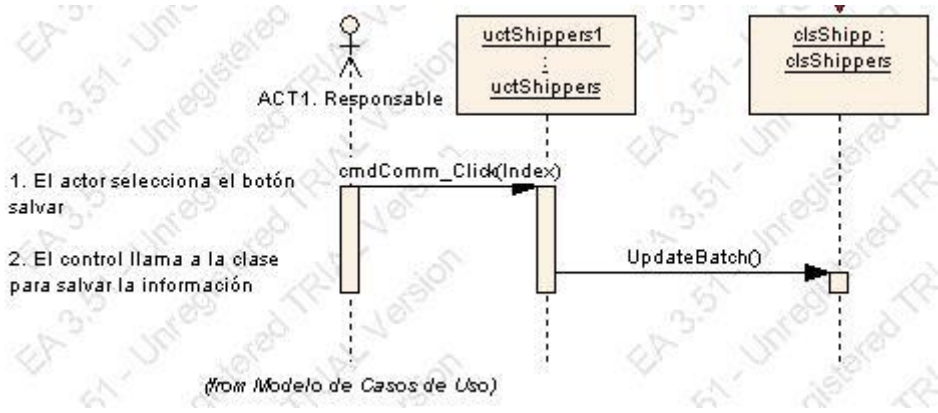


Fig. 39. Salvar Transportador

Tabla 30. Salvar Transportador Mensajes

ID	Mensaje	Del Objeto	Al Objeto	Notas
1	cmdComm_Click(Integer*)	ACT1. Responsable	uctShippers1	El actor selecciona el botón salvar
2	UpdateBatch()	uctShippers1	clsShipp	Procedimiento de actualización por lotes

UCA45. Buscar Transportador

Tipo: *public* **Use Case**
Estado: Terminado. Versión 1.0. Fase 1.0.
Paquete: Subsistema Transportadores

Objetivo:
- Buscar un transportador determinado

Escenarios

Buscar Transportador {Basic Path}.
1. El actor selecciona el botón Buscar
2. El control llama a la clase para buscar la información

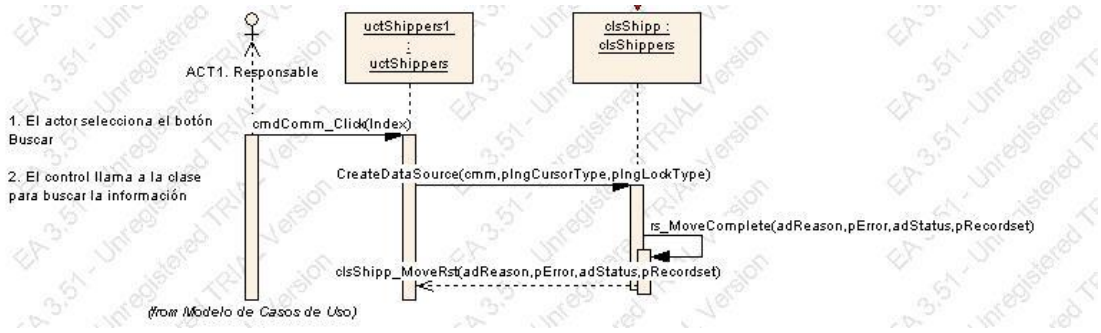


Fig. 40. Buscar

Tabla 31. Buscar Mensajes

ID	Mensaje	Del Objeto	Al Objeto	Notas
1	cmdComm_Click(Integer*)	ACT1. Responsable	uctShippers1	El actor selecciona el botón Buscar
2	CreateDataSource(ADODB.Command*, ADODB.CursorTypeEnum*, ADODB.LockTypeEnum*)	uctShippers1	clsShipp	Abre la conexión y efectúa la busqueda de registros en la persistencia
3	rs_MoveComplete(ADODB.EventReasonEnum, ADODB.Error, ADODB.EventStatusEnum*, ADODB.Recordset)	clsShipp	clsShipp	Evento llamado despues del cambio de posición del recordset
4	clsShipp_MoveRst(ADODB.EventReasonEnum, ADODB.Error, ADODB.EventStatusEnum*, ADODB.Recordset)	clsShipp	uctShippers1	el control escucha el Evento llamado despues del cambio de posición del recordset

Compras de Mercancía

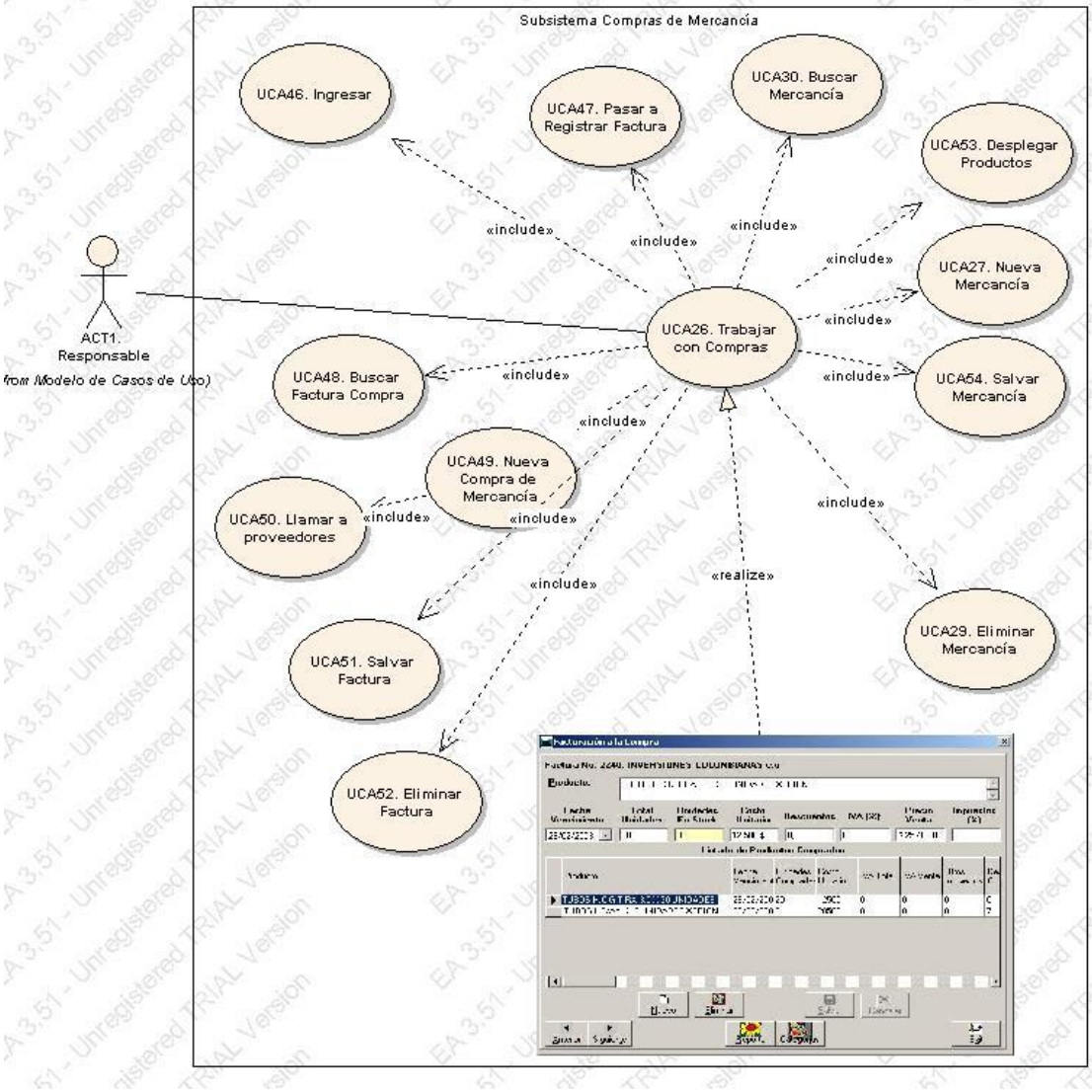


Fig. 41. Compras de Mercancía

UCA26. Trabajar con Compras

Tipo: *public* **Use Case**
Estado: Terminado. Versión 1.0. Fase 1.0.
Paquete: Compras de Mercancía

Al trabajar con las compras, es posible:
- Adicionar una nueva compra
- Modificar o actualizar una compra

- Buscar la información de una compra
- Registrar detalles de la compra
- Actualizar detalles de la compra
- Eliminar detalles de la compra

UCA27. Nueva Mercancía

Tipo: *public* **Use Case**
Estado: Terminado. Versión 1.0. Fase 1.0.
Paquete: Compras de Mercancía

Objetivo:
- Adicionar un nuevo producto a la compra

Escenarios

Nueva Mercancía {Basic Path}.
1. El actor selecciona Nuevo
2. El sistema llama a la clase para proceder a almacenar la información
3. Se carga la información de la factura

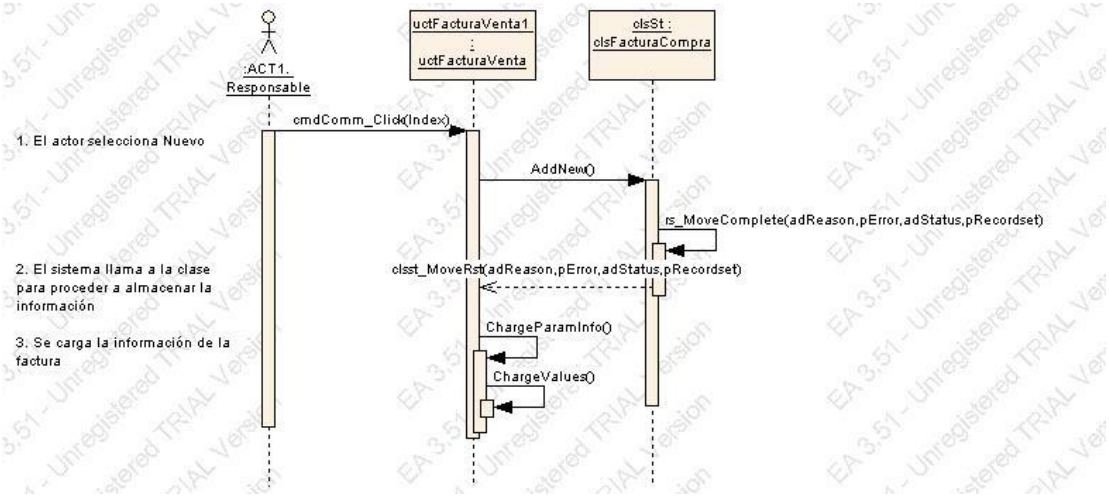


Fig. 42. Nueva Mercancía

Tabla 32. Nueva Mercancía Mensajes

ID	Mensaje	Del Objeto	Al Objeto	Notas
1	cmdComm_Click(Integer*)		uctFacturaVenta1	El actor selecciona Nuevo
2	AddNew()	uctFacturaVenta1	clsSt	Crea un nuevo registro en el recordset
3	rs_MoveComp	clsSt	clsSt	Evento llamado despues del cambio de posición

	lete(ADODB.EventReasonEnum, ADODB.Error, ADODB.EventStatusEnum*, ADODB.Recordset)			del recordset
4	clsst_MoveRst (ADODB.EventReasonEnum, ADODB.Error, ADODB.EventStatusEnum*, ADODB.Recordset)	clsSt	uctFacturaVenta1	el control escucha el Evento llamado despues del cambio de posición del recordset
5	ChargeParamInfo()	uctFacturaVenta1	uctFacturaVenta1	Busca la información del precio y actualiza los controles relacionados con los mismos
6	ChargeValues()	uctFacturaVenta1	uctFacturaVenta1	Permite cargar los calores de fecha de vencimiento y unidades en stock

UCA29. Eliminar Mercancía

Tipo: *public* **Use Case**
Estado: Terminado. Versión 1.0. Fase 1.0.
Paquete: Compras de Mercancía

Objetivo:
- Eliminar un producto de la compra

Escenarios

Eliminar Productos {Basic Path}.
1. El actor selecciona Eliminar
2. El sistema llama a la clase para proceder a eliminar la información
3. Se carga la información de la factura
4. Se carga la información de la grid

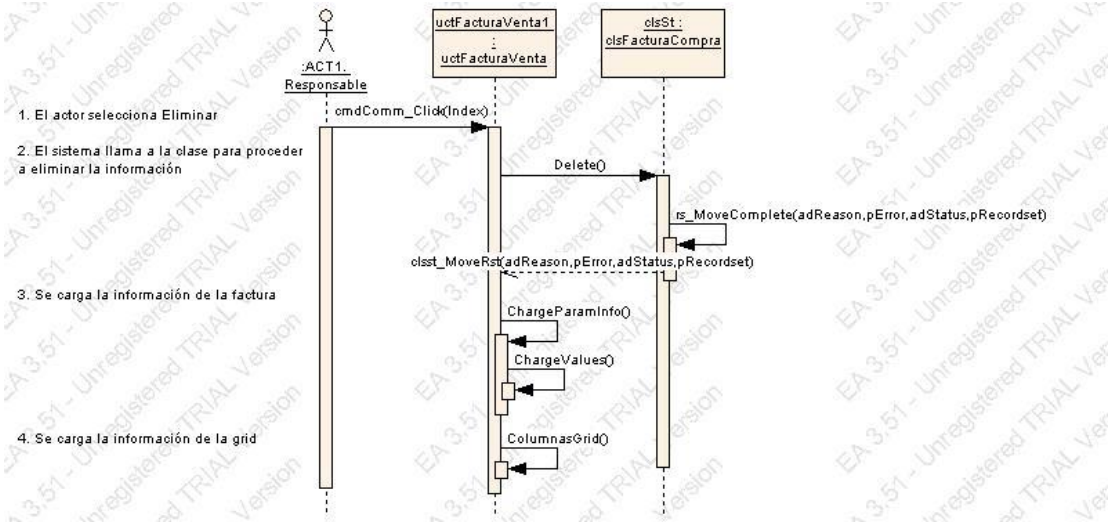


Fig. 43. Eliminar Mercancía

Tabla 33. Eliminar Mercancía Mensajes

ID	Mensaje	Del Objeto	Al Objeto	Notas
1	cmdComm_Click(Integer*)		uctFacturaVenta1	El actor selecciona Eliminar
2	Delete()	uctFacturaVenta1	clsSt	Procedimiento Eliminar del Recordset
3	rs_MoveComplete(ADODB.EventReasonEnum, ADODB.Error, ADODB.EventStatusEnum*, ADODB.Recordset)	clsSt	clsSt	Evento llamado despues del cambio de posición del recordset
4	clsst_MoveRst(ADODB.EventReasonEnum, ADODB.Error, ADODB.EventStatusEnum*, ADODB.Recordset)	clsSt	uctFacturaVenta1	el control escucha el Evento llamado despues del cambio de posición del recordset
5	ChargeParamInfo()	uctFacturaVenta1	uctFacturaVenta1	Busca la información del precio y actualiza los controles relacionados con los mismos
6	ChargeValues()	uctFacturaVenta1	uctFacturaVenta1	Permite cargar los calores de fecha de vencimiento y unidades en stock
7	ColumnasGrid()	uctFacturaVenta1	uctFacturaVenta1	Procedimiento que permite el formateo de los títulos de las columnas

UCA30. Buscar Mercancía

Tipo: *public* **Use Case**
Estado: Terminado. Versión 1.0. Fase 1.0.
Paquete: Compras de Mercancía

Objetivo:
- Buscar los productos relacionados con una factura

Escenarios

Buscar Productos {Basic Path}.
El actor desea buscar una información del detalle de la factura

1. Se busca el id de la factura
2. Se recurpera el número del detalle la factura
3. Se busca la información del detalle la factura
4. se despliega el detalle de la factura

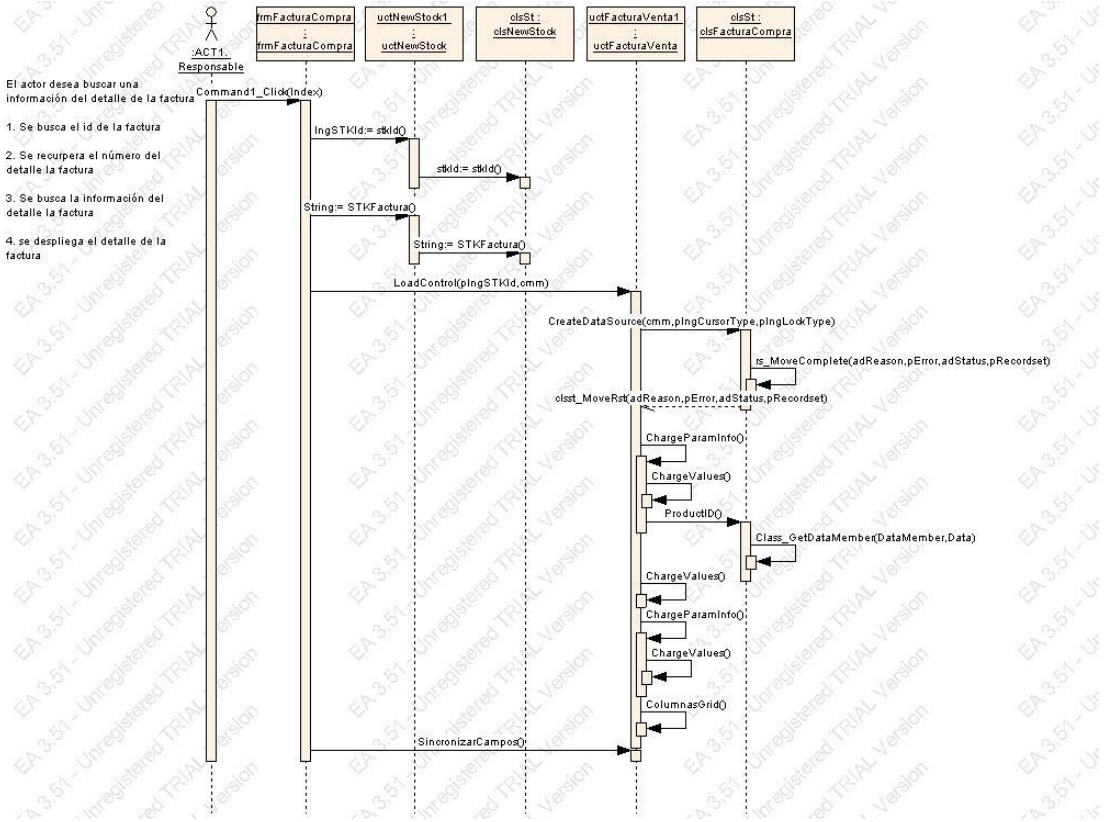


Fig. 44. UctFacturaVenta.ChargeValues

Tabla 34. UctFacturaVenta.ChargeValues Mensajes

ID	Mensaje	Del Objeto	Al Objeto	Notas
1	Command1_Click(Integer*)		frmFacturaCompra	
2	stkId()	frmFacturaCompra	uctNewStock1	Devuelve el id del stock
3	stkId()	uctNewStock1	clsSt	Devuelve el id del stock
4	STKFactura()	frmFacturaCompra	uctNewStock1	Devuelve el número de la factura
5	STKFactura()	uctNewStock1	clsSt	Devuelve el número de la factura
6	LoadControl(Long*, ADODB.Command*)	frmFacturaCompra	uctFacturaVenta1	
7	CreateDataSource(ADODB.Command*, ADODB.CursorTypeEnum*, ADODB.LockTypeEnum*)	uctFacturaVenta1	clsSt	Abre la conexión y efectúa la busqueda de registros en la persistencia
8	rs_MoveComplete(ADODB.EventReasonEnum, ADODB.Error, ADODB.EventStatusEnum*, ADODB.Recordset)	clsSt	clsSt	Evento llamado despues del cambio de posición del recordset
9	clsst_MoveRst(ADODB.EventReasonEnum, ADODB.Error, ADODB.EventStatusEnum*, ADODB.Recordset)	clsSt	uctFacturaVenta1	el control escucha el Evento llamado despues del cambio de posición del recordset
10	ChargeParamInfo()	uctFacturaVenta1	uctFacturaVenta1	Busca la información del precio y actualiza los controles relacionados con los mismos
11	ChargeValues()	uctFacturaVenta1	uctFacturaVenta1	Permite cargar los calores de fecha de vencimiento y unidades en stock
12	ProductID()	uctFacturaVenta1	clsSt	Devuelve el Id del Producto
13	Class_GetDataMember(String*, Object*)	clsSt	clsSt	Clase dataSource. Ocurre cuando un consumidor de datos requiere una nueva fuente de datos
14	ChargeValues()	uctFacturaVenta1	uctFacturaVenta1	Permite cargar los calores de fecha de vencimiento y unidades en stock
15	ChargeParamInfo()	uctFacturaVenta1	uctFacturaVenta1	Busca la información del precio y actualiza los controles relacionados con los mismos
16	ChargeValues()	uctFacturaVenta1	uctFacturaVenta1	Permite cargar los calores de fecha de vencimiento y unidades en stock
17	ColumnasGrid	uctFacturaVent	uctFacturaVent	Procedimiento que permite el formateo de los

	()	a1	a1	títulos de las columnas
18	SincronizarCampos()	frmFacturaCompra	uctFacturaVenta1	

UCA46. Ingresar

Tipo: *public* **Use Case**
Estado: Terminado. Versión 1.0. Fase 1.0.
Paquete: Compras de Mercancía

Objetivo:
- Permitir al usuario ingresar a las compras de mercancia

Escenarios

Ingresar a compras {Basic Path}.
El actor ingresa a la página de facturación a la compra

1. El actor selecciona el menu que Llama al formulario de factura de compra
2. Se inicializa el formulario
3. Se inicializa el control de factura
4. se inicializa la clase del control de la factura
5. Se inicializa el contro de detalle de la factura
6. Se inicializa la clase del detalle de la factura
7. Se carga el control
8. Se asigna la conexión al control de detalle de la factura, y a su clase asociada
9. Se asigna la conexión al control de factura de compra, y a su clase asociada

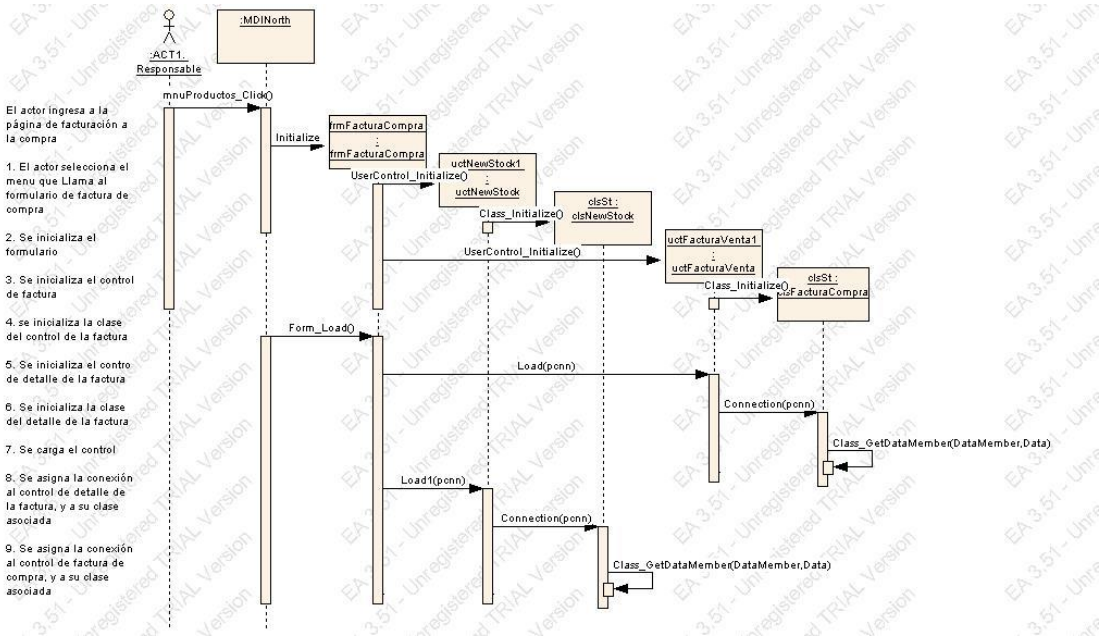


Fig. 45. Ingresar

Tabla 35. Ingresar Mensajes

ID	Mensaje	Del Objeto	Al Objeto	Notas
1	mnuProductos_Click()			Llama al formulario de factura de compra
2	Initialize		frmFacturaCompra	Se inicializa el formulario de factua de compra
3	UserControl_Initialize()	frmFacturaCompra	uctNewStock1	Se inicializa el control
4	Class_Initialize()	uctNewStock1	clsSt	Se inicializa la clase
5	UserControl_Initialize()	frmFacturaCompra	uctFacturaVenta1	Se inicializa el control de detalle de la factura
6	Class_Initialize()	uctFacturaVenta1	clsSt	Se inicializa la clase asociada al control de factura de venta
7	Form_Load()		frmFacturaCompra	Carga el formulario, asigna las imágenes
8	Load(ADODB.Connection*)	frmFacturaCompra	uctFacturaVenta1	Evento público de carga del control, el cual puede ser llamado para asignar los objetos internos del control
9	Connection(ADODB.Connection*)	uctFacturaVenta1	clsSt	Procedimiento que permite assignar la conexión a la clase
10	Class_GetDataMember(String*, Object*)	clsSt	clsSt	Clase dataSource. Ocurre cuando un consumidor de datos requiere una nueva fuente de datos
11	Load1(ADODB.Connection*)	frmFacturaCompra	uctNewStock1	Método que permite la carga del control, al ser llamado por el formulario
12	Connection(ADODB.Connection*)	uctNewStock1	clsSt	Procedimiento que permite assignar la conexión a la clase
13	Class_GetDataMember(String*, Object*)	clsSt	clsSt	Clase dataSource. Ocurre cuando un consumidor de datos requiere una nueva fuente de datos

UCA47. Pasar a Registrar Factura

Tipo: *public* **Use Case**
Estado: Terminado. Versión 1.0. Fase 1.0.
Paquete: Compras de Mercancía

Objetivo:
- Permitir al actor registrar la factura de compra

Escenarios

Pasar a "Registrar Factura" {Basic Path}.
El actor desea ingresar una nueva factura

- El actor pasa a siguiente

- El sistema asigna el año
- Se carga el command del control de factura con la información requerida para la búsqueda de la información
- se busca la información
- Se despliega el formulario

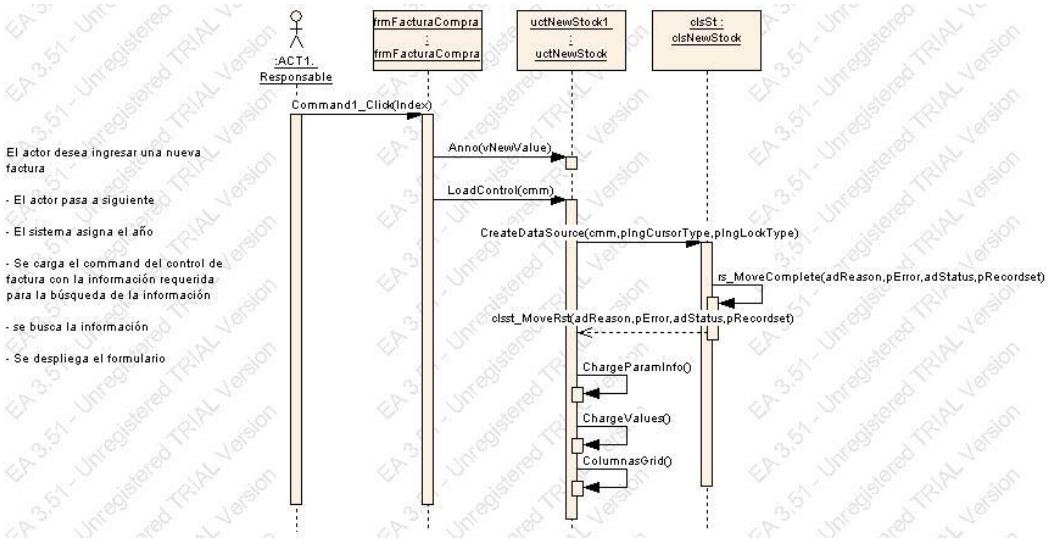


Fig. 46. Pasar a Registrar Factura

Tabla 36. Pasar a Registrar Factura Mensajes

ID	Mensaje	Del Objeto	Al Objeto	Notas
1	Command1_Click(Integer*)		frmFacturaCompra	El actor desea ingresar una nueva factura
2	Anno(Long)	frmFacturaCompra	uctNewStock1	Asigna el año
3	LoadControl(ADODB.Command*)	frmFacturaCompra	uctNewStock1	Procedimiento que permite la carga de la clase datasource
4	CreateDataSource(ADODB.Command*, ADODB.CursorTypeEnum*, ADODB.LockTypeEnum*)	uctNewStock1	clsSt	Abre la conexión y efectúa la busqueda de registros en la persistencia
5	rs_MoveComplete(ADODB.EventReasonEnum, ADODB.Error, ADODB.EventStatusEnum*, ADODB.Recordset)	clsSt	clsSt	Evento llamado despues del cambio de posición del recordset

6	clsst_MoveRst (ADODB.EventReasonEnum, ADODB.Error, ADODB.EventStatusEnum*, ADODB.Recordset)	clsSt	uctNewStock1	el control escucha el Evento llamado despues del cambio de posición del recordset
7	ChargeParamInfo()	uctNewStock1	uctNewStock1	Actualiza la información en los controles vinculados
8	ChargeValues()	uctNewStock1	uctNewStock1	Asigna los valores de la clase datasource a los controles
9	ColumnasGrid()	uctNewStock1	uctNewStock1	Procedimiento que permite el formateo de los títulos de las columnas

UCA48. Buscar Factura Compra

Tipo: *public* **Use Case**
Estado: Terminado. Versión 1.0. Fase 1.0.
Paquete: Compras de Mercancía

Objetivo:
- Permitir al actor buscar una factura de compra

Escenarios

Buscar Factura de Compra {Basic Path}.
El actor desea buscar una factura de compra

1. El actor ingresa la información a buscar
2. El actor hace click en buscar
3. El control llama a su clase asociada, para que esta busque la información en la persistencia
4. Se despliega la información en la pantalla

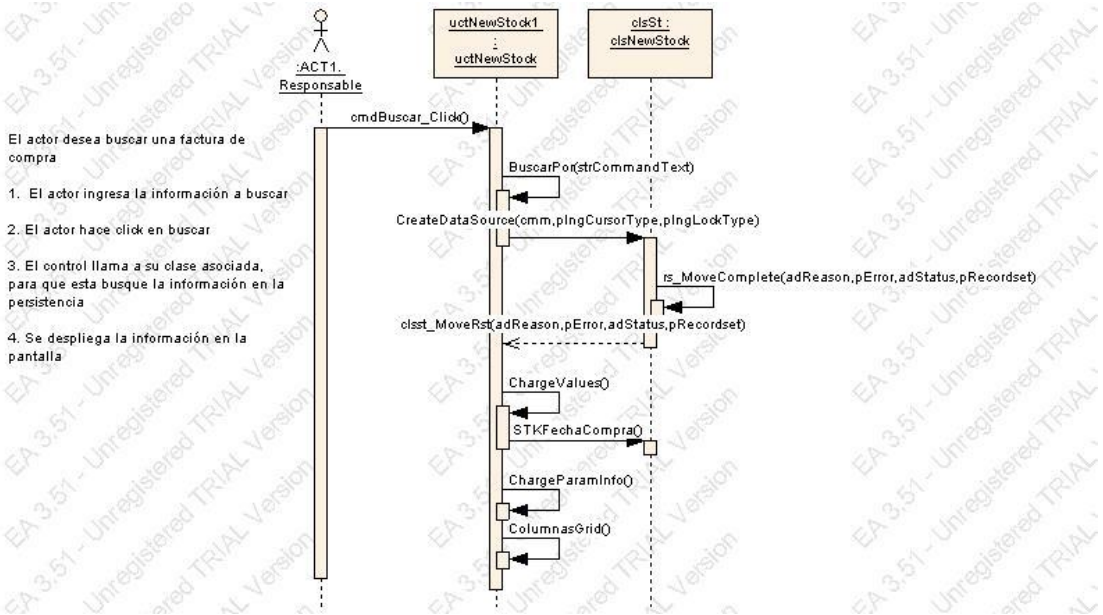


Fig. 47. Buscar Factura de Compra

Tabla 37. Buscar Factura de Compra Mensajes

ID	Mensaje	Del Objeto	Al Objeto	Notas
1	cmdBuscar_Click()		uctNewStock1	Despliega el menú de búsqueda
2	BuscarPor(String*)	uctNewStock1	uctNewStock1	Procedimiento que permite buscar por un criterio específico en la base de datos
3	CreateDataSource(ADODB.Command*, ADODB.CursorTypeEnum*, ADODB.LockTypeEnum*)	uctNewStock1	clsSt	Abre la conexión y efectúa la busqueda de registros en la persistencia
4	rs_MoveComplete(ADODB.EventReasonEnum, ADODB.Error, ADODB.EventStatusEnum*, ADODB.Recordset)	clsSt	clsSt	Evento llamado despues del cambio de posición del recordset
5	clsst_MoveRst(ADODB.EventReasonEnum, ADODB.Error, ADODB.EventStatusEnum*, ADODB.Recor	clsSt	uctNewStock1	el control escucha el Evento llamado despues del cambio de posición del recordset

	dset)			
6	ChargeValues( )	uctNewStock1	uctNewStock1	Asigna los valores de la clase datasource a los controles
7	STKFechaCo mpra()	uctNewStock1	clsSt	Devuelve la información de la fecha de compra
8	ChargeParamI nfo()	uctNewStock1	uctNewStock1	Actualiza la información en los controles vinculados
9	ColumnasGrid ()	uctNewStock1	uctNewStock1	Procedimiento que permite el formateo de los títulos de las columnas

UCA49. Nueva Compra de Mercancía

Tipo: *public* **Use Case**
Estado: Terminado. Versión 1.0. Fase 1.0.
Paquete: Compras de Mercancía

Objetivo:
- Permitir al actor adicionar una nueva factura de compra

Escenarios

Nueva compra de productos {Basic Path}.
El actor desea agregar una nueva compra de mercancía

1. El actor hace click en el botón nuevo
2. Se ingresan los valores de la factura

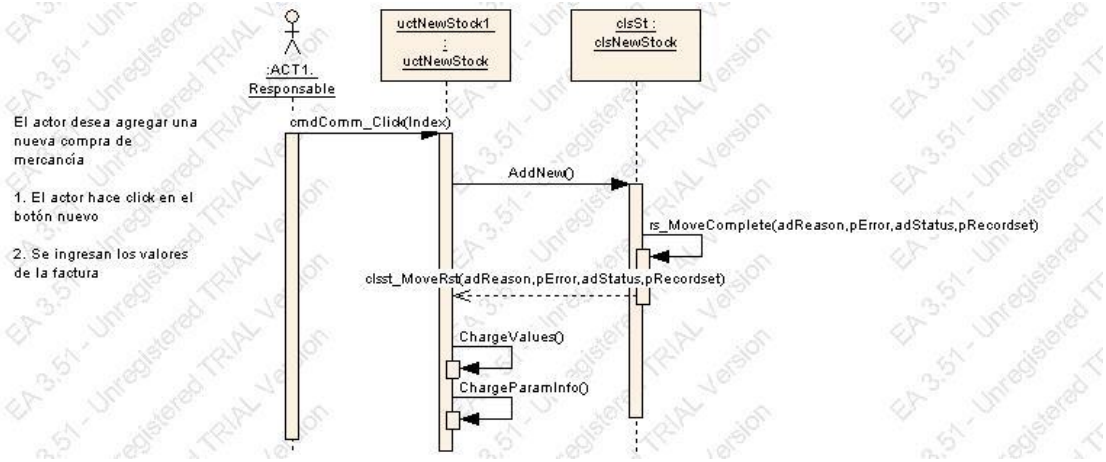


Fig. 48. Nueva Compra de Mercancía

Tabla 38. Nueva Compra de Mercancía Mensajes

ID	Mensaje	Del Objeto	Al Objeto	Notas
1	cmdComm_Cli		uctNewStock1	Procedimiento que permite llamar a las rutinas

	ck(Integer*)			de actualización - inserción, eliminación, adición o cancelación
2	AddNew()	uctNewStock1	clsSt	Crea un nuevo registro en el recordset
3	rs_MoveComplete(ADODB.EventReasonEnum, ADODB.Error, ADODB.EventStatusEnum*, ADODB.Recordset)	clsSt	clsSt	Evento llamado despues del cambio de posición del recordset
4	clsst_MoveRst(ADODB.EventReasonEnum, ADODB.Error, ADODB.EventStatusEnum*, ADODB.Recordset)	clsSt	uctNewStock1	el control escucha el Evento llamado despues del cambio de posición del recordset
5	ChargeValues()	uctNewStock1	uctNewStock1	Asigna los valores de la clase datasource a los controles
6	ChargeParamInfo()	uctNewStock1	uctNewStock1	Actualiza la información en los controles vinculados

UCA50. Llamar a proveedores

Tipo: *public* **Use Case**
Estado: Terminado. Versión 1.0. Fase 1.0.
Paquete: Compras de Mercancía

Objetivo:
- Permitir llamar a los proveedores desde el formulario de compra de mercancias

Escenarios

llamar a proveedores {Basic Path}.
El actor desea asignar el proveedor

1. El actor selecciona el campo de proveedore
2. El actor ingresa una letra cualquiera
3. se inicializa la clase de proveedores
4. Se busca la información del proveedor (si esta existe)
5. El formulario principal escucha el evento del control, el cual trae consigo la clase proveedor
6. Se crea el formulario proveedor
7. Se asigna la clase proveedor al formulario.
8. Se despliega el formulario proveedor

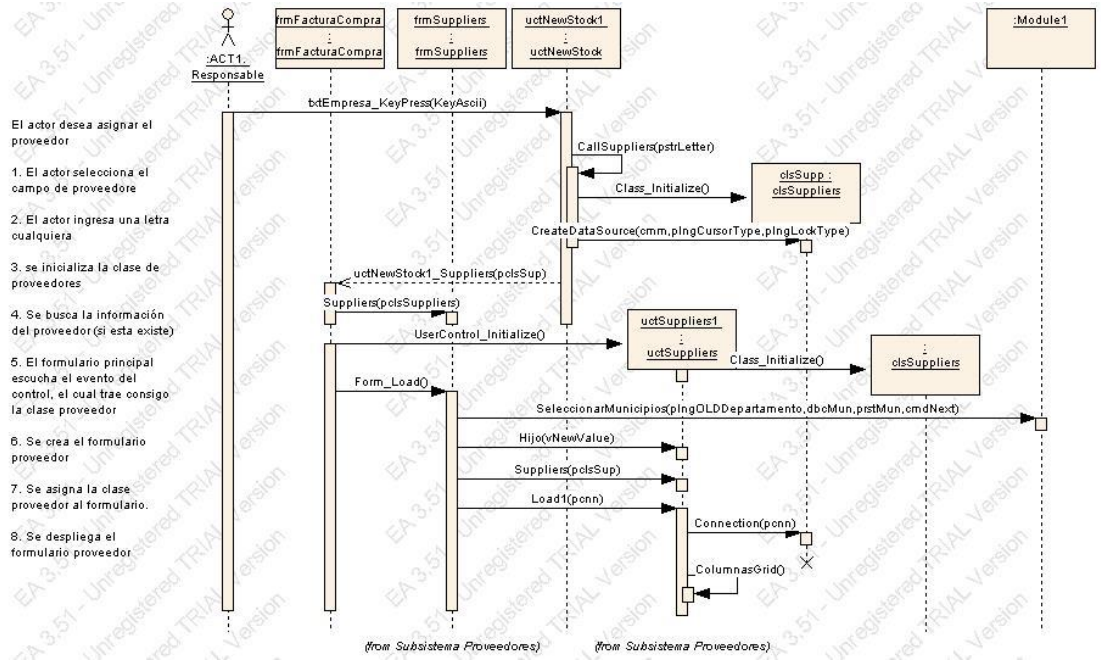


Fig. 49. Llamar a Proveedores

Tabla 39. Llamar a Proveedores Mensajes

ID	Mensaje	Del Objeto	Al Objeto	Notas
1	txtEmpresa_KeyPress(Integer*)		uctNewStock1	El actor ingresa una letra cualquiera en el campo de proveedor
2	CallSuppliers(String*)	uctNewStock1	uctNewStock1	Permite la creación de un objeto datasource específico que debe ser desplegado, y enviado a su correspondiente control para permitir la selección de un elemento vinculado con este control
3	Class_Initialize()	uctNewStock1	clsSupp	Se inicializa la clase asociada con el control de proveedores
4	CreateDataSource(ADODB.Command*, ADODB.CursorTypeEnum*, ADODB.LockTypeEnum*)	uctNewStock1	clsSupp	Abre la conexión y efectúa la busqueda de registros en la persistencia
5	uctNewStock1_Suppliers(NorthDLL.clsSuppliers*)	uctNewStock1	frmFacturaCompra	Procedimiento de escucha del control uctNewStock
6	Suppliers(NorthDLL.clsSuppliers*)	frmFacturaCompra	frmSuppliers	Asigna el objeto clsSuppliers

7	UserControl_Initialize()	frmFacturaCompra	uctSuppliers1	Se inicializa el control del formulario de proveedores
8	Class_Initialize()	uctSuppliers1		Se inicializa la clase asociada con el control
9	Form_Load()	frmFacturaCompra	frmSuppliers	Se carga el formulario
10	SeleccionarMunicipios(Variant*, DataCombo*, ADODB.Recordset*, CommandButton*)	frmSuppliers		Se seleccionan los municipios
11	Hijo(Boolean)	frmSuppliers	uctSuppliers1	Se le dice al formulario que debe actuar como un formulario hijo
12	Suppliers(NorthDLL.clsSuppliers*)	frmSuppliers	uctSuppliers1	Se le asigna al control de proveedores la clase de proveedores creada anteriormente
13	Load1(ADODB.Connection*)	frmSuppliers	uctSuppliers1	Se carga el control de proveedores
14	Connection(ADODB.Connection*)	uctSuppliers1	clsSupp	se asigna la conexión con la que trabajará la clase de proveedores
15	ColumnasGrid()	uctSuppliers1	uctSuppliers1	Procedimiento que permite el formateo de los títulos de las columnas

UCA51. Salvar Factura

Tipo: *public* **Use Case**
Estado: Terminado. Versión 1.0. Fase 1.0.
Paquete: Compras de Mercancía

Objetivo:
- Almacenar la información de la factura

Escenarios

Salvar Factura {Basic Path}.
El actor procede a salvar la factura

1. El actor hace click en el botón de salvar
2. El control llama a su clase asociada, para actualizar el registro

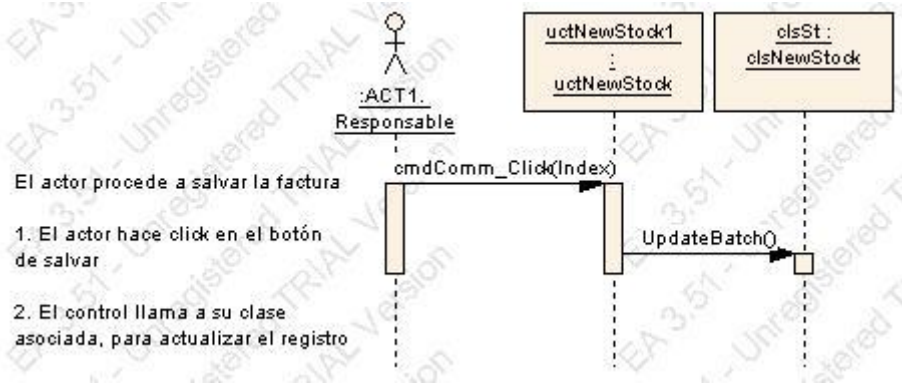


Fig. 50. Salvar Factura

Tabla 40. Salvar Factura Mensajes

ID	Mensaje	Del Objeto	Al Objeto	Notas
1	cmdComm_Click(Integer*)		uctNewStock1	Procedimiento que permite llamar a las rutinas de actualización - inserción, eliminación, adición o cancelación
2	UpdateBatch()	uctNewStock1	clsSt	Procedimiento de actualización por lotes

UCA52. Eliminar Factura

Tipo: *public* **Use Case**
Estado: Terminado. Versión 1.0. Fase 1.0.
Paquete: Compras de Mercancía

Objetivo:
- Permitir al actor eliminar la factura

Escenarios

Eliminar Factura {Basic Path}.
El actor desea eliminar la factura de compra

1. El actor hace click en eliminar la factura
2. El sistema Elimina la factura
3. El sistema actualiza el control

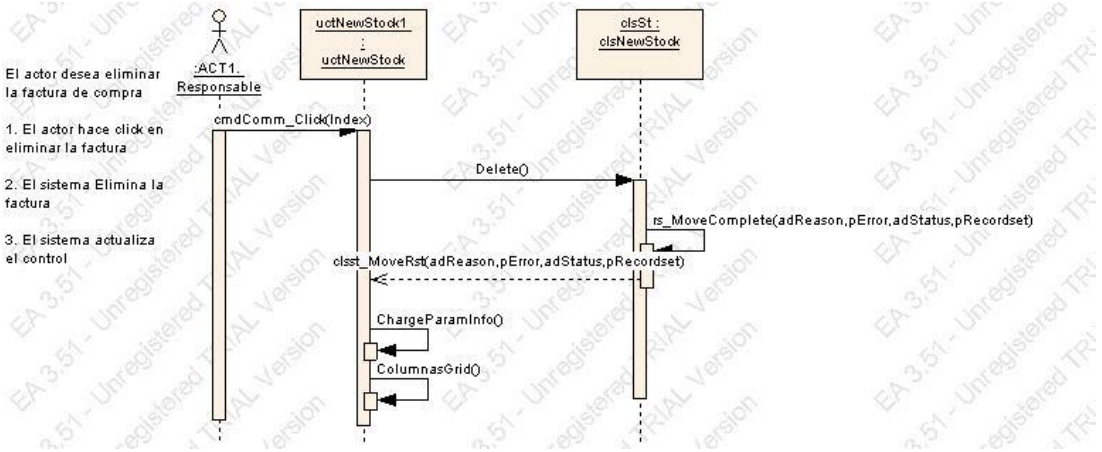


Fig. 51. Eliminar Factura

Tabla 41. Eliminar Factura Mensajes

ID	Mensaje	Del Objeto	Al Objeto	Notas
1	cmdComm_Click(Integer*)		uctNewStock1	Procedimiento que permite llamar a las rutinas de actualización - inserción, eliminación, adición o cancelación
2	Delete()	uctNewStock1	clsSt	Procedimiento Eliminar del Recordset
3	rs_MoveComplete(ADODB.EventReasonEnum, ADODB.Error, ADODB.EventStatusEnum*, ADODB.Recordset)	clsSt	clsSt	Evento llamado despues del cambio de posición del recordset
4	clsst_MoveRst (ADODB.EventReasonEnum, ADODB.Error, ADODB.EventStatusEnum*, ADODB.Recordset)	clsSt	uctNewStock1	el control escucha el Evento llamado despues del cambio de posición del recordset
5	ChargeParamInfo()	uctNewStock1	uctNewStock1	Actualiza la información en los controles vinculados
6	ColumnasGrid()	uctNewStock1	uctNewStock1	Procedimiento que permite el formateo de los títulos de las columnas

UCA53. Desplegar Productos

Tipo: *public* **Use Case**
Estado: Terminado. Versión 1.0. Fase 1.0.

Paquete: Compras de Mercancía

Objetivo:
- Desplegar los productos existentes en el sistema para que el usuario los pueda seleccionar y agregar de esta forma al detalle de la compra

Escenarios

Desplegar Productos {Basic Path}.
El actor procede a desplegar los productos, para seleccionar, ver o encontrar alguno en especial

- El control crea la clase de productos
- El control crea la cadena de búsqueda
- El control envía la clase al formulario contenedor
- El formulario contenedor asigna la clase al formulario productos
- Se carga el formulario de productos
- Se asigna la clase de productos al control del formulario productos
- Se despliega el formulario

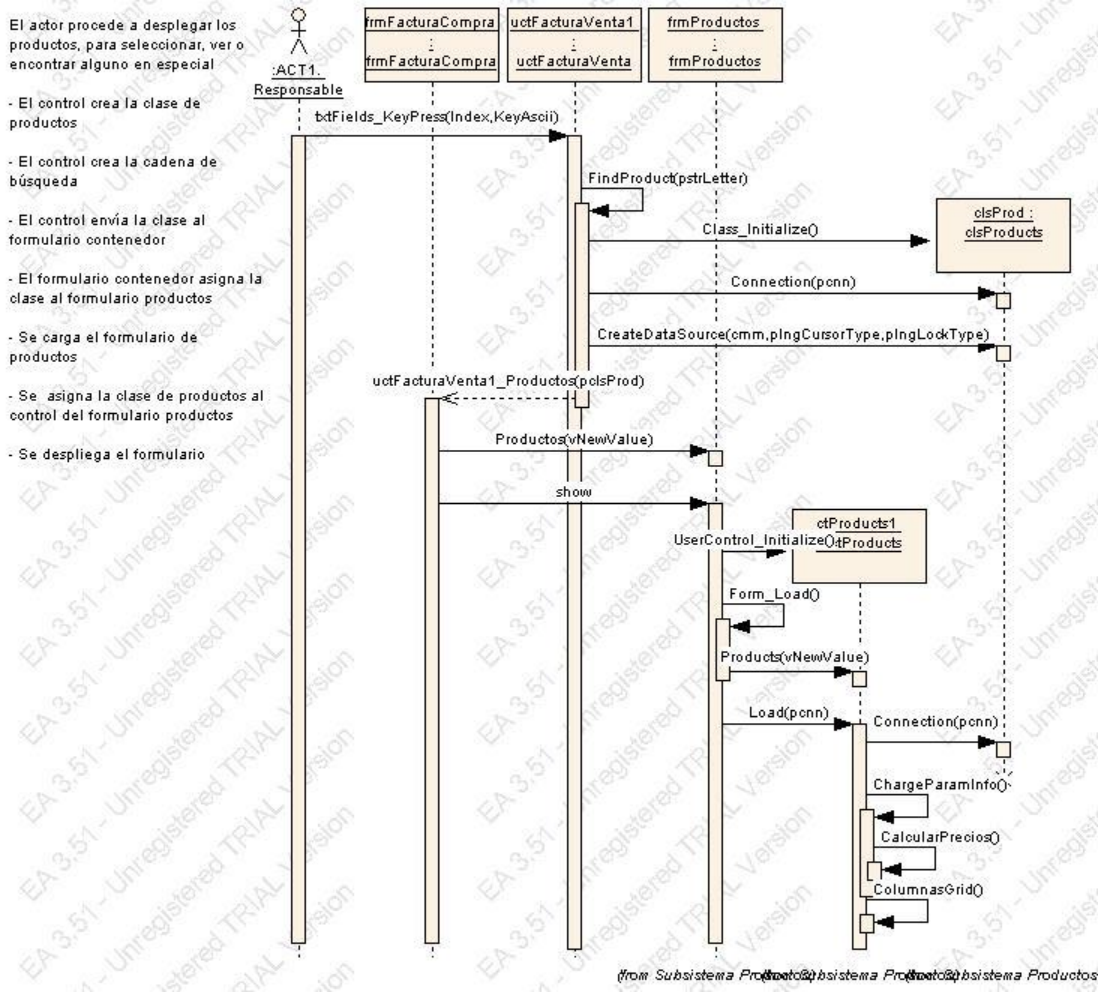


Fig. 52. Desplegar Productos

Tabla 42. Desplegar Productos Mensajes

ID	Mensaje	Del Objeto	Al Objeto	Notas
1	txtFields_KeyPress(Integer*, Integer*)		uctFacturaVenta1	Evento KeyPress de los texbox vinculados con el datasource. Permite determinar la tecla presionada
2	FindProduct(String*)	uctFacturaVenta1	uctFacturaVenta1	Permite la creación de un objeto datasource específico que debe ser desplegado, y enviado a su correspondiente control para permitir la selección de un elemento vinculado con este control
3	Class_Initialize()	uctFacturaVenta1	clsProd	Se crea la clase de productos
4	Connection(ADODB.Connection*)	uctFacturaVenta1	clsProd	Se le asigna la conexión a la clase
5	CreateDataSource(ADODB.Command*, ADODB.CursorTypeEnum*, ADODB.LockTypeEnum*)	uctFacturaVenta1	clsProd	Abre la conexión y efectúa la busqueda de registros en la persistencia
6	uctFacturaVenta1_Productos (NorthDLL.clsProducts*)	uctFacturaVenta1	frmFacturaCompra	Procedimiento de escucha del control uctFacturaVenta1
7	Productos(NorthDLL)	frmFacturaCompra	frmProductos	Asigna la variable clsProductos del formulario
8	Show	frmFacturaCompra	frmProductos	Despliega el formulario
9	UserControl_Initialize()	frmProductos	ctProducts1	Inicializa el control de productos
10	Form_Load()	frmProductos	frmProductos	Carga el formulario de productos
11	Products(Variant)	frmProductos	ctProducts1	Asigna la variable clsProductos del control
12	Load(ADODB.Connection*)	frmProductos	ctProducts1	Carga el control de productos
13	Connection(ADODB.Connection*)	ctProducts1	clsProd	Asigna la conexión a la clase de productos
14	ChargeParamInfo()	ctProducts1	ctProducts1	Busca la información del precio y actualiza los controles relacionados con los mismos
15	CalcularPrecios()	ctProducts1	ctProducts1	Calcula y despliega los precios del producto
16	ColumnasGrid()	ctProducts1	ctProducts1	Procedimiento que permite el formateo de los títulos de las columnas

UCA54. Salvar Mercancía

Tipo: *public* **Use Case**
Estado: Terminado. Versión 1.0. Fase 1.0.
Paquete: Compras de Mercancía

Objetivo:
- Permitir al actor salvar la información del detalle de la factura de compra de mercancía

Escenarios

Salvar Productos {Basic Path}.
El actor procede a salvar el detalle de la factura

- El actor hace click en salvar
- El control llama al método de actualización
- Se actualiza el inventario

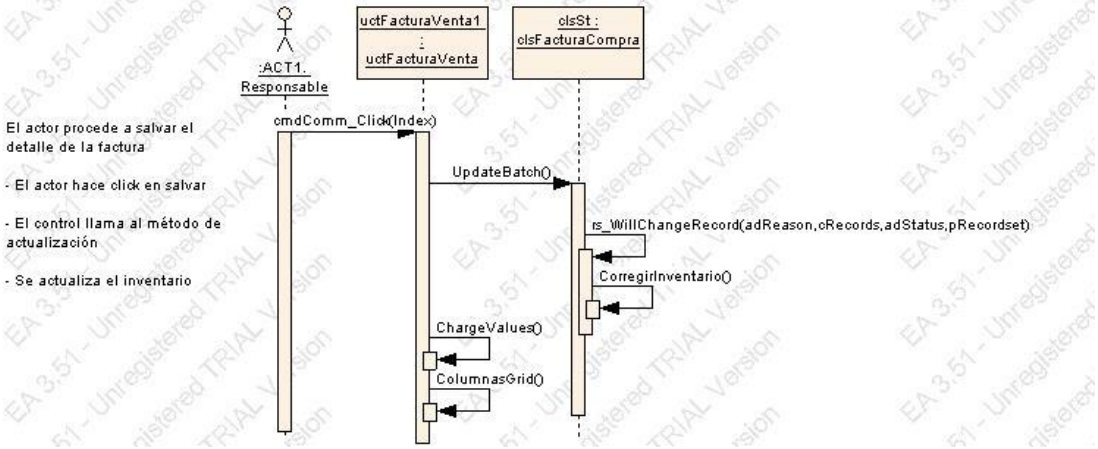


Fig. 53. Salvar Mercancía

Tabla 43. Salvar Mercancía Mensajes

ID	Mensaje	Del Objeto	Al Objeto	Notas
1	cmdComm_Click(Integer*)		uctFacturaVenta1	Procedimiento que permite llamar a las rutinas de actualización - inserción, eliminación, adición o cancelación
2	UpdateBatch()	uctFacturaVenta1	clsSt	Procedimiento de actualización por lotes
3	rs_WillChangeRecord(ADODB.EventReasonEnum, Long, ADODB.EventStatusEnum*, ADODB.Recordset)	clsSt	clsSt	Método llamado antes que uno o mas registros en el recordset cambien
4	CorregirInventario()	clsSt	clsSt	Corrige la información del inventario
5	ChargeValues(	uctFacturaVent	uctFacturaVent	Permite cargar los calores de fecha de

	)	a1	a1	vencimiento y unidades en stock
6	ColumnasGrid ()	uctFacturaVenta1	uctFacturaVenta1	Procedimiento que permite el formateo de los títulos de las columnas

Ventas de Productos

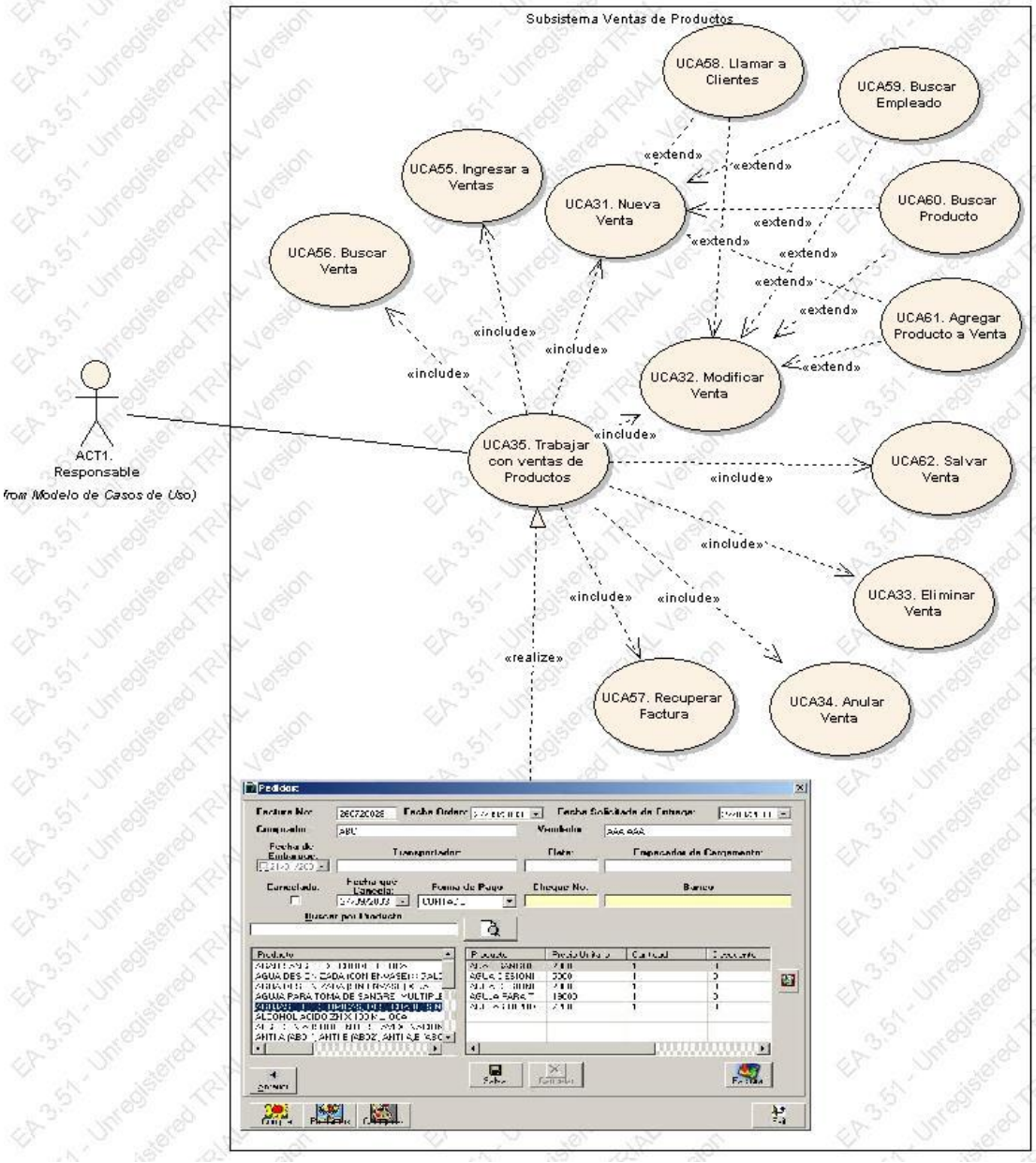


Fig. 54. Ventas de Productos

UCA31. Nueva Venta

Tipo: *public* **Use Case**
Estado: Terminado. Versión 1.0. Fase 1.0.
Paquete: Ventas de Productos

Objetivo:
- crear un nuevo registro de venta

Escenarios

Nueva Venta {Basic Path}.
El actor inicia una nueva venta de mercancia

1. El actor selecciona nuevo
2. El sistema despliega el formulario que permite ingresar nuevos productos para su venta

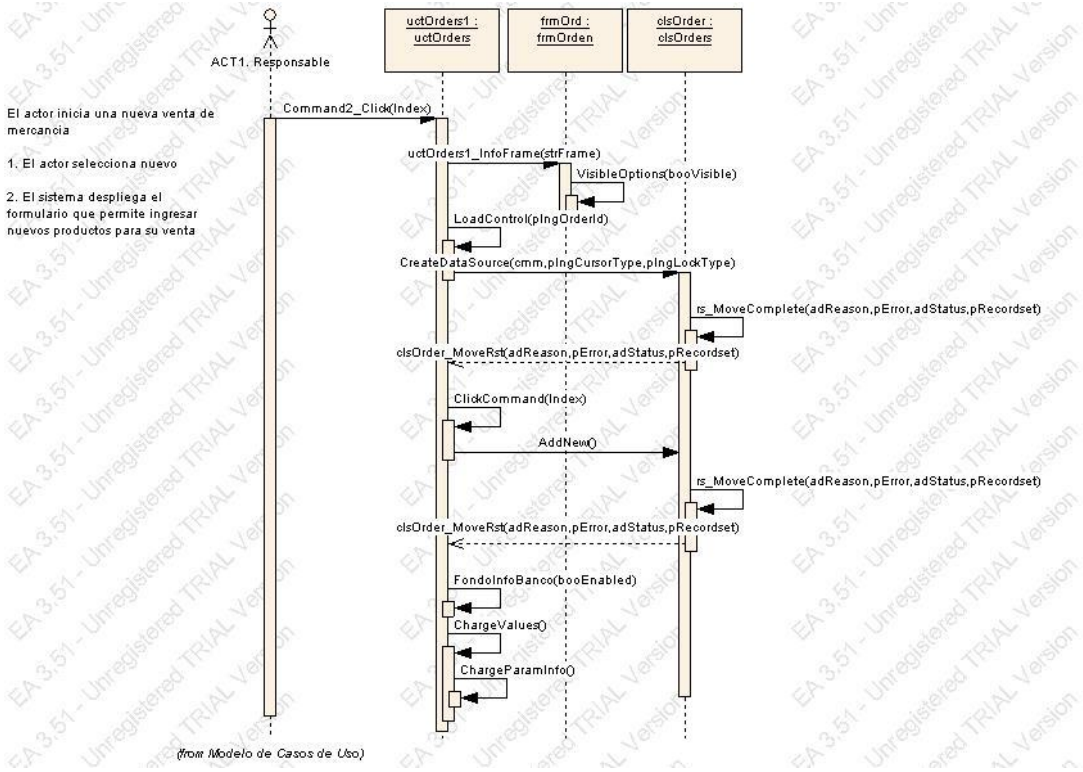


Fig. 55. Nueva Venta

Tabla 44. Nueva Venta Mensajes

ID	Mensaje	Del Objeto	Al Objeto	Notas
1	Command2_Click(Integer*)	ACT1. Responsable	uctOrders1	Permite Buscar o agregar una factura
2	uctOrders1_InfoFrame(String*)	uctOrders1	frmOrd	Frame visible en el control

3	VisibleOptions (Boolean*)	frmOrd	frmOrd	Visualiza o oculta los botones
4	LoadControl(Long*)	uctOrders1	uctOrders1	Procedimiento que permite la carga de la clase datasource
5	CreateDataSource(ADODB.Command*, ADODB.CursorTypeEnum*, ADODB.LockTypeEnum*)	uctOrders1	clsOrder	Abre la conexión y efectúa la busqueda de registros en la persistencia
6	rs_MoveComplete(ADODB.EventReasonEnum, ADODB.Error, ADODB.EventStatusEnum*, ADODB.Recordset)	clsOrder	clsOrder	Evento llamado despues del cambio de posición del recordset
7	clsOrder_MoveRst(ADODB.EventReasonEnum, ADODB.Error, ADODB.EventStatusEnum*, ADODB.Recordset)	clsOrder	uctOrders1	el control escucha el Evento llamado despues del cambio de posición del recordset
8	ClickCommand(Integer*)	uctOrders1	uctOrders1	Procedimiento que permite llamar a las rutinas de actualización - inserción, eliminación, adición o cancelación
9	AddNew()	uctOrders1	clsOrder	Crea un nuevo registro en el recordset
10	rs_MoveComplete(ADODB.EventReasonEnum, ADODB.Error, ADODB.EventStatusEnum*, ADODB.Recordset)	clsOrder	clsOrder	Evento llamado despues del cambio de posición del recordset
11	clsOrder_MoveRst(ADODB.EventReasonEnum, ADODB.Error, ADODB.EventStatusEnum*, ADODB.Recordset)	clsOrder	uctOrders1	el control escucha el Evento llamado despues del cambio de posición del recordset
12	FondoInfoBanco(Boolean*)	uctOrders1	uctOrders1	Cambia el control de fondo de los controles

13	ChargeValues( )	uctOrders1	uctOrders1	Asigna los valores de la clase datasource a los controles
14	ChargeParamI nfo()	uctOrders1	uctOrders1	Busca la información relacionada con la órden

UCA32. Modificar Venta

Tipo: *public* **Use Case**
Estado: Terminado. Versión 1.0. Fase 1.0.
Paquete: Ventas de Productos

Objetivo:
- Modificar la información almacenada de una venta

Escenarios

Modificar Venta {Basic Path}.
El actor desea modificar la información de una venta

1. El actor cambia la información de la factura de venta
2. El actor almacena la información
3. El sistema actualiza la clase de detalles de la orden y la salva
4. Se actualiza el stock

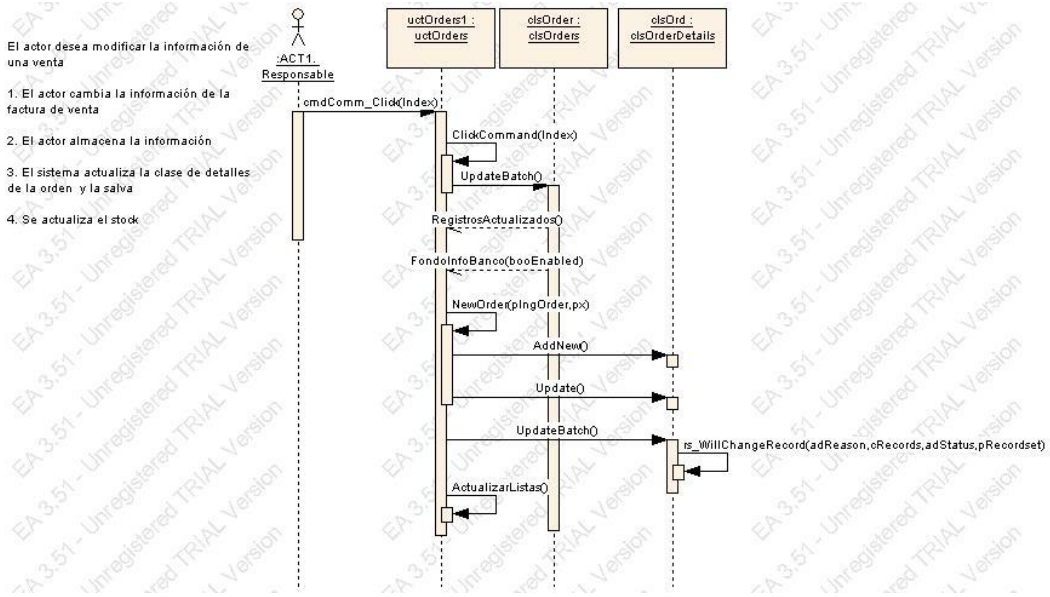


Fig. 56. Modificar Venta

Tabla 45. Modificar Venta Mensajes

ID	Mensaje	Del Objeto	Al Objeto	Notas
1	cmdComm_Cli ck(Integer*)		uctOrders1	Procedimiento que permite llamar a las rutinas de Anulación, actualización, de la factura
2	ClickComman	uctOrders1	uctOrders1	Procedimiento que permite llamar a las rutinas

	d(Integer*)			de actualización - inserción, eliminación, adición o cancelación
3	UpdateBatch()	uctOrders1	clsOrder	Procedimiento de actualización por lotes
4	RegistrosActualizados()	clsOrder	uctOrders1	Inhabilita el botón de actualizar y habilita el botón de cancelar
5	FondoInfoBanco(Boolean*)	clsOrder	uctOrders1	Cambia el color de fondo de los controles
6	NewOrder(Long*, ListItem*)	uctOrders1	uctOrders1	Actualiza la clase de detalles de la orden y la salva
7	AddNew()	uctOrders1	clsOrd	Crea un nuevo registro en el recordset
8	Update()	uctOrders1	clsOrd	Procedimiento para actualizar el recordset
9	UpdateBatch()	uctOrders1	clsOrd	Procedimiento de actualización por lotes
10	rs_WillChangeRecord(ADODB.EventReasonEnum, Long, ADODB.EventStatusEnum*, ADODB.Recordset)	clsOrd	clsOrd	Evento llamado despues del cambio de posición del recordset
11	ActualizarListas()	uctOrders1	uctOrders1	Actualiza las vistas de productos disponibles y productos seleccionados

UCA33. Eliminar Venta

Tipo: *public* **Use Case**
Estado: Terminado. Versión 1.0. Fase 1.0.
Paquete: Ventas de Productos

Objetivo:
- Eliminar la información de una venta

Escenarios

Eliminar Venta {Basic Path}.
El actor desea eliminar un producto de la venta

1. El actor selecciona el producto a eliminar
2. El actor procede a eliminarlo
3. El sistema elimina el registro de la persistencia

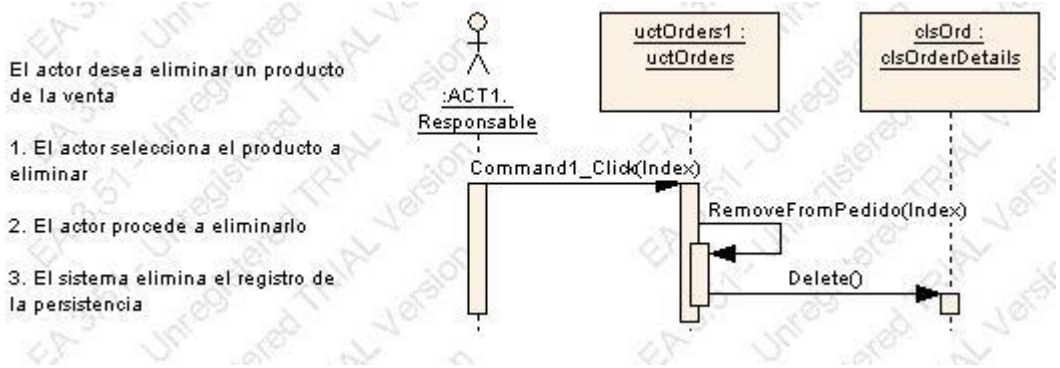


Fig. 57. Eliminar Venta

Tabla 46. Eliminar Venta Mensajes

ID	Mensaje	Del Objeto	Al Objeto	Notas
1	Command1_Click(Integer*)		uctOrders1	Permite visualizar los controles contenidos en cada uno de los frames
2	RemoveFromPedido(String*)	uctOrders1	uctOrders1	Remueve un producto pedido de la lista
3	Delete()	uctOrders1	clsOrd	Procedimiento Eliminar del Recordset

UCA34. Anular Venta

Tipo: *public* **Use Case**
Estado: Terminado. Versión 1.0. Fase 1.0.
Paquete: Ventas de Productos

Objetivo:
- Anular una venta

Escenarios

Anular Venta {Basic Path}.
El actor decide anular una venta

1. El actor selecciona el botón de anular venta
2. El sistema actualiza la lista de productos vendidos
3. Se actualiza el sistema

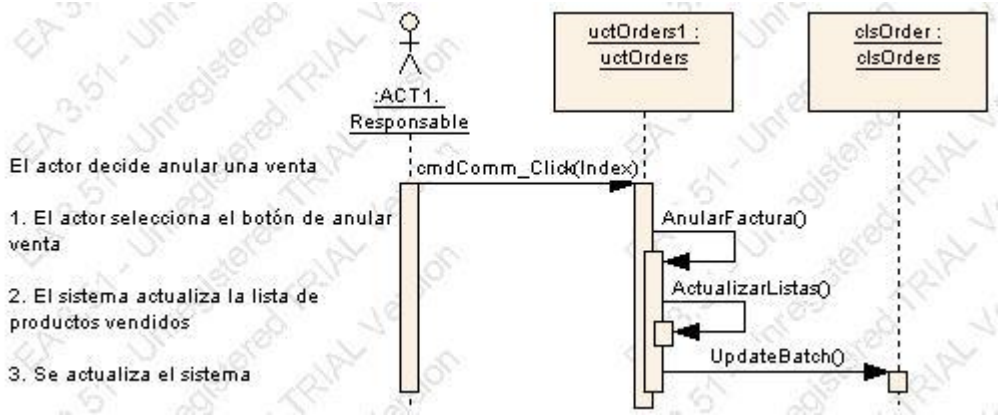


Fig. 58. Anular Factura

Tabla 47. Anular Factura Mensajes

ID	Mensaje	Del Objeto	Al Objeto	Notas
1	cmdComm_Click(Integer*)		uctOrders1	Procedimiento que permite llamar a las rutinas de Anulación, actualización, de la factura
2	AnularFactura()	uctOrders1	uctOrders1	Anula la factura
3	ActualizarListas()	uctOrders1	uctOrders1	Actualiza las vistas de productos disponibles y productos seleccionados
4	UpdateBatch()	uctOrders1	clsOrder	Procedimiento de actualización por lotes

UCA35. Trabajar con ventas de Productos

Tipo: *public* **Use Case**
Estado: Terminado. Versión 1.0. Fase 1.0.
Paquete: Ventas de Productos

Al trabajar con las ventas de los productos, es posible:
- Agregar una nueva venta
- Modificar una venta
- Eliminar una venta
- Anular una venta
- Adicionar productos a la venta
- Buscar una venta
- buscar un producto disponible

UCA55. Ingresar a Ventas

Tipo: *public* **Use Case**
Estado: Terminado. Versión 1.0. Fase 1.0.
Paquete: Ventas de Productos

Objetivo:
- Permitir el ingreso del actor al formulario de ventas de reactivos

Escenarios

Ingresar a Ventas {Basic Path}.
El actor decide ingresar al formulario de ventas

1. El actor selecciona el menú Ventas de mercancía
2. El sistema llama al formulario frmOrden
3. El formulario inicializa el control de ordenes
4. El control inicializa la clase asociada de ordenes
5. La clase inicia a su clase de Embarcadores y empleados
6. Se carga el formulario de órdenes
7. Se carga el control de ordenes
8. Se asigna la conexión a la clase de ordenes
9. Se asigna el empleado por defecto
10. Se asigna el embarcador por defecto

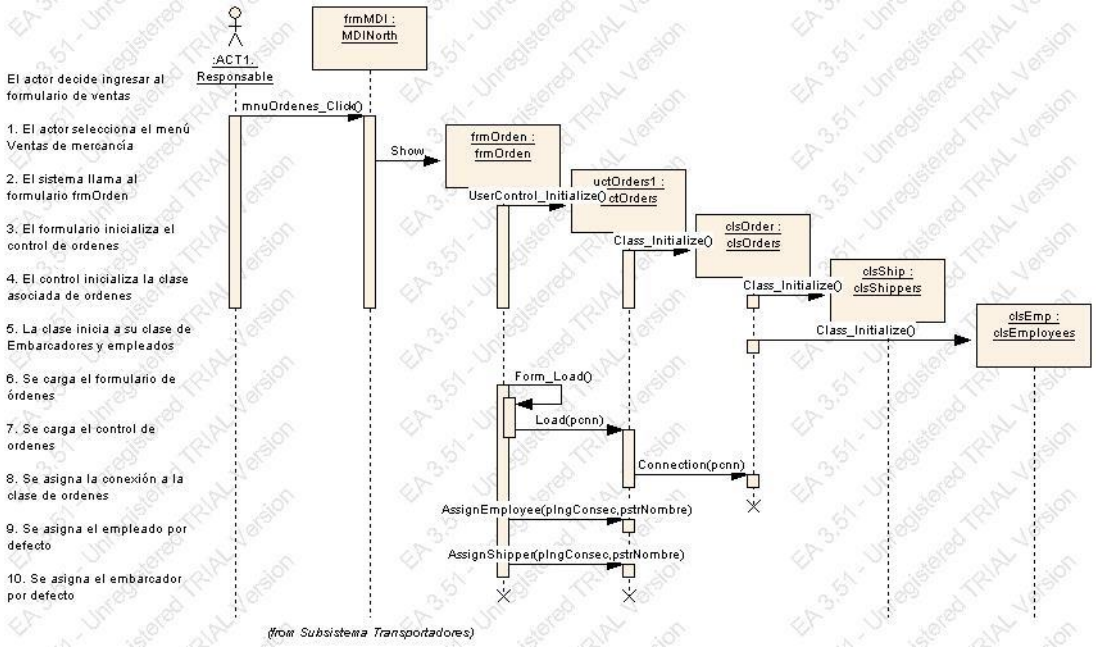


Fig. 59. Ingresar a Ventas

Tabla 48. Ingresar a Ventas Mensajes

ID	Mensaje	Del Objeto	Al Objeto	Notas
1	mnuOrdenes_ Click()		frmMDI	Llama al formulario de órdenes
2	Show	frmMDI	frmOrden	Despliega el formulario
3	UserControl_I nitialize()	frmOrden	uctOrders1	Inicializa el control
4	Class_Initialize ()	uctOrders1	clsOrder	Inicializa la clase asociada
5	Class_Initialize	clsOrder	clsShip	Inicializa la clase asociada

	()			
6	Class_Initialize()	clsOrder	clsEmp	Inicializa la clase de empleados
7	Form_Load()	frmOrden	frmOrden	Carga el formulario
8	Load(ADODB.Connection*)	frmOrden	uctOrders1	Comienza la carga del control
9	Connection(ADODB.Connection*)	uctOrders1	clsOrder	Asigna la conexión a la clase ordenes
10	AssignEmployee(Long*, String*)	frmOrden	uctOrders1	Asigna el nombre y el id del empleado por defecto
11	AssignShipper(Long*, String*)	frmOrden	uctOrders1	Asigna el embarcador por defecto

UCA56. Buscar Venta

Tipo: *public* **Use Case**
Estado: Terminado. Versión 1.0. Fase 1.0.
Paquete: Ventas de Productos

Objetivo:
- Buscar las ventas registradas, por el criterio especificado por el usuario

Escenarios

Buscar Venta {Basic Path}.
El actor desea buscar una orden en especial

1. el actor selecciona el botón de buscar
2. El sistema arma el criterio de busqueda
3. El sistema comienza la búsqueda de registros
4. El sistema despliega la pantalla de resultados o factura, dependiendo de si son uno, ninguno, o muchos los resultados devueltos

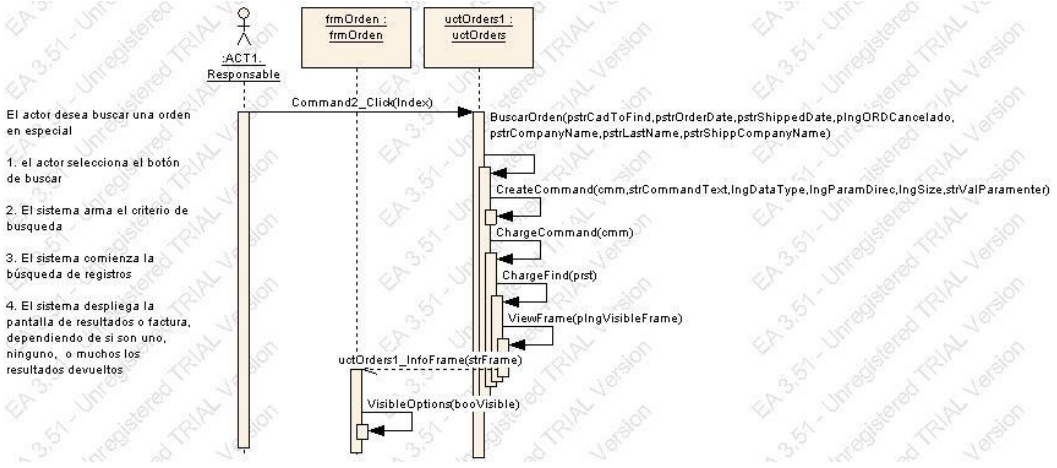


Fig. 60. Buscar Venta

Tabla 49. Buscar Venta Mensajes

ID	Mensaje	Del Objeto	Al Objeto	Notas
1	Command2_Click(Integer*)		uctOrders1	
2	BuscarOrden(String*, String*, String*, Integer*, String*, String*, String*)	uctOrders1	uctOrders1	Busca una orden de acuerdo al criterio seleccionado
3	CreateCommand(ADODB.Command*, String*, ADODB.DataTypeEnum*, ADODB.ParameterDirectionEnum*, Long*, String*)	uctOrders1	uctOrders1	Crea el objeto command, asigna la conexión y los parámetros
4	ChargeCommand(ADODB.Command*)	uctOrders1	uctOrders1	Abre el recordset que busca la factura
5	ChargeFind(ADODB.Recordset*)	uctOrders1	uctOrders1	Carga el listview si existen muchos registros coincidentes, visualiza la factura si existe solo uno, o avisa al usuario si no existen coincidencias
6	ViewFrame(Integer*)	uctOrders1	uctOrders1	Visualiza o oculta los elementos del frame 1, y genera el evento para cada control
7	uctOrders1_InfoFrame(String*)	uctOrders1	frmOrden	Devuelve la información del frame que se debe estar visualizando
8	VisibleOptions(Boolean*)	frmOrden	frmOrden	Visualiza o oculta los botones

UCA57. Recuperar Factura

Tipo: *public* **Use Case**
Estado: Terminado. Versión 1.0. Fase 1.0.
Paquete: Ventas de Productos

Objetivo:
- Permitir al actor recuperar las facturas registradas en el sistema

Escenarios

Recuperar Factura {Basic Path}.
El actor desea recuperar una de las facturas desplegadas en el listview de búsqueda

1. El actor da doble click en la factura
2. El sistema busca la factura seleccionada por el usuario
3. El sistema despliega la factura y sus productos

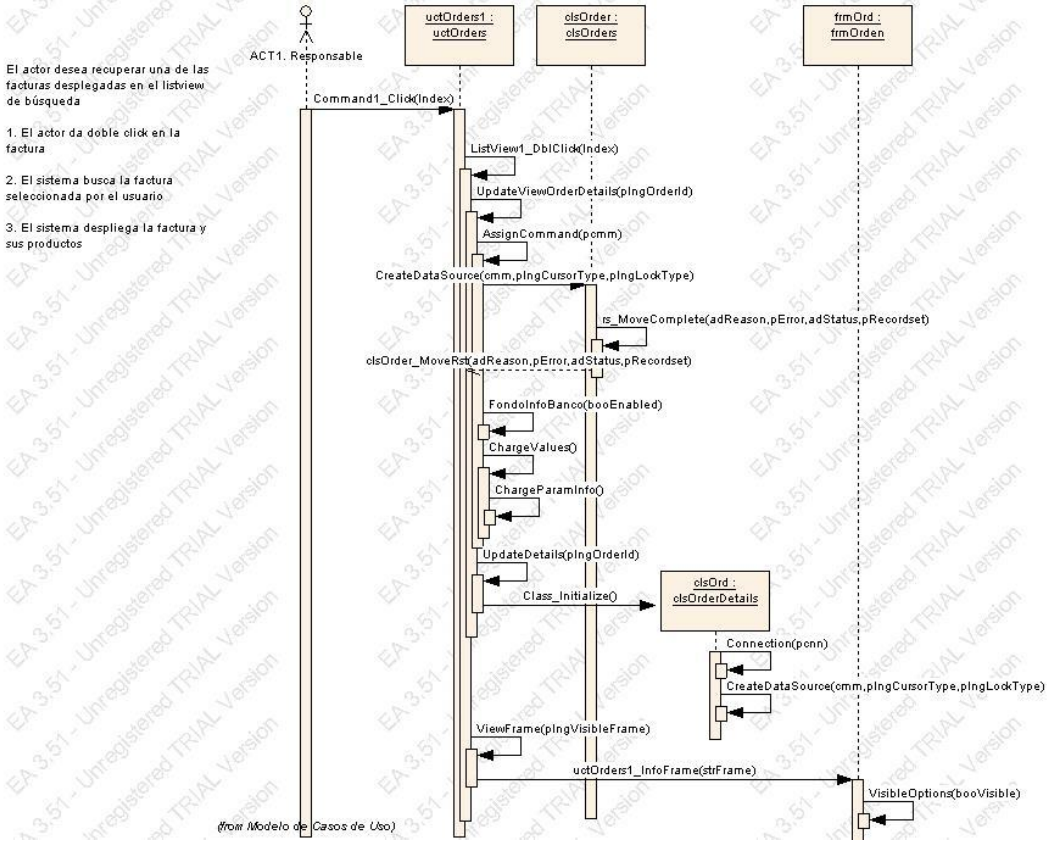


Fig. 61. Recuperar Factura

Tabla 50. Recuperar Factura Mensajes

ID	Mensaje	Del Objeto	Al Objeto	Notas
1	Command1_Click(Integer*)	ACT1. Responsable	uctOrders1	Permite visualizar los controles contenidos en cada uno de los frames
2	ListView1_DblClick(Integer*)	uctOrders1	uctOrders1	Envía la información de la lista de productos disponibles a la de productos seleccionados
3	UpdateViewOrderDetails(Long*)	uctOrders1	uctOrders1	Actualiza la vista de detalles de la órden
4	AssignCommand(ADODB.Command*)	uctOrders1	uctOrders1	Permite llamar al procedimiento que crea el datasource en la clase datasource
5	CreateDataSo	uctOrders1	clsOrder	

	urce(ADODB.Command*, ADODB.CursorTypeEnum*, ADODB.LockTypeEnum*)			
6	rs_MoveComplete(ADODB.EventReasonEnum, ADODB.Error, ADODB.EventStatusEnum*, ADODB.Recordset)	clsOrder	clsOrder	Evento llamado despues del cambio de posición del recordset
7	clsOrder_MoveRst(ADODB.EventReasonEnum, ADODB.Error, ADODB.EventStatusEnum*, ADODB.Recordset)	clsOrder	uctOrders1	El control escucha el Evento llamado despues del cambio de posición del recordset
8	FondoInfoBanco(Boolean*)	uctOrders1	uctOrders1	Cambia el color de fondo de los controles
9	ChargeValues()	uctOrders1	uctOrders1	Asigna los valores de la clase datasource a los controles
10	ChargeParamInfo()	uctOrders1	uctOrders1	Busca la información relacionada con la órden
11	UpdateDetails(Long*)	uctOrders1	uctOrders1	Actuliza la listview de productos seleccionados en la órden
12	Class_Initialize()	uctOrders1	clsOrd	Inicializa la clase de detalles de la orden
13	Connection(ADODB.Connection*)	clsOrd	clsOrd	establece la conexión
14	CreateDataSource(ADODB.Command*, ADODB.CursorTypeEnum*, ADODB.LockTypeEnum*)	clsOrd	clsOrd	Abre la conexión y efectúa la busqueda de registros en la persistencia
15	ViewFrame(Integer*)	uctOrders1	uctOrders1	Visualiza o oculta los elementos del frame 1, y genera el evento para cada control
16	uctOrders1_InfoFrame(String*)	uctOrders1	frmOrd	Despliega los botones adecuados, dependiendo el frame
17	VisibleOptions(Boolean*)	frmOrd	frmOrd	Visualiza o oculta los botones

UCA58. Llamar a Clientes

Tipo: *public* **Use Case**
Estado: Terminado. Versión 1.0. Fase 1.0.
Paquete: Ventas de Productos

Objetivo:
- Permitir al actor seleccionar el cliente al cual se le vendió el producto

Escenarios

llamar a Clientes {Basic Path}.
El actor desea agregar o modificar un cliente

1. El actor selecciona el textbox de clientes
2. El actor oprime una tecla
3. El sistema despliega la pantalla de clientes
4. El actor busca o modifica la información del cliente
5. El actor sale de la pantalla
6. El sistema despliega la información del cliente seleccionado

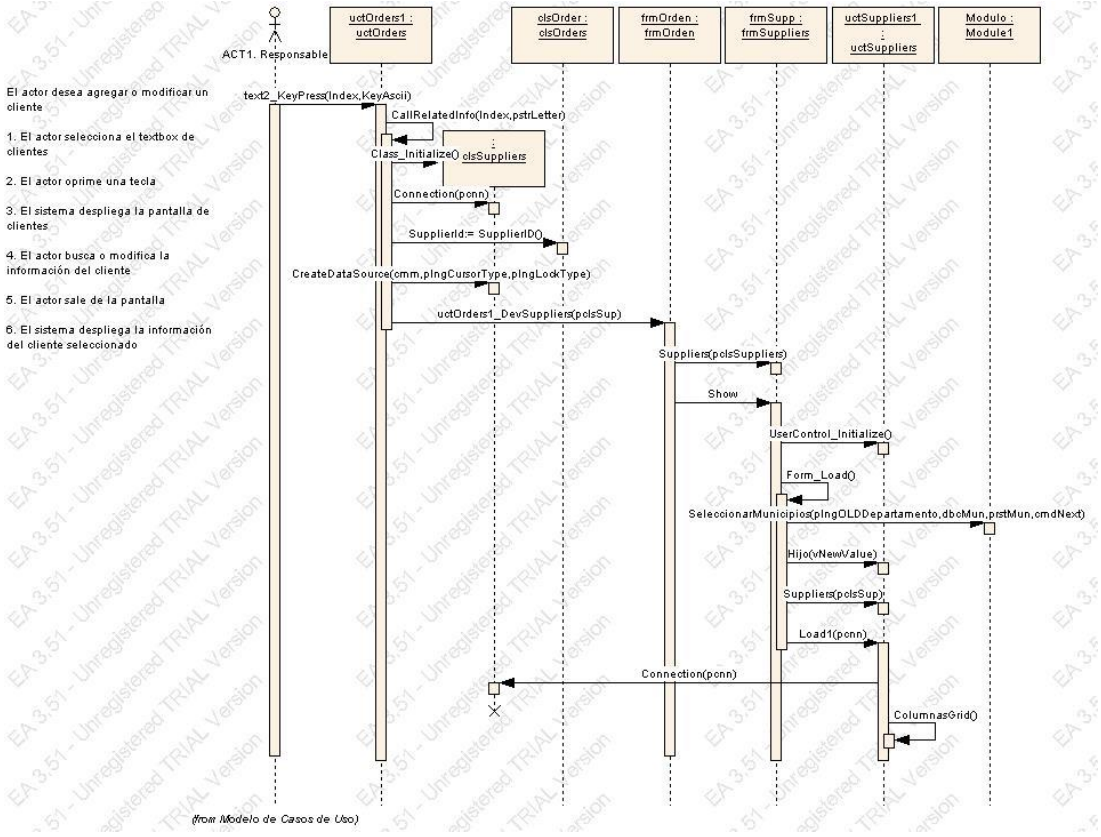


Fig. 62. Llamar a Proveedores

Tabla 51. Llamar a Proveedores Mensajes

ID	Mensaje	Del Objeto	Al Objeto	Notas
1	text2_KeyPress(Integer*, Integer*)	ACT1. Responsable	uctOrders1	Evento keypress de los textbox Comprador, vendedor o transportador
2	CallRelatedInfo(Integer*, String*)	uctOrders1	uctOrders1	Permite la creación de un objeto datasource específico que debe ser desplegado, y enviado a su correspondiente control para permitir la selección de un elemento vinculado con este control
3	Class_Initialize()	uctOrders1		Se inicializa la clase de proveedores (Clientes)
4	Connection(ADODB.Connection*)	uctOrders1		Se asigna la conexión a la clase
5	SupplierID()	uctOrders1	clsOrder	Busca el Id del cliente
6	CreateDataSource(ADODB.Command*, ADODB.CursorTypeEnum*, ADODB.LockTypeEnum*)	uctOrders1		Abre la conexión y efectúa la busqueda de registros en la persistencia
7	uctOrders1_DevSuppliers(NorthDLL.clsSuppliers*)	uctOrders1	frmOrden	Devuelve en el evento la clase proveedor creada, para que esta sea utilizada por el formulario frmOrden
8	Suppliers(NorthDLL.clsSuppliers*)	frmOrden	frmSupp	El formulario asigna a el formulario frmSuppliers la clase de proveedores recibida
9	Show	frmOrden	frmSupp	Se despliega el formulario de proveedores
10	UserControl_Initialize()	frmSupp	uctSuppliers1	Se inicializa el control de proveedores
11	Form_Load()	frmSupp	frmSupp	Se carga el formulario
12	SeleccionarMunicipios(Variant*, DataCombo*, ADODB.Recordset*, CommandButton*)	frmSupp	Modulo	Se buscan los municipios de acuerdo al departamento seleccionado
13	Hijo(Boolean)	frmSupp	uctSuppliers1	Se le dice al formulario que debe comportarse como un formulario hijo
14	Suppliers(NorthDLL.clsSuppliers*)	frmSupp	uctSuppliers1	Se asigna la clase proveedores anteriormente recibida
15	Load1(ADODB.Connection*)	frmSupp	uctSuppliers1	Se carga el control con la clase asignada
16	Connection(ADODB.Connection*)	uctSuppliers1		Se asigna la conexión a la clase de proveedores
17	ColumnasGrid	uctSuppliers1	uctSuppliers1	Procedimiento que permite el formateo de los

	()			títulos de las columnas

UCA59. Buscar Empleado

Tipo: *public* **Use Case**
Estado: Terminado. Versión 1.0. Fase 1.0.
Paquete: Ventas de Productos

Objetivo:
- Permitir al actor determinar el empleado responsable de la venta del producto

Escenarios

Buscar Empleado {Basic Path}.
El actor desea asignar o modificar el empleado que vendío el/los producto(s)

1. El actor selecciona el campo de empleados
2. El actor oprime una tecla
3. El sistema despliega el formulario de empleados, para que el actor seleccione el empleado deseado
4. El actor termina
5. El sistema asigna la información del empleado al formulario de venta

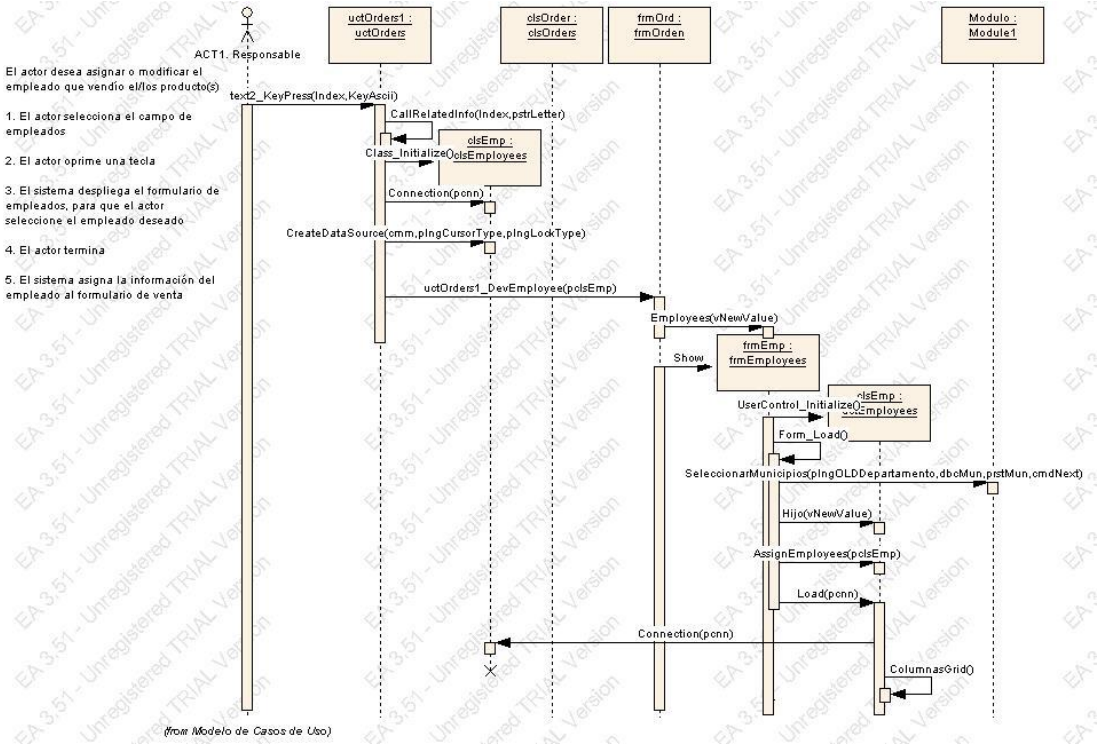


Fig. 63. Buscar Empleado

Tabla 52. Buscar Empleado Mensajes

ID	Mensaje	Del Objeto	Al Objeto	Notas
1	text2_KeyPress(Integer*, Integer*)	ACT1. Responsable	uctOrders1	Evento keypress de los textbox Comprador, vendedor o transportador
2	CallRelatedInfo(Integer*, String*)	uctOrders1	uctOrders1	Permite la creación de un objeto datasource específico que debe ser desplegado, y enviado a su correspondiente control para permitir la selección de un elemento vinculado con este control
3	Class_Initialize()	uctOrders1	clsEmp	Se inicializa la clase de empleados
4	Connection(ADODB.Connection*)	uctOrders1	clsEmp	Se asigna la conexión a la clase
5	CreateDataSource(ADODB.Command*, ADODB.CursorTypeEnum*, ADODB.LockTypeEnum*)	uctOrders1	clsEmp	Abre la conexión y efectúa la busqueda de registros en la persistencia
6	uctOrders1_DevEmployee(NorthDLL.clsEmployees*)	uctOrders1	frmOrd	Se devuelve la clase empleado cargada al formulario para que este proceda a desplegar el formulario correspondiente
7	Employees(NorthDLL)	frmOrd	frmEmp	Asigna la clase empleado al formulario
8	Show	frmOrd	frmEmp	Se visualiza el formulario
9	UserControl_Initialize()	frmEmp	clsEmp	Se inicializa el control de empleados
10	Form_Load()	frmEmp	frmEmp	Se carga el formulario
11	SeleccionarMunicipios(Variant*, DataCombo*, ADODB.Recordset*, CommandButton*)	frmEmp	Modulo	Se buscan los municipios de acuerdo al departamento
12	Hijo(Boolean)	frmEmp	clsEmp	Se le dice al formulario que debe compartarse como un formulario hijo
13	AssignEmployees(NorthDLL.clsEmployees*)	frmEmp	clsEmp	Se asigna la clase empleado anteriormente recibida al control
14	Load(ADODB.Connection*)	frmEmp	clsEmp	Se procede con la carga del control
15	Connection(ADODB.Connection*)	clsEmp	clsEmp	Se asigna la conexión a la clase empleados

16	ColumnasGrid ()	clsEmp	clsEmp	Procedimiento que permite el formateo de los títulos de las columnas

UCA60. Buscar Producto

Tipo: *public* **Use Case**
Estado: Terminado. Versión 1.0. Fase 1.0.
Paquete: Ventas de Productos

Objetivo:
- Permitir al actor buscar los productos registrados en el sistema y disponibles para su venta

Escenarios

Buscar Producto {Basic Path}.
El actor desea buscar un producto

1. El actor ingresa la información del producto
2. El actor procede a buscar el producto
3. El sistema despliega los productos que cumplan con el criterio de búsqueda

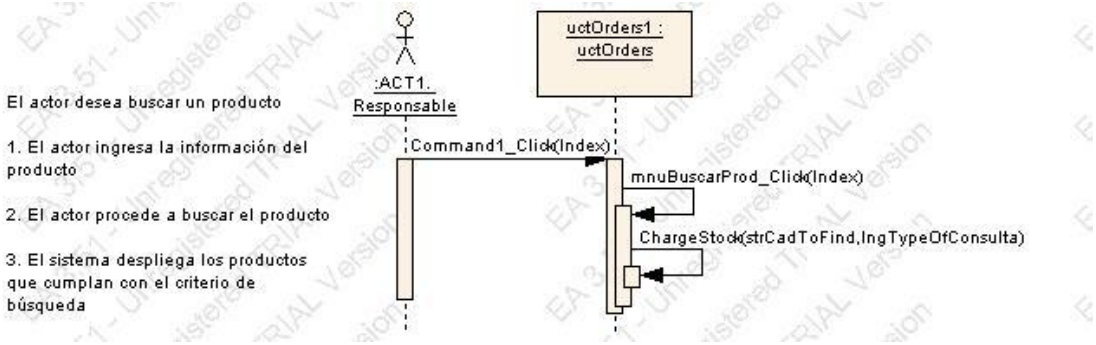


Fig. 64. Buscar Producto

Tabla 53. Buscar Producto Mensajes

ID	Mensaje	Del Objeto	Al Objeto	Notas
1	Command1_Click(Integer*)		uctOrders1	Permite visualizar los controles contenidos en cada uno de los frames
2	mnuBuscarProd_Click(Integer*)	uctOrders1	uctOrders1	Permite cargar la clase datasource, de acuerdo a un criterio seleccionado
3	ChargeStock(String*, Long*)	uctOrders1	uctOrders1	Carga la lista de productos disponibles

UCA61. Agregar Producto a Venta

Tipo: *public* **Use Case**
Estado: Terminado. Versión 1.0. Fase 1.0.
Paquete: Ventas de Productos

Objetivo:
- Agregar un nuevo producto a la venta

Escenarios

Agregar Producto a venta {Basic Path}.
El actor desea agregar un producto a la lista de productos vendidos

1. El actor selecciona el producto
2. El actor envía el producto al área de productos vendidos
3. El sistema quita el producto del área de disponibles y los lleva al área de vendidos

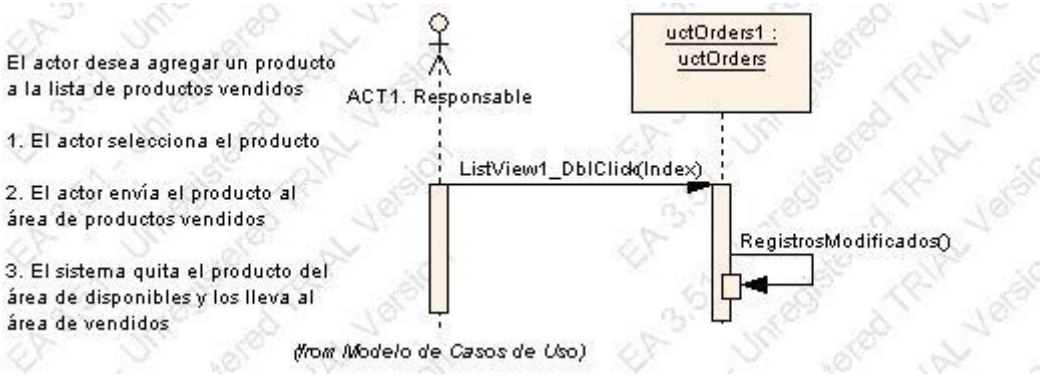


Fig. 65. Agregar Producto a Venta

Tabla 54. Agregar Producto a Venta Mensajes

ID	Mensaje	Del Objeto	Al Objeto	Notas
1	ListView1_Dbl Click(Integer*)	ACT1. Responsable	uctOrders1	Envía la información de la lista de productos disponibles a la de productos seleccionados
2	RegistrosModif icados()	uctOrders1	uctOrders1	Habilita o inhabilita los botones de actualizar y cancelar

UCA62. Salvar Venta

Tipo: *public* **Use Case**
Estado: Terminado. Versión 1.0. Fase 1.0.
Paquete: Ventas de Productos

Objetivo:
- Almacenar la venta en la persistencia

Escenarios

Salvar Venta {Basic Path}.
El actor ha concluido la venta, y desea salvar la información

1. El actor selecciona el botón de guardar

2. El sistema almacena la información de la factura y del detalle de la misma

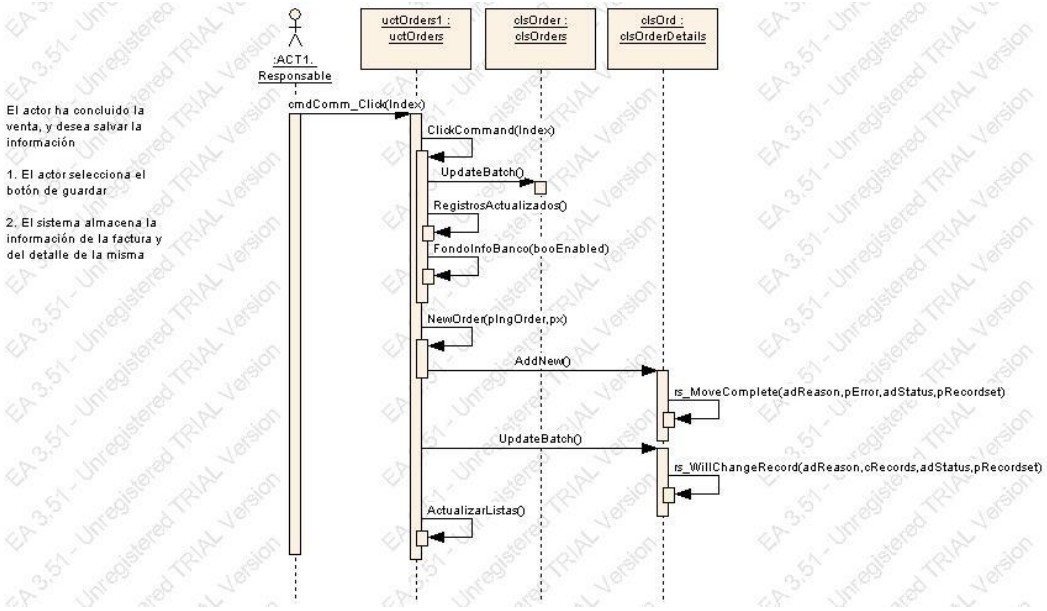


Fig. 66. Salvar Venta

Tabla 55. Salvar Venta Mensajes

ID	Mensaje	Del Objeto	Al Objeto	Notas
1	cmdComm_Click(Integer*)		uctOrders1	Procedimiento que permite llamar a las rutinas de Anulación, actualización, de la factura
2	ClickCommand(Integer*)	uctOrders1	uctOrders1	Procedimiento que permite llamar a las rutinas de actualización - inserción, eliminación, adición o cancelación
3	UpdateBatch()	uctOrders1	clsOrder	Procedimiento de actualización por lotes
4	RegistrosActualizados()	uctOrders1	uctOrders1	Inhabilita el botón de actualizar y habilita el botón de cancelar
5	FondoInfoBanco(Boolean*)	uctOrders1	uctOrders1	Cambia el color de fondo de los controles
6	NewOrder(Long*, ListItem*)	uctOrders1	uctOrders1	Actualiza la clase de detalles de la orden y la salva
7	AddNew()	uctOrders1	clsOrd	Crea un nuevo registro en el recordset
8	rs_MoveComplete(ADODB.EventReasonEnum, ADODB.Error, ADODB.EventStatusEnum*, ADODB.Recordset)	clsOrd	clsOrd	Evento llamado despues del cambio de posición del recordset
9	UpdateBatch()	uctOrders1	clsOrd	Procedimiento de actualización por lotes

10	rs_WillChangeRecord(ADODB.EventReasonEnum, Long, ADODB.EventStatusEnum*, ADODB.Recordset)	clsOrd	clsOrd	Método llamado antes que uno o mas registros en el recordset cambien
11	ActualizarListas()	uctOrders1	uctOrders1	Actualiza las vistas de productos disponibles y productos seleccionados

Vista Lógica

Modelo de Datos

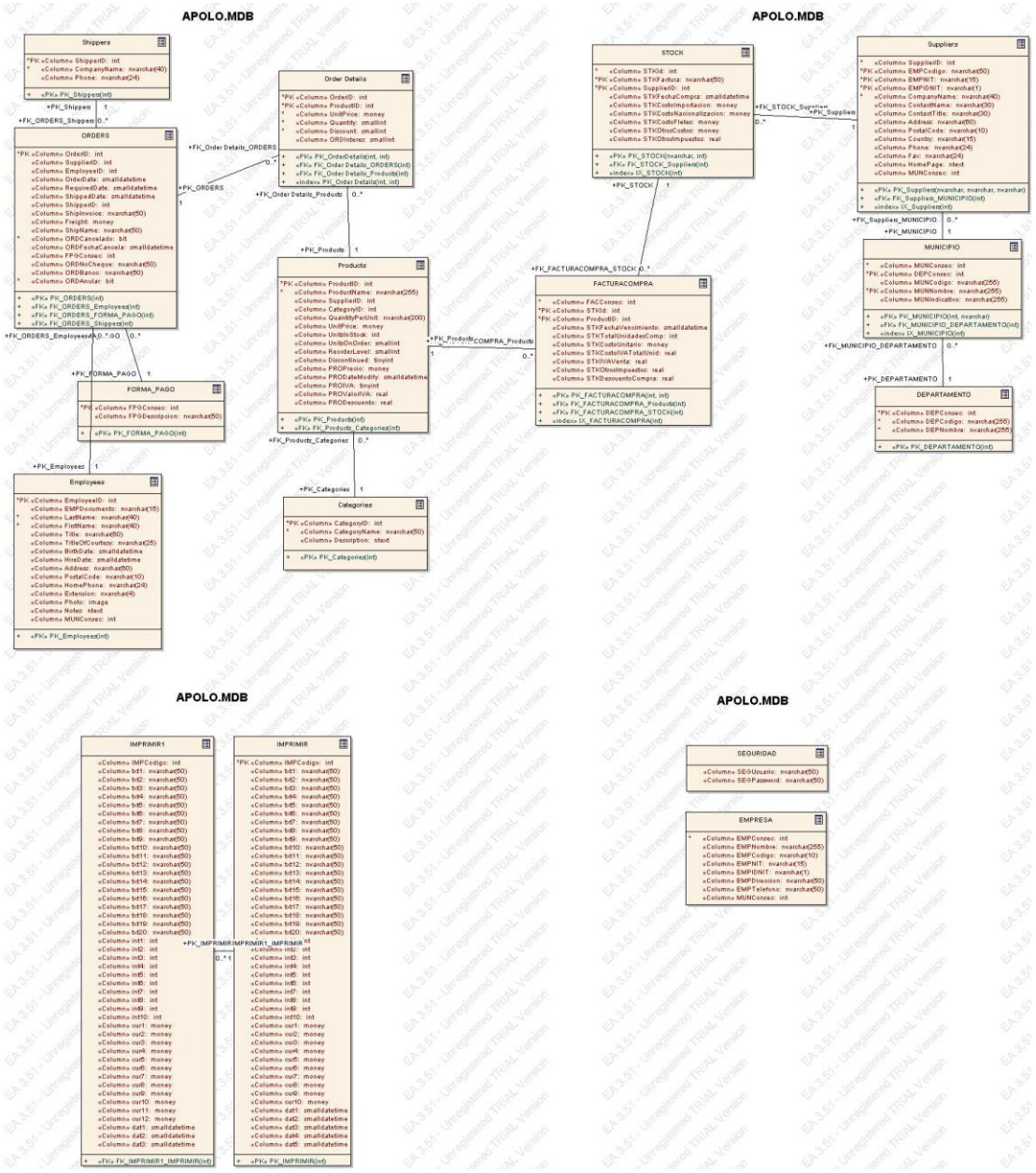


Fig. 67. Modelo de Datos

Diccionario de Datos

Categories

Tipo: *public «table»* **Class**
Estado: Terminado. Versión 1.0. Fase 1.0.
Paquete: DataModel

Almacena la información de las categorías

Conexiones

- Vínculo de Asociación con la Clase *Products*

Tabla 56. Categories Atributos

Atributo	Tipo	Notas
CategoryID	public: *int*	id de la categoría Primary Key; No Null;
CategoryName	public: *nvarchar*(50)	Nombre de la categoría No Null;
Description	public: *ntext*	Descripción

Tabla 57. Categories Métodos

Método	Tipo	Notas
PK_Categories (*int*)	«PK» public:	param: CategoryID [int - in]

DEPARTAMENTO

Tipo: *public «table»* **Class**
Estado: Terminado. Versión 1.0. Fase 1.0.
Paquete: DataModel

Almacena la información de los departamentos

Conexiones

- Vínculo de Asociación con la Clase *MUNICIPIO*

Tabla 58. DEPARTAMENTO Atributos

Atributo	Tipo	Notas
DEPConsec	Public: *int*	Id del departamento

		Primary Key; No Null;
DEPCodigo	Public: *nvarchar*(255)	Código del departamento No Null;
DEPNombre	Public: *nvarchar*(255)	Nombre del departamento No Null;

Tabla 59. DEPARTAMENTO Métodos

Método	Tipo	Notas
PK_DEPARTAMENTO (*int*)	«PK» public:	param: DEPConsec [int - in]

Employees

Tipo: *public «table»* **Class**
Estado: Terminado. Versión 1.0. Fase 1.0.
Paquete: DataModel

Almacena la información de los empleados

Conexiones

- Vínculo de Asociación con la Clase *ORDERS*

Tabla 60. Employees Atributos

Atributo	Tipo	Notas
EmployeeID	public: *int*	Numero autogenerado para cada nuevo empleado Primary Key; No Null;
EMPDocumento	public: *nvarchar*(15)	Documento de Identificación
LastName	public: *nvarchar*(40)	Apellidos No Null;
FirstName	public: *nvarchar*(40)	Nombre No Null;
Title	public: *nvarchar*(60)	título del empleado
TitleOfCourtesy	public: *nvarchar*(25)	Título usado en el saludo
BirthDate	public: *smalldatetime*	Fecha de nacimiento
HireDate	public: *smalldatetime*	Fecha de Ingreso del Empleado
Address	public: *nvarchar*(60)	Dirección
PostalCode	public: *nvarchar*(10)	Código postal
HomePhone	public: *nvarchar*(24)	Teléfono de la casa
Extension	public:	Número de extensión

	nvarchar(4)	
Photo	public: *image*	Foto del empleado
Notes	public: *ntext*	Notas del empleado
MUNConsec	public: *int*	Foreing Key Municipio

Tabla 61. Employees Métodos

Método	Tipo	Notas
PK_Employees (*int*)	«PK» public:	param: EmployeeID [int - in]

EMPRESA

Tipo: *public «table»* **Class**
Estado: Terminado. Versión 1.0. Fase 1.0.
Paquete: DataModel

Almacena la información de la empresa

Tabla 62. EMPRESA Atributos

Atributo	Tipo	Notas
EMPConsec	Public: *int*	Consecutivo de la Empresa No Null;
EMPNombre	Public: *nvarchar*(255)	Nombre de la Persona o la Empresa
EMPCodigo	Public: *nvarchar*(10)	Codigo de la Empresa
EMPNIT	Public: *nvarchar*(15)	Nit
EMPIDNIT	Public: *nvarchar*(1)	Identificador del Nit
EMPDireccion	Public: *nvarchar*(50)	Dirección de la Empresa
EMPTelefono	public: *nvarchar*(50)	Teléfonos de la Empresa
MUNConsec	public: *int*	Foreing Key con Municipios

FACTURACOMPRA

Tipo: *public «table»* **Class**
Estado: Terminado. Versión 1.0. Fase 1.0.
Paquete: DataModel

Almacena la información de los detalles de la factura de compra

Conexiones

- Mensaje de Asociación con la Tabla Primaria *STOCK*
- Mensaje de Asociación con la Tabla Primaria *Products*

Tabla 63. FACTURACOMPRA Atributos

Atributo	Tipo	Notas
FACConsec	public: *int*	Numero autogenerado para cada nueva factura de compra No Null;
STKId	public: *int*	Foreign Key Stock. Id d la factura Primary Key; No Null;
ProductID	public: *int*	Foreing Key Product ID Primary Key; No Null;
STKFechaVencimiento	public: *smalldatetime*	Fecha de Vencimiento del stock
STKTotalUnidadesComp	public: *int*	Número total de Unidades que se compraron
STKCostoUnitario	public: *money*	Costo Unitario a la compra
STKCostoIVATotalUnid	public: *real*	IVA al Número total de Unidades que se compraron
STKIVAVenta	public: *real*	IVA a la Venta
STKOtrosImpuestos	public: *real*	Otros Impuestos que se adicinan a la venta
STKDescuentoCompra	public: *real*	Descuento en la Compra

Tabla 64. FACTURACOMPRA Métodos

Método	Tipo	Notas
PK_FACTURACOMPRA (*int, int*)	«PK» public:	param: STKId [int - in] param: ProductID [int - in]
FK_FACTURACOMPRA_ Products (*int*)	«FK» public:	param: ProductID [int - in]
FK_FACTURACOMPRA_ STOCK (*int*)	«FK» public:	param: STKId [int - in]
IX_FACTURACOMPRA (*int*)	«index» public:	param: FACConsec [int - in]

FORMA_PAGO

Tipo: *public «table»* **Class**
Estado: Terminado. Versión 1.0. Fase 1.0.
Paquete: DataModel

Forma como se Cancela un pedido

Conexiones

- Vínculo de Asociación con la Clase *ORDERS*

Tabla 65. FORMA_PAGO Atributos

Atributo	Tipo	Notas
FPGConsec	public: *int*	Numero autogenerado para cada nueva forma de pago Primary Key; No Null;
FPGDescripcion	public: *nvarchar*(50)	Descripción de la forma de pago

Tabla 66. FORMA_PAGO Métodos

Método	Tipo	Notas
PK_FORMA_PAGO (*int*)	«PK» public:	param: FPGConsec [int - in]

IMPRIMIR

Tipo: *public «table»* **Class**
Estado: Terminado. Versión 1.0. Fase 1.0.
Paquete: DataModel

Tabla que permite la generación de reportes

Conexiones

- Vínculo de Asociación con la Clase *IMPRIMIR1*

Tabla 67. IMPRIMIR Atributos

Atributo	Tipo	Notas
IMPCodigo	public: *int*	Id de Imprimir Primary Key; No Null;
txt1	public: *nvarchar*(50)	Texto 1
txt2	public: *nvarchar*(50)	Texto 2
txt3	public: *nvarchar*(50)	Texto 3
txt4	public: *nvarchar*(50)	Texto 4
txt5	public: *nvarchar*(50)	Texto 5
txt6	public: *nvarchar*(50)	Texto 6
txt7	public: *nvarchar*(50)	Texto 7
txt8	public: *nvarchar*(50)	Texto 8
txt9	public:	Texto 9

	nvarchar(50)	
txt10	public: *nvarchar*(50)	Texto 10
txt11	public: *nvarchar*(50)	Texto 11
txt12	public: *nvarchar*(50)	Texto 12
txt13	public: *nvarchar*(50)	Texto 13
txt14	public: *nvarchar*(50)	Texto 14
txt15	public: *nvarchar*(50)	Texto 15
txt16	public: *nvarchar*(50)	Texto 16
txt17	public: *nvarchar*(50)	Texto 17
txt18	public: *nvarchar*(50)	Texto 18
txt19	public: *nvarchar*(50)	Texto 19
txt20	public: *nvarchar*(50)	Texto 20
int1	public: *int*	entero 1
int2	public: *int*	entero 2
int3	public: *int*	entero 3
int4	public: *int*	entero 4
int5	public: *int*	entero 5
int6	public: *int*	entero 6
int7	public: *int*	entero 7
int8	public: *int*	entero 8
int9	public: *int*	entero 9
int10	public: *int*	entero 10
cur1	public: *money*	Moneda 1
cur2	public: *money*	Moneda 2
cur3	public: *money*	Moneda 3
cur4	public: *money*	Moneda 4
cur5	public: *money*	Moneda 5
cur6	public: *money*	Moneda 6
cur7	public: *money*	Moneda 7
cur8	public: *money*	Moneda 8
cur9	public: *money*	Moneda 9
cur10	public: *money*	Moneda 10
dat1	public: *smalldatetime*	Fecha 1
dat2	public: *smalldatetime*	Fecha 2
dat3	public: *smalldatetime*	Fecha 3
dat4	public: *smalldatetime*	Fecha 4
dat5	public:	Fecha 5

	smalldatetime	

Tabla 68. IMPRIMIR Métodos

Método	Tipo	Notas
PK_IMPRIMIR (*int*)	«PK» public:	param: IMPCodigo [int - in]

IMPRIMIR1

Tipo: *public «table»* **Class**
Estado: Terminado. Versión 1.0. Fase 1.0.
Paquete: DataModel

Tabla que permite la generación de reportes

Conexiones

- Mensaje de Asociación con la Tabla Primaria *IMPRIMIR*

Tabla 69. IMPRIMIR1 Atributos

Atributo	Tipo	Notas
IMPCodigo	public: *int*	Id del codigo imprimir
txt1	public: *nvarchar*(50)	texto 1
txt2	public: *nvarchar*(50)	texto 2
txt3	public: *nvarchar*(50)	texto 3
txt4	public: *nvarchar*(50)	texto 4
txt5	public: *nvarchar*(50)	texto 5
txt6	public: *nvarchar*(50)	texto 6
txt7	public: *nvarchar*(50)	texto 7
txt8	public: *nvarchar*(50)	texto 8
txt9	public: *nvarchar*(50)	texto 9
txt10	public: *nvarchar*(50)	texto 10
txt11	public: *nvarchar*(50)	texto 11
txt12	public: *nvarchar*(50)	texto 12
txt13	public:	texto 13

	nvarchar(50)	
txt14	public: *nvarchar*(50)	texto 14
txt15	public: *nvarchar*(50)	texto 15
txt16	public: *nvarchar*(50)	texto 16
txt17	public: *nvarchar*(50)	texto 17
txt18	public: *nvarchar*(50)	texto 18
txt19	public: *nvarchar*(50)	texto 19
txt20	public: *nvarchar*(50)	texto 20
int1	public: *int*	Entero 1
int2	public: *int*	Entero 2
int3	public: *int*	Entero 3
int4	public: *int*	Entero 4
int5	public: *int*	Entero 5
int6	public: *int*	Entero 6
int7	public: *int*	Entero 7
int8	public: *int*	Entero 8
int9	public: *int*	Entero 9
int10	public: *int*	Entero 10
cur1	public: *money*	Moneda 1
cur2	public: *money*	Moneda 2
cur3	public: *money*	Moneda 3
cur4	public: *money*	Moneda 4
cur5	public: *money*	Moneda 5
cur6	public: *money*	Moneda 6
cur7	public: *money*	Moneda 7
cur8	public: *money*	Moneda 8
cur9	public: *money*	Moneda 9
cur10	public: *money*	Moneda 10
cur11	public: *money*	Moneda 11
cur12	public: *money*	Moneda 12
dat1	public: *smalldatetime*	Fecha 1
dat2	public: *smalldatetime*	Fecha 2
dat3	public: *smalldatetime*	Fecha 3

Tabla 70. IMPRIMIR1 Métodos

Método	Tipo	Notas
FK_IMPRIMIR1_IMPRIMIR (*int*)	«FK» public:	param: IMPCodigo [int - in]

MUNICIPIO

Tipo: *public «table»* **Class**
Estado: Terminado. Versión 1.0. Fase 1.0.
Paquete: DataModel

Almacena la información de los municipios

Conexiones

- Vínculo de Asociación con la Clase *Suppliers*
- Mensaje de Asociación con la Tabla Primaria *DEPARTAMENTO*

Tabla 71. MUNICIPIO Atributos

Atributo	Tipo	Notas
MUNConsec	public: *int*	Número automáticamente asignado a un nuevo municipio No Null;
DEPConsec	public: *int*	Id del departamento Primary Key; No Null;
MUNCodigo	public: *nvarchar*(255)	Código del municipio
MUNNombre	public: *nvarchar*(255)	Nombre del municipio Primary Key; No Null;
MUNIndicativo	public: *nvarchar*(255)	Indicativo del municipio

Tabla 72. MUNICIPIO Métodos

Método	Tipo	Notas
PK_MUNICIPIO (*int, nvarchar*)	«PK» public:	param: DEPConsec [int - in] param: MUNNombre [nvarchar - in]
FK_MUNICIPIO_DEPARTAMENTO (*int*)	«FK» public:	param: DEPConsec [int - in]
IX_MUNICIPIO (*int*)	«index» public:	param: MUNConsec [int - in]

Order Details

Tipo: *public «table»* **Class**
Estado: Terminado. Versión 1.0. Fase 1.0.
Paquete: DataModel

Almacena la información de los detalles de la factura de venta

Conexiones

- Mensaje de Asociación con la Tabla Primaria *Products*
- Mensaje de Asociación con la Tabla Primaria *ORDERS*

Tabla 73. Order Details Atributos

Atributo	Tipo	Notas
OrderID	public: *int*	Id de la orden Primary Key; No Null;
ProductID	public: *int*	id del producto Primary Key; No Null;
UnitPrice	public: *money*	Precio unitario No Null;
Quantity	public: *smallint*	Cantidad No Null;
Discount	public: *smallint*	Descuento No Null;
ORDInteres	public: *smallint*	Impuesto a aplicar

Tabla 74. Order Details Métodos

Método	Tipo	Notas
PK_OrderDetails (*int, int*)	«PK» public:	param: OrderID [int - in] param: ProductID [int - in]
FK_Order Details_ORDERS (*int*)	«FK» public:	param: OrderID [int - in]
FK_Order Details_Products (*int*)	«FK» public:	param: ProductID [int - in]
PK_Order Details (*int, int*)	«index» public:	param: OrderID [int - in] param: ProductID [int - in]

ORDERS

Tipo: *public «table»* **Class**
Estado: Terminado. Versión 1.0. Fase 1.0.
Paquete: DataModel

Almacena la información de la factura de venta

Conexiones

- Mensaje de Asociación con la Tabla Primaria *Shippers*
- Mensaje de Asociación con la Tabla Primaria *FORMA_PAGO*
- Mensaje de Asociación con la Tabla Primaria *Employees*

- Vínculo de Asociación con la Clase *Order Details*

Tabla 75. ORDERS Atributos

Atributo	Tipo	Notas
OrderID	Public: *int*	Número automáticamente asignado a una nueva orden Primary Key; No Null;
SupplierID	Public: *int*	Id del cliente
EmployeeID	Public: *int*	Id del empleado que vende el producto
OrderDate	public: *smalldatetime*	Fecha de la orden
RequiredDate	public: *smalldatetime*	Fecha requerida
ShippedDate	public: *smalldatetime*	Fecha de embarque
ShipperID	Public: *int*	Id del embarcador
ShipInvoice	public: *nvarchar*(50)	Costo del embarque
Freight	Public: *money*	Costo del transporte
ShipName	public: *nvarchar*(50)	
ORDCancelado	Public: *bit*	Si se ha pagado o no la factura No Null;
ORDFechaCancela	public: *smalldatetime*	Fecha que se paga la factura
FPGConsec	Public: *int*	
ORDNoCheque	public: *nvarchar*(50)	Número del cheque
ORDBanco	public: *nvarchar*(50)	banco al cual pertenece el cheque
ORDAnular	Public: *bit*	Anular o No la Factura No Null;

Tabla 76. ORDERS Métodos

Método	Tipo	Notas
PK_ORDERS (*int*)	«PK» public:	param: OrderID [int - in]
FK_ORDERS_Employees (*int*)	«FK» public:	param: EmployeeID [int - in]
FK_ORDERS_FORMA_PAGO (*int*)	«FK» public:	param: FPGConsec [int - in]
FK_ORDERS_Shippers (*int*)	«FK» public:	param: ShipperID [int - in]

Products

Tipo: *public «table»* **Class**
Estado: Terminado. Versión 1.0. Fase 1.0.
Paquete: DataModel

Almacena la información de los productos

Conexiones

- Mensaje de Asociación con la Tabla Primaria *Categories*
- Vínculo de Asociación con la Clase *Order Details*
- Vínculo de Asociación con la Clase *FACTURACOMPRA*

Tabla 77. Products Atributos

Atributo	Tipo	Notas
ProductID	public: *int*	Número automáticamente asignado a un nuevo producto Primary Key; No Null;
ProductName	public: *nvarchar*(255)	Nombre del producto No Null;
SupplierID	public: *int*	Id del proveedor
CategoryID	public: *int*	Id de la categoría del producto
QuantityPerUnit	public: *nvarchar*(200)	Cantidad del elemento por unidad
UnitPrice	public: *money*	Precio unitario
UnitsInStock	public: *int*	Unidades en stock
UnitsOnOrder	public: *smallint*	Unidades en la orden
ReorderLevel	public: *smallint*	Mínimo número de unidades a mantener en stock
Discontinued	public: *tinyint*	Determina si el producto esta o no descontinuado
PROPrecio	public: *money*	Precio
PRODateModify	public: *smalldatetime*	Fecha de última modificación
PROIVA	public: *tinyint*	Si El producto causa o no IVA
PROValorIVA	public: *real*	% del Valor del Iva
PRODescuento	public: *real*	% de descuento

Tabla 78. Products Métodos

Método	Tipo	Notas
PK_Products (*int*)	«PK» public:	param: ProductID [int - in]
FK_Products_Categories (*int*)	«FK» public:	param: CategoryID [int - in]

SEGURIDAD

Tipo: *public «table»* **Class**
Estado: Terminado. Versión 1.0. Fase 1.0.

Paquete: DataModel

Almacena la información de los usuarios registrados en el sistema

Tabla 79. SEGURIDAD Atributos

Atributo	Tipo	Notas
SEGUsuario	public: *nvarchar*(50)	Nombre del usuario
SEGPassword	public: *nvarchar*(50)	Password

Shippers

Tipo: *public «table»* **Class**
Estado: Terminado. Versión 1.0. Fase 1.0.
Paquete: DataModel

Almacena la información de los transportadores o embarcadores de mercancía

Conexiones

- Vínculo de Asociación con la Clase *ORDERS*

Tabla 80. Shippers Atributos

Atributo	Tipo	Notas
ShipperID	public: *int*	Número automáticamente asignado a un transportador Primary Key; No Null;
CompanyName	public: *nvarchar*(40)	Nombre de la compañia No Null;
Phone	public: *nvarchar*(24)	Teléfono de la compañia

Tabla 81. Shippers Métodos

Método	Tipo	Notas
PK_Shippers (*int*)	«PK» public:	param: ShipperID [int - in]

STOCK

Tipo: *public «table»* **Class**
Estado: Terminado. Versión 1.0. Fase 1.0.
Paquete: DataModel

Almacena la información de la factura de compra

Conexiones

- Mensaje de Asociación con la Tabla Primaria *Suppliers*
- Vínculo de Asociación con la Clase *FACTURACOMPRA*

Tabla 82. STOCK Atributos

Atributo	Tipo	Notas
STKId	public: *int*	Número automáticamente asignado a la factura de compra No Null;
STKFactura	public: *nvarchar*(50)	Factura de Compra Primary Key; No Null;
SupplierID	public: *int*	Foreing Key Suppliers Primary Key; No Null;
STKFechaCompra	public: *smalldatetime*	Fecha de Compra de la Mercancia
STKCostoImportacion	public: *money*	Costo de la Importación
STKCostoNacionalizacion	public: *money*	Costo de la Nacionalizacion de la Mercancía
STKCostoFletes	public: *money*	Costo de los Fletes
STKOtrosCostos	public: *money*	Otros Costos
STKOtrosImpuestos	public: *real*	Otros Impuestos que se adicinan a la venta

Tabla 83. STOCK Métodos

Método	Tipo	Notas
PK_STOCK (*nvarchar, int*)	«PK» public:	param: STKFactura [nvarchar - in] param: SupplierID [int - in]
FK_STOCK_Suppliers (*int*)	«FK» public:	param: SupplierID [int - in]
IX_STOCK (*int*)	«index» public:	param: STKId [int - in]

Suppliers

Tipo: *public «table»* **Class**
Estado: Terminado. Versión 1.0. Fase 1.0.
Paquete: DataModel

Almacena la información de los proveedores

Conexiones

- Mensaje de Asociación con la Tabla Primaria *MUNICIPIO*
- Vínculo de Asociación con la Clase *STOCK*

Tabla 84. Suppliers Atributos

Atributo	Tipo	Notas
SupplierID	Public: *int*	Número automáticamente asignado a un proveedor No Null;
EMPCodigo	public: *nvarchar*(50)	Código de la Empresa Primary Key; No Null;
EMPNIT	public: *nvarchar*(15)	Nit de la Empresa Primary Key; No Null;
EMPIDNIT	public: *nvarchar*(1)	Identificador del Nit Primary Key; No Null;
CompanyName	public: *nvarchar*(40)	Nombre de la compañia No Null;
ContactName	public: *nvarchar*(30)	Nombre del contacto
ContactTitle	public: *nvarchar*(30)	Título del contacto
Address	public: *nvarchar*(60)	Dirección
PostalCode	public: *nvarchar*(10)	Código postal
Country	public: *nvarchar*(15)	Pais
Phone	public: *nvarchar*(24)	Telefono
Fax	public: *nvarchar*(24)	Fax
HomePage	Public: *ntext*	Página WEB
MUNConsec	Public: *int*	Consecutivo del Departamento

Tabla 85. Suppliers Métodos

Método	Tipo	Notas
PK_Suppliers (*nvarchar, nvarchar, nvarchar*)	«PK» public:	param: EMPCodigo [nvarchar - in] param: EMPNIT [nvarchar - in] param: EMPIDNIT [nvarchar - in]
FK_Suppliers_MUNICIPIO (*int*)	«FK» public:	param: MUNConsec [int - in]
IX_Suppliers (*int*)	«index» public:	param: SupplierID [int - in]

Modelo Lógico

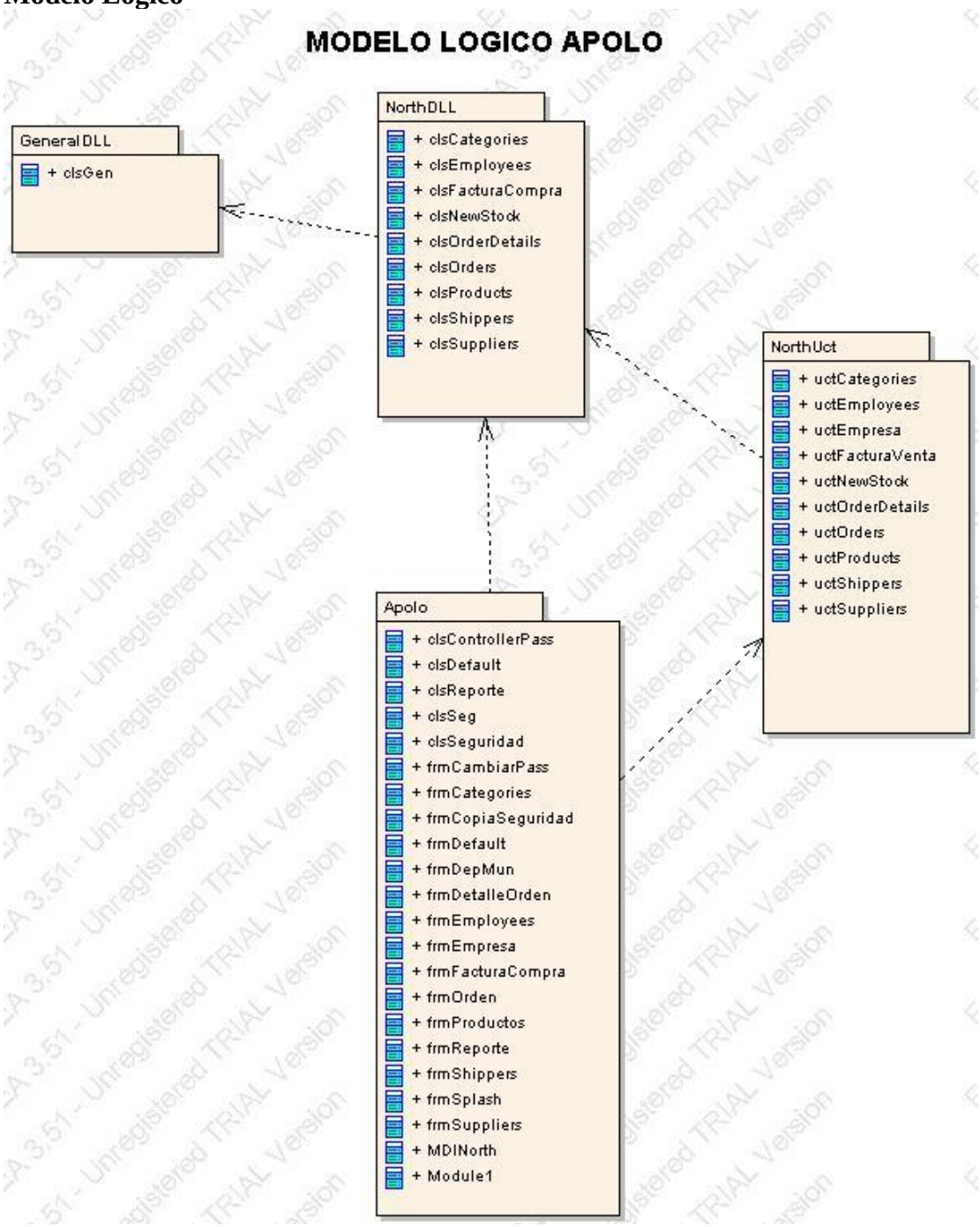


Fig. 68. Modelo Lógico

Apolo

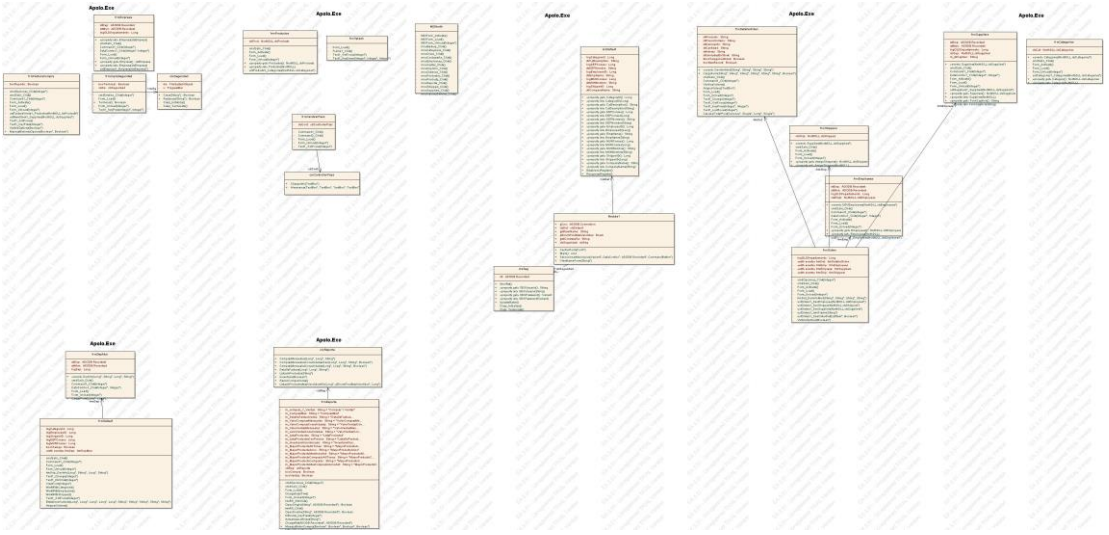


Fig. 69. Apolo

GeneralDLL

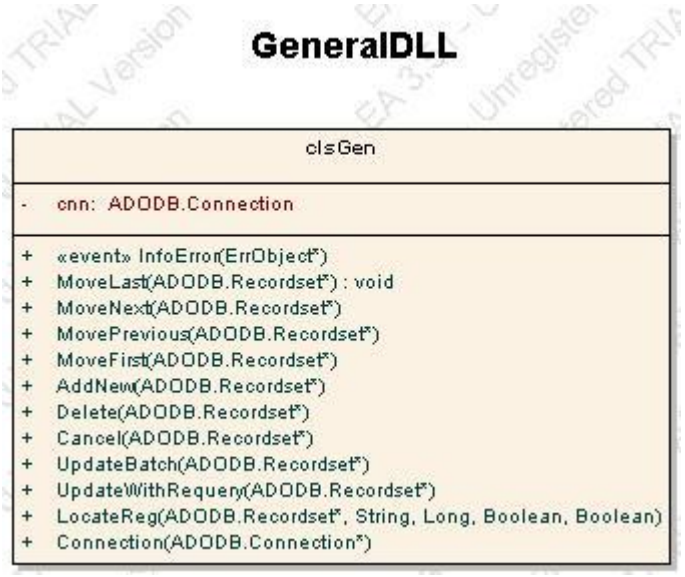


Fig. 70. GeneralDLL

NorthDLL

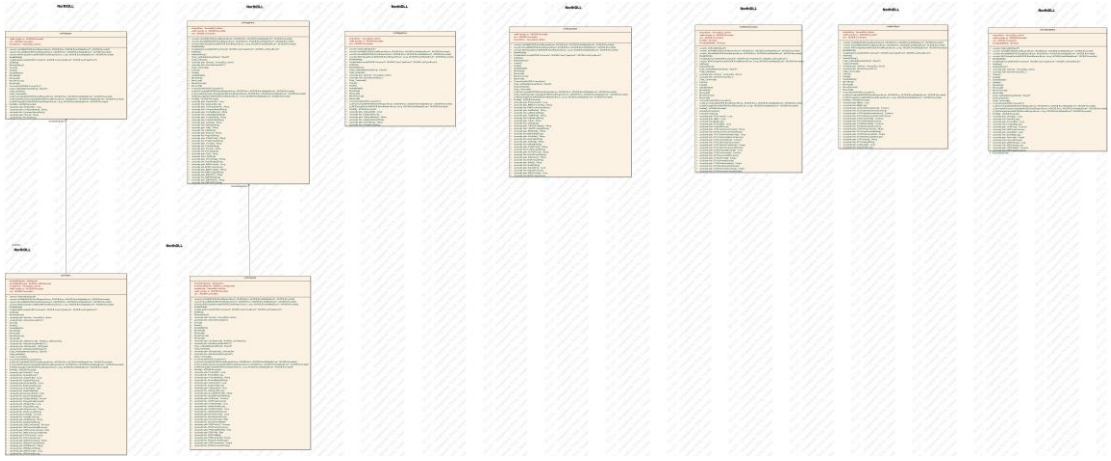


Fig. 71. NorthDLL

NorthUct

Fig. 1. NortUct

Vista de Componentes

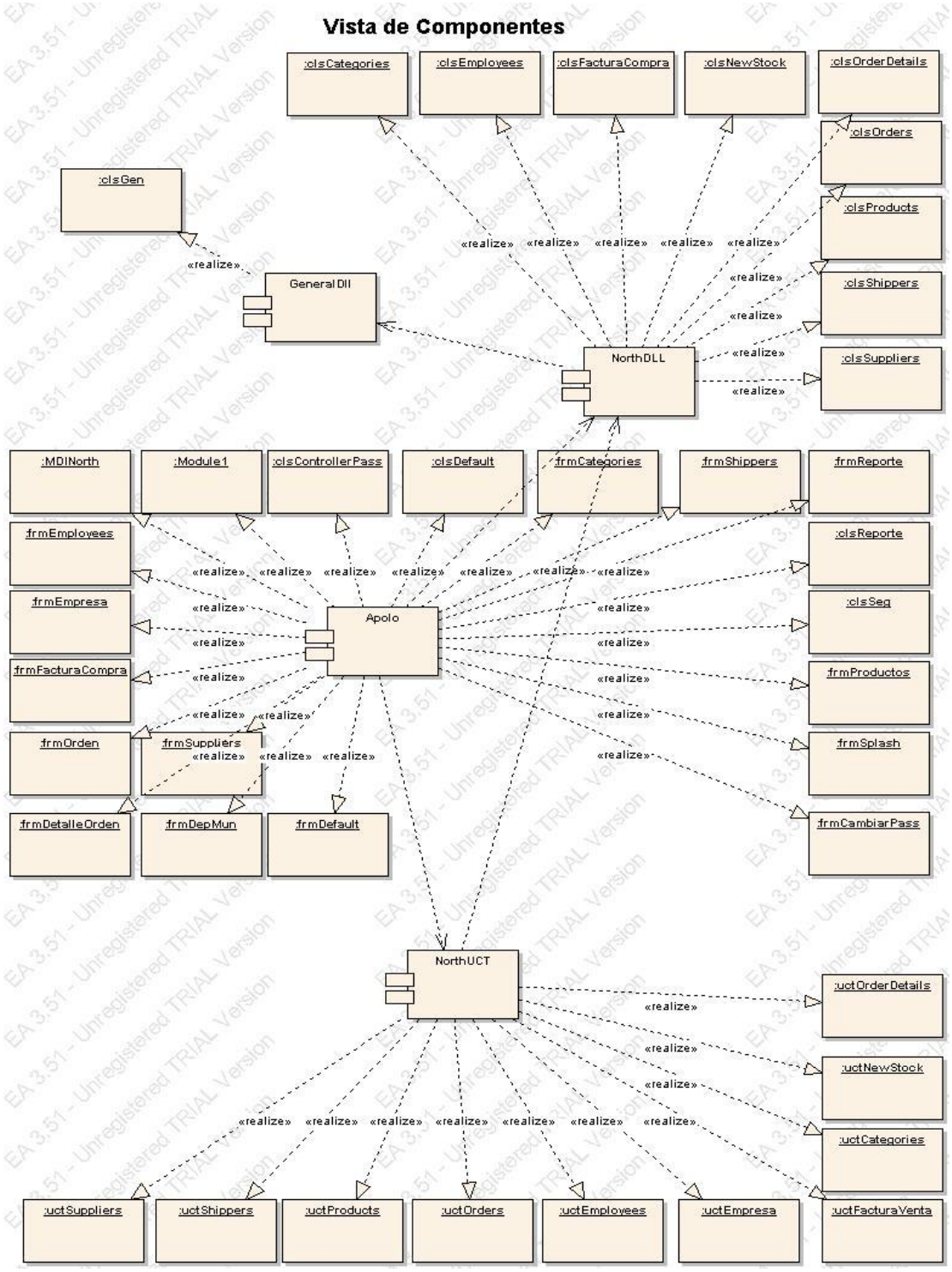


Fig. 72. Modelo de Componentes

Apolo

Tipo: *public* **Component**
Estado: Terminado. Versión 1.0. Fase 1.0.
Paquete: Component Model

Ejecutable principal de la aplicación

GeneralDll

Tipo: *public* **Component**
Estado: Terminado. Versión 1.0. Fase 1.0.
Paquete: Component Model

Objeto manejador del recordset

NorthDLL

Tipo: *public* **Component**
Estado: Terminado. Versión 1.0. Fase 1.0.
Paquete: Component Model

Objetos del Negocio

NorthUCT

Tipo: *public* **Component**
Estado: Terminado. Versión 1.0. Fase 1.0.
Paquete: Component Model

Controles de Usuario

Vista de Despliegue

Modelo de Despliegue

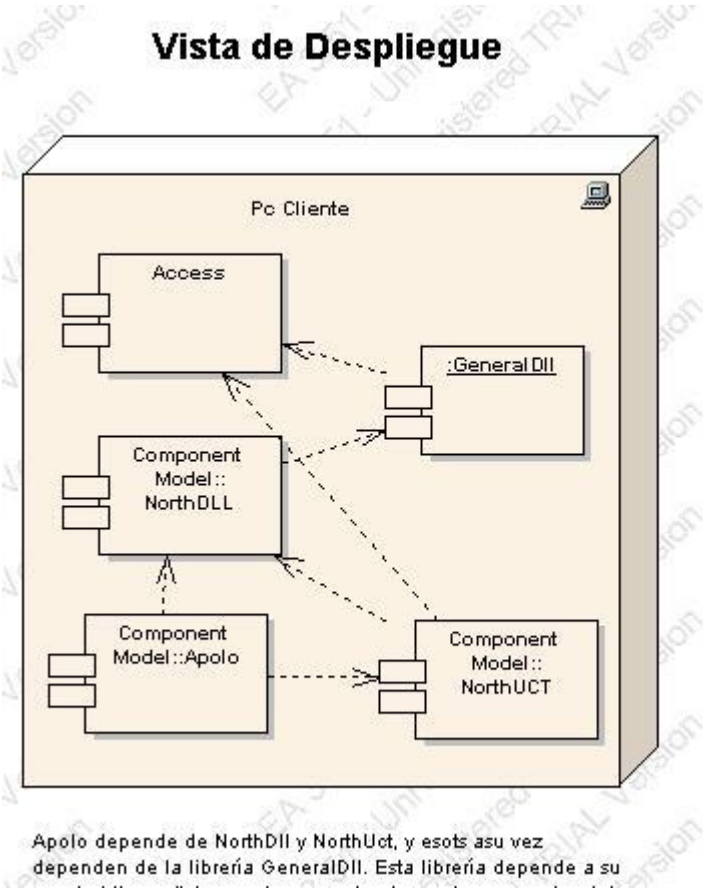


Fig. 73. Modelo de Despliegue

Pc Cliente

Tipo: *public «pc client»* **Node**
Estado: Terminado. Versión 1.0. Fase 1.0.
Paquete: Deployment Model

Máquina Cliente

- Software = Access 2000.
- Software = MDACTYP.EXE.
- Software = Windows 2000.

4.2.1.1. Diagrama de Hipo

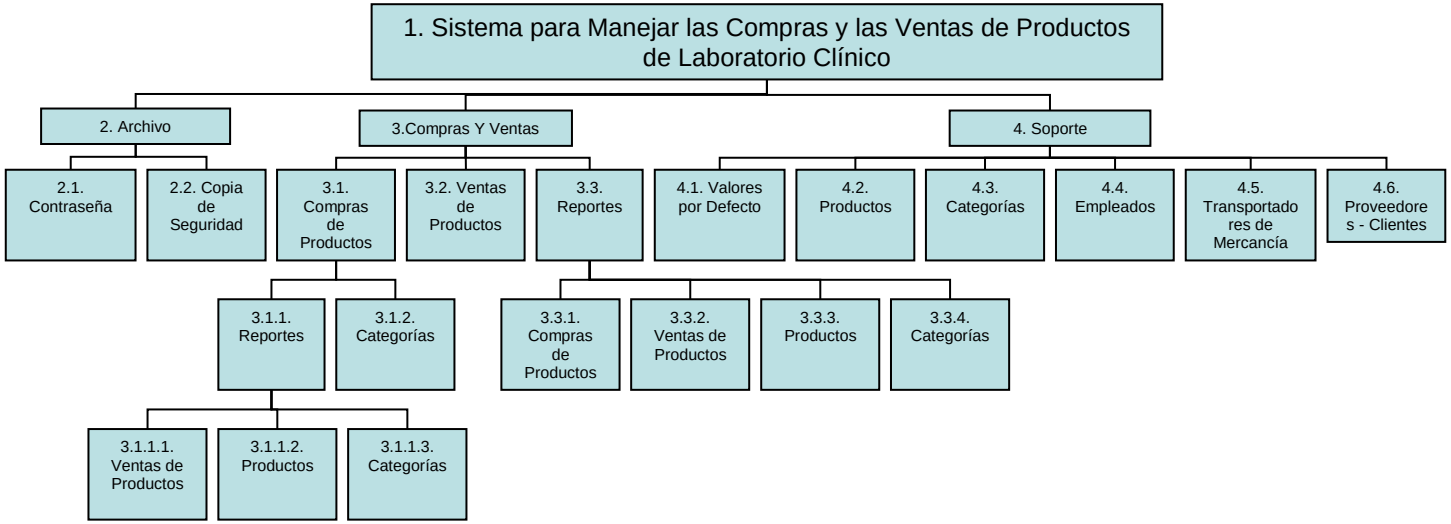


Fig. 74. Diagrama de HIPO

- 1. Sistema para manejar las Compras y las Ventas de Productos de Laboratorio Clínico: El sistema permitirá capturar la información de las compras efectuadas, las ventas realizadas, y generará una serie de reportes relacionados con las mismas
- 2. Archivo: Tiene como función permitir al actor identificarse en el sistema y efectuar la copia de seguridad del mismo.
  2.1. Contraseña: Permite cambiar la contraseña del usuario del sistema
  2.2. Copia de Seguridad: Comprime la base de datos y la almacena en una ubicación seleccionada por el usuario o permite restaurarla desde esta ubicación
- 3. Compras y Ventas: Permite al actor ingresar la información de las compras y las ventas ocurridas
  3.1. Compras de Productos: Permite almacenar la información de los productos comprados. Permite desplegar las pantallas de reportes y categorías
  3.2. Ventas de Productos: Captura la información de los productos vendidos.
  3.3. Reportes: Despliega los reportes del sistema. Despliega las pantallas de Compras y ventas de productos, productos y categorías
- 4. Soporte: Permite al actor actualizar la información general del sistema
  4.1. Valores por Defecto: Permite establecer los valores por defecto con los cuales trabajará el sistema.
  4.2. Productos: Permite Actualizar, modificar, agregar información de los productos
  4.3. Categorías: Permite Actualizar, modificar, agregar información de las categorías
  4.4. Empleados: Permite Actualizar, Modificar, Agregar información de los empleados de la empresa
  4.5. Transportadores de Mercancía: Permite Actualizar, modificar, agregar información de las empresas que distribuyen los productos
  4.6. Proveedores – Cliente: Permite Actualizar, modificar, agregar información de los proveedores – clientes de la empresa

4.3. Implantación

Para la implantación del sistema primero se instalará el sistema, siguiendo el método de conversión paralelo, en el cual, los usuarios continuarán usando el sistema anterior mientras utilizan el nuevo.

Instalado el sistema se llevará a cabo una capacitación “en casa”, sobre el uso del mismo, tanto a los operadores como a los usuarios del mismo.

4.3.1. Capacitación

La capacitación será llevada a cabo en las instalaciones de la empresa (“en Casa”). Se utilizará como guía el Manual del Usuario, en el caso de los usuarios, y el Manual técnico para los operadores del sistema

4.3.1.1. Capacitacion De Operadores

- Instalación del sistema
- Copia de Seguridad del Sistema
- Operación del sistema.
- Captura y manejo de los datos
- Diagrama Entidad – Relación
- Tablas de La base de Datos (Diccionario de Datos)
- Objetos del Sistema
- Ejecutables del Sistema
- Mensajes dentro del sistema
- Reutilización de Librerías y controles del sistema

4.3.1.2. Capacitacion De Usuarios

- Compras de productos
- Ventas de productos
- Generación de Factura
- Trabajar con categorías
- Trabajar con Empleados
- Trabajar con Proveedores – Clientes
- Transportadores
- Valores por defecto.
- Uso de reportes

En Cada uno de estos items se enseñará:

- La Búsqueda de elementos, y creación de filtros de búsqueda
- El agregar nuevos elementos
- La Modificación de Elementos
- La Eliminación de Elementos
- La cancelación de operaciones
- Comó editar datos
- Generación de Factura
- Generación de Reportes

4.3.2. Conversión De Datos

El sistema que se utilizará para llevar a cabo la conversión de los datos será el paralelo.

En este, tanto el sistema antiguo (Manual) como el nuevo convivirán, garantizando, que en el caso de errores o problemas de procesamiento, pueda continuar en funcionamiento la empresa.

4.3.3. Plan De Conversión

- Verificación del Hardware y Software
- Instalación del sistema
- Capacitación a operadores
- Capacitación a Usuarios
- Carga del sistema.
    - Inserción de Productos
    - Captura de Categorías
    - Captura de Empleados
    - Captura de Compras y Proveedores a partir de las facturas de Compras
    - Captura de ventas, Proveedores, transportadores y clientes a partir de las facturas de ventas
    - Evaluación de los reportes, para determinar coherencia de la información y fiabilidad y confiabilidad del sistema

4.3.4. Acondicionamiento De Las Instalaciones

Se revisará el cumplimiento por parte del hardware y el software de especificaciones, tal y como se estipula en el manual del usuario y el manual técnico.

4.3.5. Preparación de Datos y Archivos

Una vez Instalado el sistema y llevado a cabo la capacitación, se procederá al registro de categorías, productos, proveedores - clientes, facturas de ventas y de compra, y se generarán los primeros reportes del sistema para determinar el estado a la fecha de compras y ventas de la empresa

4.4. Fase De Puesta En Marcha

Se llevará a cabo la pruebas de Casos de Uso del Sistema. Para ello se determinará que el sistema cumpla con las especificaciones propuestas.

PRUEBAS DEL SISTEMA

- **Lugar: Pruebas en Casa**
- **Hardware: Pentium II, 400 Mhz, Disco Duro 10 GB, Monitor 17”**
- **Software: Windows 98 2Ed.**
-

Caso de Uso	Fecha	Descripción	Resultado
Trabajar con Apolo {Basic Path}.	01/11/2004	1. El Actor accede al sistema 2. El sistema despliega la pantalla de inicio. Pregunta el usuario y la contraseña 3. El actor ingresa el usuario y la contraseña 4. El actor ingresa a las pantallas del sistema 5. El actor finaliza la sesión	CONFORME
Ingresar al Sistema	01/11/2004	1. El actor inicia Apolo 2. El sistema despliega la pantalla de bienvenida 3. El actor ingresa el Usuario y la contraseña y procede a ingresar (Enter) 4. El sistema establece la conexión con la persistencia 5. El sistema valida el usuario y la contraseña. 6. El sistema despliega el formulario Principal 7. El sistema crea el objeto que maneja la información por defecto del sistema 8. El sistema recupera la	CONFORME

		información por defecto almacenada en la persistencia	
Establer seguridad	01/11/2004	1. El actor ingresa a la pantalla de Cambiar Password 2. El sistema despliega la información del usuario 4. El sistema despliega la pantalla de Cambiar Password 5. El actor procede a cambiar: - El usuario, - Ingresa la contraseña anterior (Sirve para Verificar que el que hace el cambio, si es el usuario que ingresó a la aplicación) - Ingresa la contraseña nueva - Confirma la contraseña 6. El actor procede a efectuar los cambios (Click en aceptar) 7. El sistema valida los cambios 8. El sistema almacena la información 9. El sistema descarga la pantalla	CONFORME
Trabajar con Categorías (WorkWithCategories)	01/11/2004	1. El actor desea cambiar las categorías 2. Crea el objeto categorías 3. Se crea el objeto de enlace a la persistencia 4. Se asigna la conexión global 5. Se trae la información de la base de datos 6. Se asigna el objeto creado y abierto al formulario 7. Se llama al formulario categorías	CONFORME

llamar categorías desde otro formulario	01/11/2004	1. Se inicializa el control de categorías 2. El control crea el objeto Categoria por defecto 3. Se inicia la carga del formulario 4. El control es un hijo 5. Se asigna el objeto de categorias creado al control 6. Se carga el control 7. Se asigna la fuente de datos 8. Se le da formato a la grid del usuario	CONFORME
Agregar Una nueva Categoría	01/11/2004	El actor desea agregar una nueva categoría 1. El actor selecciona el botón nuevo 2. El sistema le dice a la clase categorías que se va a adicionar un registro 3. El actor ingresa la información solicitada por el sistema 4. El actor Guarda la información	CONFORME
Eliminar Categoría	01/11/2004	Continúa, seleccionando el botón eliminar 2. El sistema determina si el usuario desea eliminar el registro. En caso afirmativo, se actualizan los campos, y los botones	CONFORME
Buscar Categorías	01/11/2004	1. El actor desea buscar información de categorías de productos. El sistema despliega el menú de búsqueda	CONFORME

		2. El actor selecciona una de las opciones del menú 3. El sistema comienza la búsqueda de la información 4. Se Actualizan los campos y los botones de comando (Nuevo, Agregar, Eliminar, etc) 6. Actualiza la grid de categorías 7. El sistema devuelve control al actor	
Salvar Información	01/11/2004	El actor desea Salvar nueva categoría (Nueva o modificada) 1. El actor Selecciona Guardar 2. El sistema Almacena la información	CONFORME
Trabajar con empleados	01/11/2004	1. El actor puede Agregar Empleados. Incluye UCA13. Agregar Empleados 2. El actor puede Eliminar empleados. Incluye UCA15. Eliminar Empleados 3. El actor puede Buscar empleados. Incluye UCA 25. Salvar Empleados 5. El actor puede Salvar la información nueva o modificada. Incluye UCA 14. Salvar Empleados	CONFORME
Llamar Empleados	01/11/2004	1. El actor oprime una tecla cualquiera 2. El sistema despliega el formulario de Emplados	CONFORME
Agregar Empleados	01/11/2004	El actor desea agregar un nuevo registro	CONFORME

		1. El actor presiona el botón Nuevo 2. El sistema dispone los controles y los botones para el ingreso y captura de la información 3. Una vez ingresados los datos, el sistema se dispone a salvar la información	
Eliminar Empleados	01/11/2004	El actor selecciona un registro y procede a su eliminación 1. El actor selecciona el botón de eliminar 2. El sistema informa a la clase empleados interna que debe eliminar un registro 3. El sistema actualiza la información en la GUI	CONFORME
Buscar Empleados	01/11/2004	El actor va a buscar por los apellidos de los empleados 1. El actor busca. Selecciona el boton buscar 2. Se actualizan los controles y los botones de comando. Se despliega la información solicitada	CONFORME
Salvar Empleados	01/11/2004	El actor procede a salvar la información (nueva o modificada) 1. El actor oprime la tecla salvar 2. El sistema le informa a la clase que proceda a salvar la información 3. El sistema almacena la información	CONFORME

		4. El sistema actualiza controles de edición y botones de comando	
Ingresar a Productos	01/11/2004	1. El actor selecciona el menú de productos 2. El sistema carga el formulario y despliega la información de los productos	CONFORME
Llamar a Categorías	01/11/2004	1. El actor desea observar las categorías del producto (presiona una tecla en el campo de categorías) 2. Se inicializa el formulario de categorias con la información de la categoría 3. Se devuelve la información al formulario principal	CONFORME
Salvar Productos	01/11/2004	1. El actor selecciona el botón salvar 2. el control informa a la clase que debe almacenar la información 3. Se calculan los precios del producto	CONFORME
Agregar Productos	01/11/2004	1. El actor selecciona el botón nuevo 2. se carga la información necesaria a partir de la persistencia 3. Se calculan los precios	CONFORME
Eliminar Producto	01/11/2004	1. El actor selecciona el botón eliminar 2. El sistema confirma las intenciones del actor	CONFORME

		3. El actor confirma la eliminación 4. El sistema elimina el producto. Se formatean las columnas de la grid para la salida 5. Se calculan los precios	
Ingresar a proveedores	01/11/2004	1. El actor selecciona del menú la opción de proveedores 2. El sistema despliega la pantalla de proveedores	CONFORME
Nuevo Proveedor	01/11/2004	1. El actor selecciona el botón nuevo 2. El sistema limpia los controles para permitir el ingreso de nuevos datos	CONFORME
Eliminar	01/11/2004	1. El actor selecciona eliminar proveedor 2. el sistema elimina el proveedor	CONFORME
Salvar	01/11/2004	1. El actor selecciona Salvar proveedor 2. El sistema salva la información del proveedor	CONFORME
Nuevo Transportador	01/11/2004	1. El actor selecciona Nuevo proveedor 2. El sistema limpia los controles para permitir elñ ingreso del nuevo transportador	CONFORME
Eliminar Transportador	01/11/2004	1. El actor selecciona Eliminar proveedor 2. El sistema verificar las intenciones del actor	CONFORME

		3. El sistema Elimina el proveedor	
Ingresar a Proveedores desde el Menú	01/11/2004	1. El actor selecciona Ingresar a transportadores 2. Se llama al formulario de proveedores 3. se inicializa el control de proveedore, y su clase 4. Se carga el formulario 5. Se carga el control 6. Se carga la información de la persistencia	CONFORME
Salvar	01/11/2004	1. El actor selecciona el botón salvar 2. El sistema salva la información del transportador	CONFORME
Buscar Transportador	01/11/2004	1. El actor escribe los criterios de búsqueda y selecciona el botón Buscar 2. El sistema busca la información del transportador y la despliega	CONFORME
Pasar a "Registrar Factura"	01/11/2004	El actor desea ingresar una nueva factura - El actor pasa a siguiente - El sistema asigna el año - Se despliega el formulario de ingreso de facturas de compra	CONFORME
Buscar Factura de Compra	01/11/2004	El actor desea buscar una factura de compra	CONFORME

		1. El actor ingresa la información a buscar 2. El actor hace click en buscar 3. El control llama a su clase asociada, para que esta busque la información en la persistencia 4. Se despliega la información en la pantalla	
Nueva Mercancía	01/11/2004	1. El actor selecciona Nuevo 2. El sistema limpia los campos para permitir el ingreso de la nueva mercancía	CONFORME
Eliminar Productos	01/11/2004	1. El actor selecciona Eliminar 2. El sistema confirma las intenciones del usuario 3. Se carga la información de la factura 4. Se carga la información de la grid	CONFORME
Buscar Productos	01/11/2004	El actor desea ver el detalle de la factura 1. El actor selecciona siguiente. 2. El sistema busca la información de la factura y la despliega	CONFORME
Nueva compra de productos	01/11/2004	El actor desea agregar una nueva compra de mercancía 1. El actor hace click en el botón nuevo 2. El sistema limpia los campos para permitir el ingreso del producto comprado	CONFORME

lamar a proveedores	01/11/2004	El actor desea asignar el proveedor 1. El actor selecciona el campo de proveedore 2. El actor ingresa una letra cualquiera 3. El sistema despliega el formulario proveedor	CONFORME
Salvar Factura	01/11/2004	El actor procede a salvar la factura 1. El actor hace click en el botón de salvar 2. El sistema almacena la información	CONFORME
Eliminar Factura	01/11/2004	El actor desea eliminar la factura de compra 1. El actor hace click en eliminar la factura 2. El sistema confirma las intenciones del usuario y procede a Eliminar la factura 3. El sistema actualiza el control	CONFORME
Desplegar Productos	01/11/2004	El actor procede a desplegar los productos, para seleccionar, ver o encontrar alguno en especial - El actor oprime una tecla - El sistema despliega el formulario de productos - El actor busca el producto - El actor cierra el formulario - El sistema despliega el	CONFORME

		producto seleccionado por el usuario	
Salvar Productos	01/11/2004	El actor procede a salvar el detalle de la factura - El actor hace click en salvar - El control llama al método de actualización - Se actualiza el inventario	CONFORME
Nueva Venta	01/11/2004	El actor inicia una nueva venta de mercancia 1. El actor selecciona nuevo 2. El sistema despliega el formulario que permite ingresar nuevos productos para su venta	CONFORME
Modificar Venta	01/11/2004	El actor desea modificar la información de una venta 1. El actor cambia la información de la factura de venta 2. El actor almacena la información 3. El sistema actualiza la clase de detalles de la orden y la salva 4. Se actualiza el stock	CONFORME
Eliminar Venta	01/11/2004	El actor desea eliminar un producto de la venta 1. El actor selecciona el producto a eliminar 2. El actor procede a eliminarlo 3. El sistema elimina el registro de la persistencia	CONFORME
Anular Venta	01/11/2004	El actor decide anular una venta	CONFORME

		1. El actor selecciona el botón de anular venta 2. El sistema actualiza la lista de productos vendidos 3. Se actualiza el sistema	
Ingresar a Ventas	01/11/2004	El actor decide ingresar al formulario de ventas 1. El actor selecciona el menú Ventas de mercancía 2. El sistema llama al formulario frmOrden 3. El formulario inicializa el control de ordenes 4. El control inicializa la clase asociada de ordenes 5. La clase inicia a su clase de Embarcadores y empleados 6. Se carga el formulario de órdenes 7. Se carga el control de ordenes 8. Se asigna la conexión a la clase de ordenes 9. Se asigna el empleado por defecto 10. Se asigna el embarcador por defecto	CONFORME
Buscar Venta	01/11/2004	El actor desea buscar una orden en especial 1. El actor selecciona el botón de buscar 2. El sistema arma el criterio de busqueda	CONFORME

		3. El sistema comienza la búsqueda de registros 4. El sistema despliega la pantalla de resultados o factura, dependiendo de si son uno, ninguno, o muchos los resultados devueltos	
Recuperar Factura	01/11/2004	El actor desea recuperar una de las facturas desplegadas en el listview de búsqueda 1. El actor da doble click en la factura 2. El sistema busca la factura seleccionada por el usuario 3. El sistema despliega la factura y sus productos	CONFORME
llamar a Clientes	01/11/2004	El actor desea agregar o modificar un cliente 1. El actor selecciona el textbox de clientes 2. El actor oprime una tecla 3. El sistema despliega la pantalla de clientes 4. El actor busca o modifica la información del cliente 5. El actor sale de la pantalla 6. El sistema despliega la información del cliente seleccionado	CONFORME
Buscar Empleado	01/11/2004	El actor desea asignar o modificar el empleado que vendío el/los producto(s)	CONFORME

		1. El actor selecciona el campo de empleados 2. El actor oprime una tecla 3. El sistema despliega el formulario de empleados, para que el actor seleccione el empleado deseado 4. El actor termina 5. El sistema asigna la información del empleado al formulario de venta	
Buscar Producto	01/11/2004	El actor desea buscar un producto 1. El actor ingresa la información del producto 2. El actor procede a buscar el producto 3. El sistema despliega los productos que cumplan con el criterio de búsqueda	CONFORME
Agregar Producto a venta	01/11/2004	El actor desea agregar un producto a la lista de productos vendidos 1. El actor selecciona el producto 2. El actor envía el producto al área de productos vendidos 3. El sistema quita el producto del área de disponibles y los lleva al área de vendidos	CONFORME
Salvar Venta	01/11/2004	El actor ha concluido la venta, y desea salvar la información 1. El actor selecciona el botón de	CONFORME

		guardar 2. El sistema almacena la información de la factura y del detalle de la misma	

CRONOGRAMA DE ACTIVIDADES.

El proyecto se ha dividido en 4 fases: Inicio, Elaboración, Construcción y Transición. Dentro de cada una de estas, se discriminan los artefactos a desarrollar, y el espacio de tiempo requerido para su desarrollo

Actividad	AGOSTO				SEPTIEMBRE				OCTUBRE			
	1 Semana	2 Semana	3 Semana	4 Semana	1 Semana	2 Semana	3 Semana	4 Semana	1 Semana	2 Semana	3 Semana	4 Semana
I. FASE DE INICIO												
- Análisis de Requisitos	**X**											
II. FASE DE ELABORACIÓN												
- Construcción de la base de datos		**X**										
- Construcción de GeneralDLL		**X**										
III. FASE DE CONSTRUCCIÓN												
1. Subsistema Ingresar y Asegurar												
- Ingreso al sistema			**X**									
- Establecer Seguridad			**X**									
- Establecer valores por defecto			**X**									
2. Subsistema Categorías												
- Agregar Categorías			**X**									
- Eliminar Categorías			**X**									

- Buscar Categorías			**X**									
- Salvar Información			**X**									
- Llamar las Categorías desde otro formulario			**X**									
3. Subsistema Empleados												
- Agregar Empleados			**X**									
- Eliminar Empleados			**X**									
- Buscar Empleados			**X**									
- Salvar Empleados			**X**									
- Llamar Empleados desde otro formulario			**X**									
4. Subsistema Producto												
- Ingresar a productos				**X**								
- Llamar a categorías				**X**								
- Salvar Productos				**X**								
- Agregar Productos				**X**								
- Eliminar Productos				**X**								
5. Subsistema Proveedores												
- Ingresar a				**X**								

Proveedores												
- Nuevo				**X**								
- Eliminar				**X**								
- Salvar				**X**								
6. Subsistema Transportadores												
- Nuevo Transportador				**X**								
- Eliminar Transportador				**X**								
- Ingresar a proveedores desde el menú				**X**								
- Salvar Transportador				**X**								
- Buscar Transportador				**X**								
7. Compras de Mercancía												
- Nueva Mercancía					**X**	**X**						
- Eliminar Mercancía					**X**	**X**						
- Buscar Mercancía					**X**	**X**						
- Ingresar					**X**	**X**						
- Pasar a registrar factura					**X**	**X**						
- Buscar Factura de compra					**X**	**X**						
- Nueva Compra de Mercancía					**X**	**X**						
- Llamar a proveedores					**X**	**X**						
- Salvar a Factura					**X**	**X**						
- Eliminar					**X**	**X**						

Factura												
- Desplegar Productos					**X**	**X**						
- Salvar Mercancía					**X**	**X**						
8. Ventas de Productos												
- Nueva Venta							**X**	**X**				
- Modificar Venta							**X**	**X**				
- Eliminar Venta							**X**	**X**				
- Anular Venta							**X**	**X**				
- Trabajar con ventas de productos							**X**	**X**				
- Ingresar a ventas							**X**	**X**				
- Buscar Venta							**X**	**X**				
- Recuperar Factura							**X**	**X**				
- Llamar a clientes							**X**	**X**				
- Buscar Empleado							**X**	**X**				
- Buscar Producto							**X**	**X**				
- Agregar Producto a Venta							**X**	**X**				
- Salvar Venta							**X**	**X**				
9. Ayuda									**X**			
IV. FASE DE TRANSICIÓN												
1. Pruebas										**X**		
2. Documentación											**X**	**X**

Tabla 86. Cronograma de Actividades

CONCLUSIONES

El presente manual describe el proceso de desarrollo de un software que permite la gestión de las compras y las ventas, la generación de la factura y reportes para una empresa que comercializa productos a laboratorios clínicos ubicados en Bogotá, de acuerdo al Proceso Unificado de Desarrollo.

Descrito de acuerdo a la notación estándar para el modelado, el sistema ha sido presentado como un proceso centrado en la arquitectura, dirigido por el riesgo e iterativo.

En este enfoque, el proyecto ha sido organizado en una serie de mini – proyectos, de duración fija llamados iteraciones; el resultado de cada uno es un sistema que puede ser probado, integrado y ejecutado. Cada iteración incluye sus propias actividades de análisis de requisitos, diseño, implementación y pruebas.

BIBLIOGRAFIA

- **Korth, H. F. Silberschatz A**. Análisis y Diseño de Sistemas. Mc Graw Hill. Segunda Edición.
- **Pressman R. S**. Ingeniería del Software. Mc Graw Hill. Cuarta Edición.
- **Gail, L. Christie, J.** Enciclopedia de Términos de Computación. PHH, Prentice may