UNIVERSIDAD DEL AZUAY

FACULTAD DE CIENCIAS DE LA ADMINISTRACION

ESCUELA DE INGENIERIA DE SISTEMAS

"APLICACIÓN DISTRIBUIDA E-COMMERCE B2C (NEGOCIO - CONSUMIDOR) PARA LA EMPRESA PLANTACIONES EL TRÉBOL"

Tesis previo a la obtención del
Título de INGENIERO DE SISTEMAS

AUTORES:

Manuel Eduardo Mendoza Peñafiel.

Juan Sebastián Ochoa Ordóñez.

DIRECTOR:

Ing. Lenín Erazo Garzón.

Cuenca, Ecuador.
2006.

Agradecimiento.

Queremos dejar constancia de nuestro más sincero agradecimiento a todas las personas que de una u otra manera colaboraron en el desarrollo de la presente Tesis, ya que sin su valioso aporte no hubiese sido posible culminar con éxito la misma.

En especial a nuestro director de tesis, Ing. Lenin Erazo Garzón por su desinteresada colaboración.

A la Universidad del Azuay en donde forjamos nuestros conocimientos durante nuestra formación Universitaria.

A Plantaciones El Trébol por abrirnos sus puertas y permitirnos realizar la investigación que sirvió de base para la elaboración de este proyecto.

Los Autores.

Dedicatoria.

Dedico este trabajo a mis Padres que han sido mi mayor apoyo en el transcurso de mi vida estudiantil ya que sus sabios consejos me incentivaron luchar para lograr uno de mis mayores sueños, a mis hermanos, sobrinos, cuñadas y cuñado por estar a mi lado en los momentos buenos y malos de mi vida brindándome siempre su apoyo, a mis amigos que sin su apoyo no hubiese podido cumplir la meta que me trace, a mis maestros por haberme brindado sus conocimientos y principalmente a Dios por ser mi fuerza para poder culminar con éxito una más de mis metas.

Juan Sebastián.

Dedicatoria.

A Dios, por concederme el privilegio de la vida y la capacidad para lograr mis metas.

A mis padres por el impulso y el apoyo constante, por el consejo sabio y oportuno en todo momento.

A mis hermanos por la motivación continúa para alcanzar mis objetivos.

A mis maestros, por sembrar en mí sus conocimientos y fertilizar mi inquietud de conocer.

Manuel Eduardo.

Autoría

Todas las ideas expresadas en el presente trabajo son de exclusiva responsabilidad de sus autores.

Manuel Eduardo Mendoza Peñafiel

Juan Sebastián Ochoa Ordóñez

Índice de Contenidos.

Resumen

La presente tesis engloba el proceso de construcción de una aplicación de comercio electrónico, tomando como objeto de estudio la empresa Plantaciones El Trébol.

El objetivo es construir una solución basada en la arquitectura multi-capa, apoyándose en:

- La Notación UML
- La Metodología Orientada a Objetos
- La plataforma de desarrollo Microsoft Visual Studio .Net 2003
- El gestor de Bases de Datos SQL Server 2000
- La utilización de servicios Web XML.

Al final se tendrá como resultado una aplicación que aproveche al máximo el conjunto de recursos utilizados, y que permitirá la realización de transacciones comerciales mediante Internet. Así mismo prestará facilidades de programación remota para los clientes del sitio.

Abstract

This thesis includes the process of construction of an electronic commerce application, taking as object study the company Plantations The Clover.

The objective is to build a solution based on the architecture multi-layer, leaning on in:

- The UML Notation.
- The Oriented Objects Methodology
- The development platform Microsoft Visual Studio. Net 2003
- The Database Management System SQL Server 2000
- The use of Web services XML.

At the end, we will have an application that takes maximum advantage of the group of used resources; wich will allow the realization of commercial transactions by the Internet. Likewise, it will lend facilities of remote programming for the clients of the site.

Introducción

En la actualidad, la forma en la que las empresas realizan sus negocios, está cambiando de manera acelerada. El gran crecimiento de las tecnologías de la información y la disminución de los costos en los servicios de comunicaciones, permite que cada día más empresas ingresen a formar parte de la Internet, incrementando así su capacidad de comercialización.

El comercio electrónico, o E-Commerce, es el proceso que permite establecer acuerdos y llevar a cabo transacciones online entre clientes y socios de negocios. El reto más importante al que se tienen que enfrentar las empresas en la actualidad, es la reducción de sus tiempos de procesamiento y la optimización de los recursos, trabajando sobre una base de información consolidada. Es así que Plantaciones El Trébol, una empresa floricultora, que exporta sus productos a más de 10 países a nivel Mundial, se ha visto en la necesidad de tener más que una mera imagen en Internet, y aprovechar este recurso para brindar un mejor servicio a sus clientes mediante el E-Commerce.

Durante la presente tesis, se ilustrará el proceso de desarrollo de una aplicación de comercio electrónico, tomando como objeto de estudio la empresa Plantaciones el Trébol, se demostrará las ventajas de la utilización de la metodología Orientada a Objetos, la notación UML y principalmente el desarrollo de aplicaciones multi-capa y los servicios Web.

Capítulo 1

Conceptos Básicos

El propósito de este capítulo es establecer el marco teórico sobre el cual se desarrollará la presente tesis, con la presentación de los conceptos básicos del Comercio Electrónico, una breve introducción al Lenguaje de Modelado Unificado UML, demostrar la importancia del Análisis y de la Programación Orientada a Objetos; y, los beneficios de un desarrollo de aplicaciones empleando la arquitectura de n-capas mediante la utilización de la plataforma de desarrollo .NET.

Finalmente se establecerá de una manera muy general las necesidades de la empresa Plantaciones El Trébol, en donde será aplicada la investigación.

1.1 Conceptos de Comercio Electrónico

El Comercio Electrónico según la Organización Mundial de Comercio (OMC), se lo entiende como la producción, distribución, comercialización, venta o entrega de bienes y servicios por medios electrónicos[1].

En general se ha hecho una percepción del comercio electrónico como un asunto eminentemente técnico relacionado con Internet, computadoras y telecomunicaciones, antes que como una actividad comercial o de negocios, cuando en realidad, su parte esencial es la relacionada precisamente con el comercio, la tecnología y las telecomunicaciones pueden apoyar para llegar a más mercados, reducir costos de producción, tornarse más competitivo y eficiente, es decir, para comerciar con mayor calidad y en mayores volúmenes.

Su evolución se ha manifestado de una forma interesante, especialmente en los modelos de negocios, a diario aparecen nuevas formas de captar la atención del consumidor, los avances tecnológicos permiten poner a disposición de un número cada vez mayor de personas las vitrinas de ofertas disponibles, ya sea en sus propias computadoras, en las escuelas, colegios y universidades, en cabinas y terminales públicas o en los cada vez más populares cibercafés, la limitante de no tener acceso a Internet deberá ser superada a fuerza de que los empresarios requieran acceso al mercado y estén dispuestos a invertir en masificar las alternativas para que el público conozca sus productos.

Las cifras del comercio electrónico son dispares y variables de acuerdo a las fuentes de consulta, mercado, época en la que se realiza el estudio, etc. pero coinciden en mostrar una tendencia siempre en alza a través del tiempo, existen algunas razones para este continuo crecimiento entre las que se pueden citar están las siguientes:

- Incremento del número de gente joven familiarizada con la tecnología.
- El afianzamiento en la seguridad de las transacciones.

[1] http://www.wto.org/spanish/tratop_s/ecom_s/e274.htm

- Adopción de normas legales en la mayoría de países con respecto a los negocios por Internet.
- Grandes campañas publicitarias.
- Gobiernos que apoyan a los modelos de negocios en línea.
- Bajo costo de publicidad en línea.

1.1.1 Condiciones para desarrollar el comercio electrónico

La evolución del comercio electrónico a nivel mundial está necesariamente ligada al desarrollo de los siguientes factores:

- Infraestructura tecnológica.
- Seguridad informática.
- Régimen legal.
- Desarrollo de hábitos por parte del consumidor.

A favor de la evolución positiva del comercio electrónico a nivel mundial están el abaratamiento de la tecnología y de los medios de acceso, las facilidades que brindan diversos medios electrónicos de pago, el desarrollo sostenido de medios de entrega eficientes, la disminución de barreras Internacionales, la capacidad de consumo de ciertos mercados.

1.1.2. Modelos de Comercio Electrónico B2C (Negocio - Consumidor)

- Vendedores On-Line.
  - Exterminadores de Márgenes.
    - www.buy.com
    - www.alcoste.com
  - Agregadores de Productos (Product Aggregators)
    - www.amazon.com
    - www.elcorteingles.es
  - Especialistas (Product Specialist)
    - www.etoys.com
- Portales Transaccionales
  - Portales Horizontales
    - www.terra.es

        - www.yahoo.com
    - Portales Verticales
        - www.invertia.es
        - www.expedia.com
- Metamediarios
    - Transaction Handlers
        - www.theknot.com (Bodas)
    - Transaction Facilitators
        - www.decide.com
- Precios Dinámicos
    - Subastas Directas
        - www.mercadolibre.com
    - Subastas Inversas
        - www.mercata.com
    - Compra Sindicada (Agregadores de Demanda)
        - www.lestbuyit.com
    - Agentes Inteligentes
        - A futuro

1.2 Conceptos de Modelado Unificado UML

El lenguaje de modelado unificado UML es una notación compuesta por un conjunto de diagramas y elementos que pueden ser organizados para describir el diseño de un sistema de software: UML no es un proceso mucho menos un método.

En teoría, se puede aplicar aspectos de esta notación de acuerdo a la metodología escogida (Cascada, RAD, etc), pero existen procesos específicos desarrollados para complementar la notación UML.

1.2.1 Por qué usamos UML

En el contexto del desarrollo de software, la necesidad de un diseño formal se vuelve cada vez más importante dependiendo del tamaño, la complejidad del proyecto y sobre todo de la cantidad de personas involucradas. Básicamente nos sirve para:

- Construir el diseño del proyecto

- Estimar y planificar el tiempo y los materiales
- Comunicar las ideas dentro del equipo de trabajo y a otras personas
- Documentar el proyecto
- Garantizar la calidad del software

1.2.2 Notación UML

La notación UML está compuesta por un conjunto de diagramas como son:

- Diagramas de Actividades
- Diagramas de Casos de Uso
- Diagramas de secuencias y colaboración
- Diagramas de estados
- Diagramas de estructura estática
- Diagramas de componentes
- Diagramas de implementación

Cada diagrama representa una vista diferente de la misma aplicación, y se clasifican en diagramas dinámicos y estáticos.

Diagramas Dinámicos.- Muestran el comportamiento del sistema a través del tiempo y son:

Diagrama de Actividades.- Es muy parecido a un diagrama de flujo, contiene los siguientes elementos:

- Estado Inicial.- En donde empieza el diagrama
- Control de Flujo.- Muestra la transferencia de control de una actividad a otra
- Estado.- Representa un período de tiempo durante el cual un fragmento de trabajo es ejecutado por una persona o un equipo
- Transición de Bifurcación.- Muestra un punto en el cual empiezan dos o más actividades paralelas
- Transición de Unión.- Muestra un punto en el cual dos o más actividades paralelas deben sincronizarse y converger.
- Carril de Flujo.- Permite agrupar todas las actividades efectuadas por una persona o grupo en una sola columna.
- Acción de Entrada.- Indica qué debe pasar cuando empieza la actividad

- Objeto de Estado.- Muestra un objeto que es producido o consumido en el transcurso de una actividad
- Estado Final.- En donde termina el diagrama

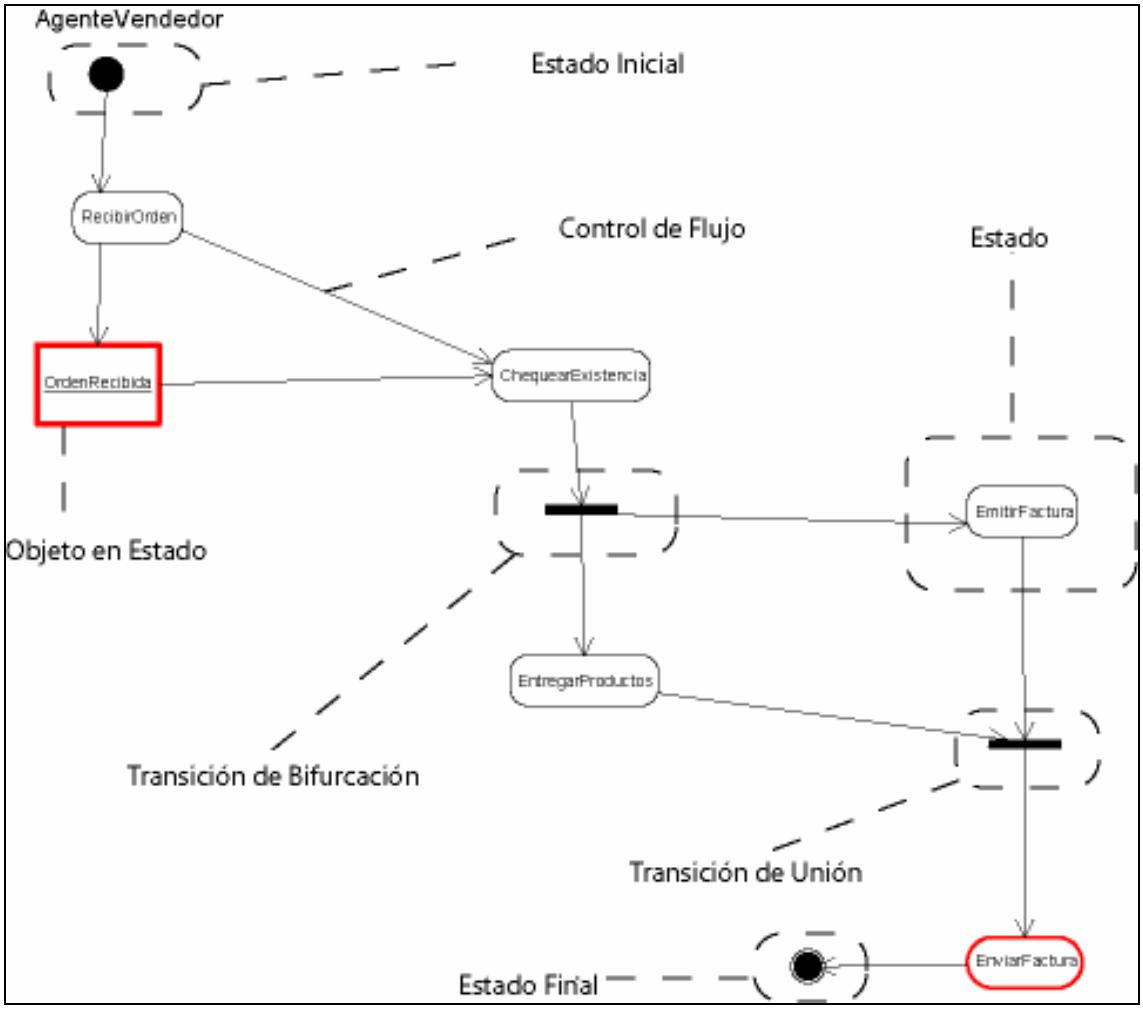


Figura 1.1. Diagrama de Actividades

Diagrama de Caso de Uso

Contiene los siguientes elementos:

Actor.- Muestra una persona o un sistema externo que inicia el caso de uso con la finalidad de obtener algún valor de la aplicación.

Caso de Uso.- Es una unidad de funcionalidad bien definida que el sistema proporciona a uno o más actores.

Notificación.- Indica que un actor específico utiliza un caso de uso en particular.

Relación <<usa>>.- Indica que una funcionalidad potencialmente reutilizable ha sido construido en un caso de uso separado.

Relación <<expandir>>.- Indica alguna funcionalidad adicional que puede ser proporcionada como soporte del caso de uso, dicha funcionalidad ha sido construida en un caso de uso separado.

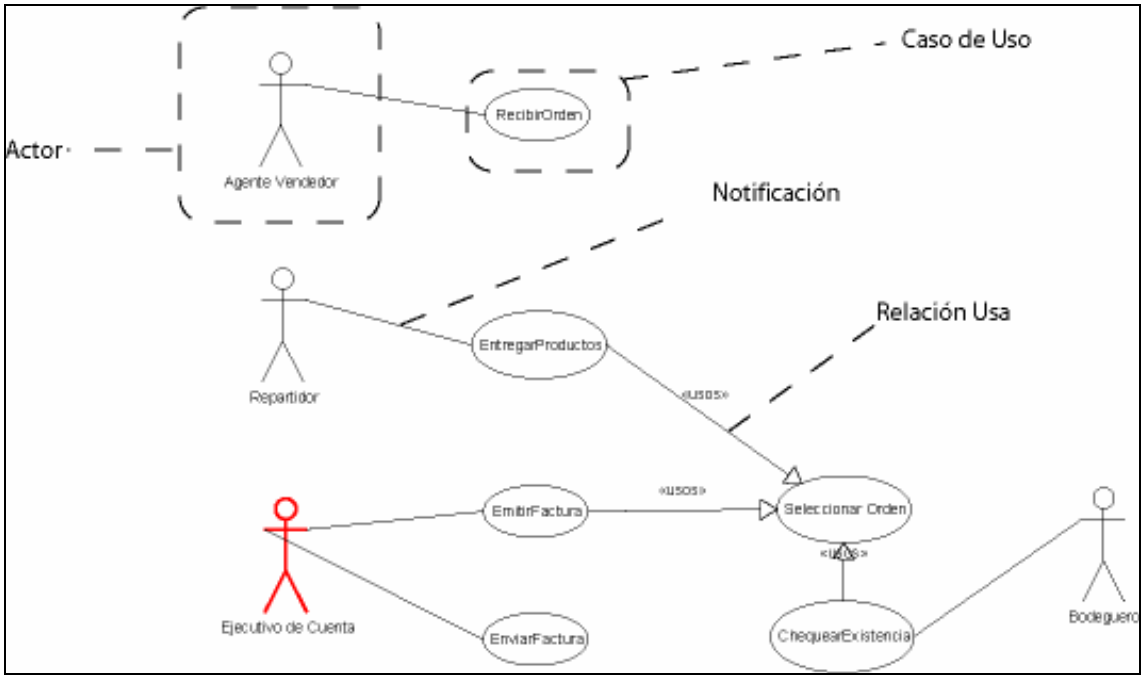


Figura 1.2. Caso de Uso

Diagramas de Interacción

Los casos de uso se realizan mediante diagramas de interacción, existen 2 tipos:

- Diagramas de secuencia
- Diagramas de Colaboración

Diagramas de Secuencia

Contiene los siguientes elementos:

- **Objeto**.- Representa una instancia de un objeto que envía o recibe mensajes hacia y desde otras instancias de objetos.
- **Mensaje**.- Muestra una interacción entre dos objetos

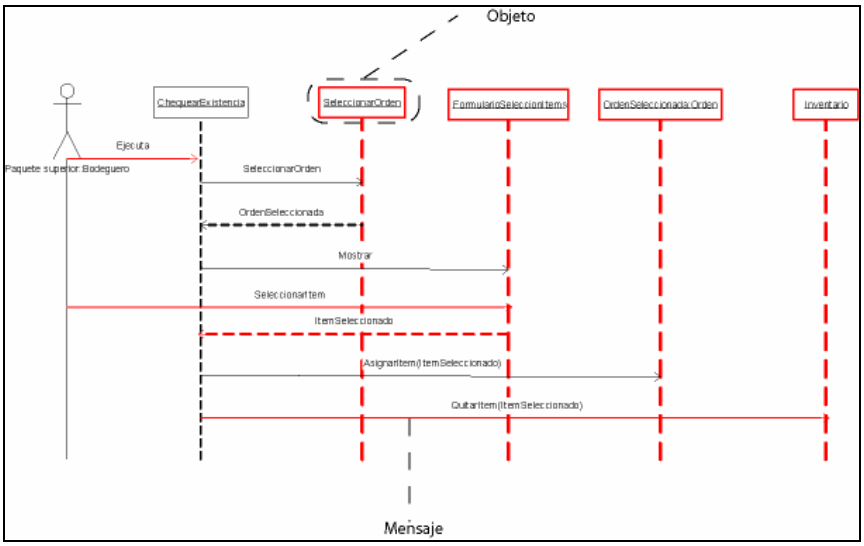


Figura 1.3. Diagrama de Secuencia

Diagrama de Colaboración

Es otra forma de representar la interacción del sistema.

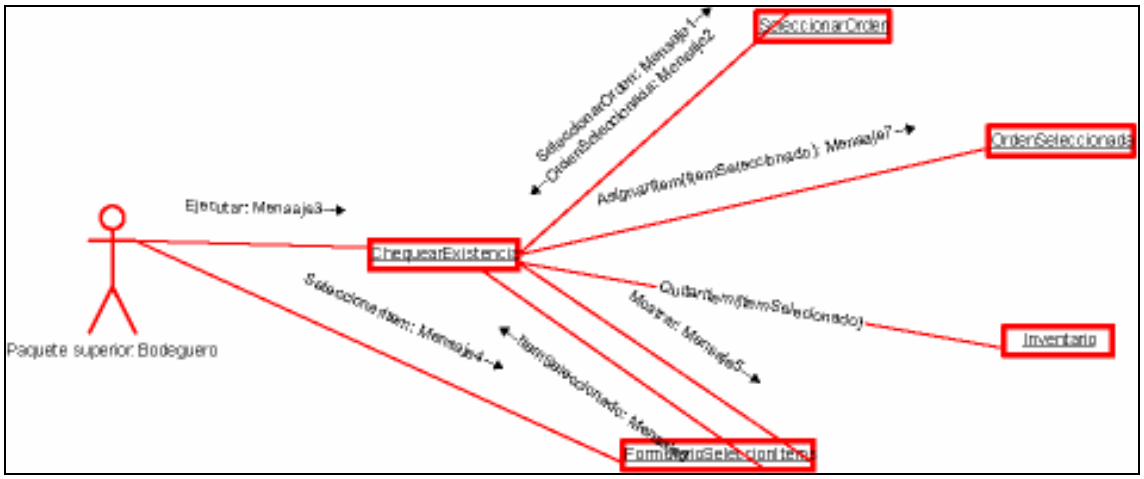


Figura 1.4. Diagrama de Colaboración

Diagramas de Estados

Sirven para mostrar de forma detallada, el conjunto completo de estados por los que atraviesa un objeto.

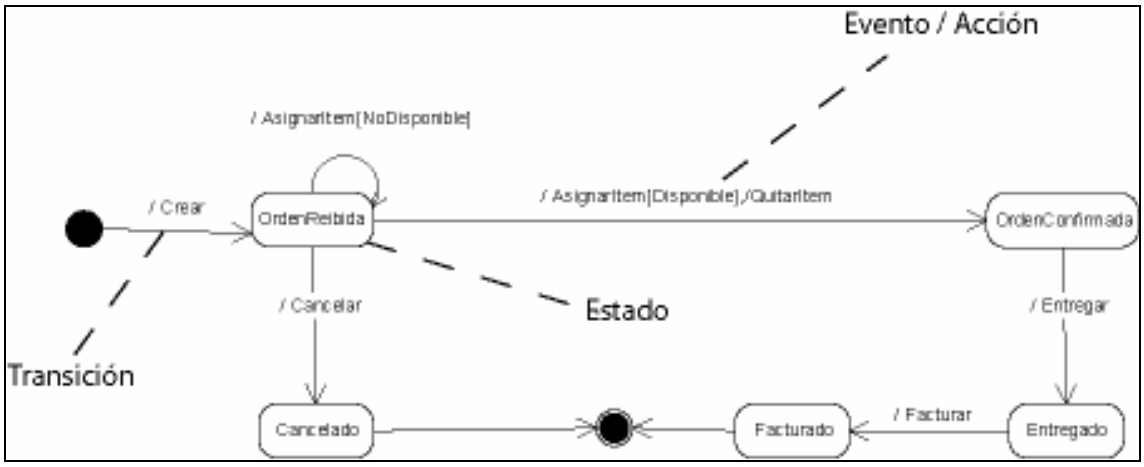


Figura 1.5. Diagrama de Estados

Diagramas Estáticos.- Muestran las relaciones y dependencias persistentes entre las clases y sus componentes, y son:

Diagrama de estructura estática

Normalmente conocido como diagrama de clases, está compuesto por los siguientes elementos:

- **Clase**.- Se representa por un rectángulo dividido en tres segmentos que contienen el nombre de la clase, los atributos miembro y las operaciones miembro.
- **Generalización.-** Es una relación de herencia entre una superclase y una o varias subclases.
- **Dependencia**.- Muestra la dependencia no persistente de una clase sobre la funcionalidad de otra
- **Composición**.- Muestra la dependencia persistente de una clase sobre otra, representa una relación propietario-.
- **Asociación.-** Es una forma más general de vincular dos clases, que no representa una relación usuario-uso.

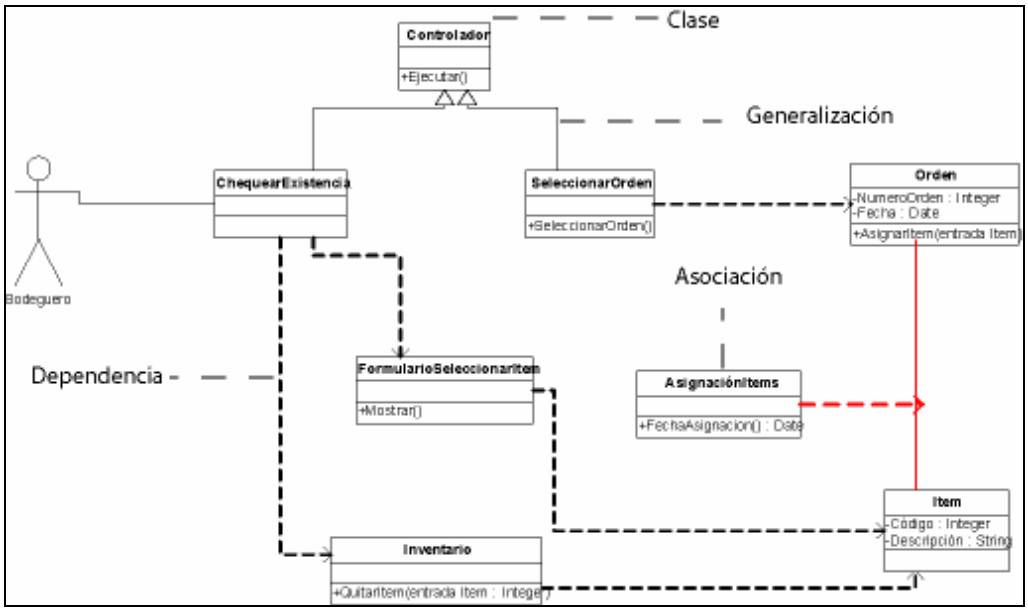


Figura 1.6. Diagrama de estructura estática

Diagrama de Componentes

Muestra cómo está organizada una aplicación en paquetes y librerías para su implementación.

- Ejecutable.- Representa un programa
- Librería.- Representa una colección de clases a las que se puede referenciar en un proyecto

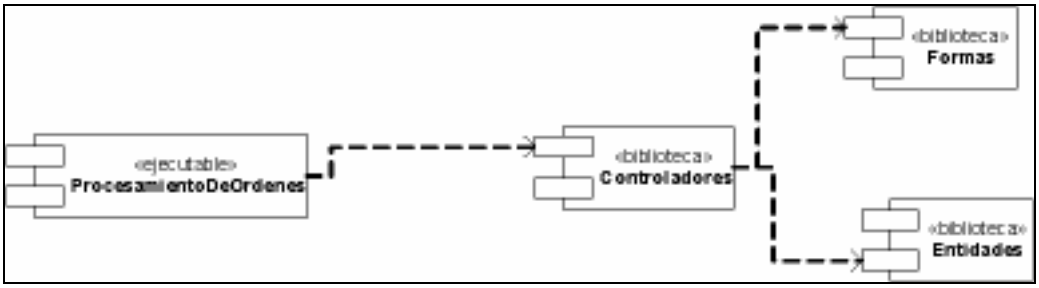


Figura 1.7. Diagrama de Componentes.

Diagrama de Implementación

Es el último diagrama UML, muestra los nodos físicos en los que los componentes de software serán instalados.

- **Nodo**.- Representa un recurso computacional en el cual se puede implementar el software

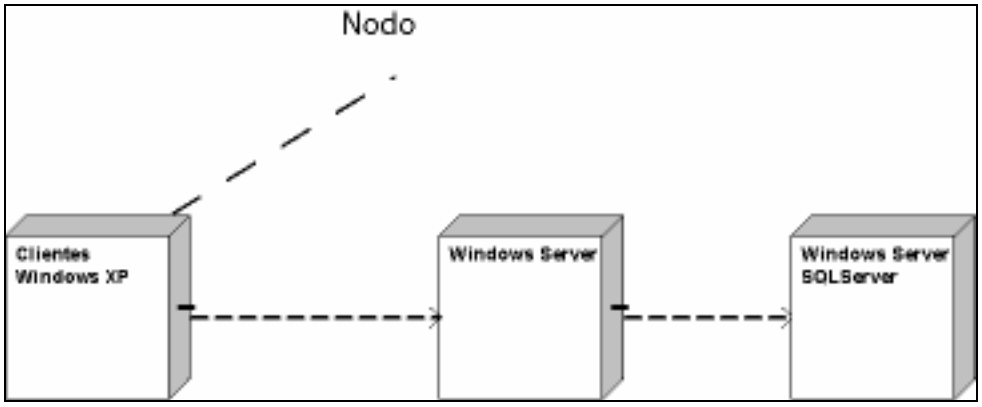


Figura 1.8. Diagrama de Implementación

1.3. Conceptos del Análisis Orientado a Objetos

El objetivo del análisis orientado a objetos es desarrollar una serie de modelos que describan el software de computadora al trabajar para satisfacer un conjunto de requisitos definidos por el cliente. El Análisis Orientado a Objetos, así como los métodos de análisis convencional forman un modelo de análisis multiparte para satisfacer este objetivo. El modelo de análisis ilustra información, funcionamiento y comportamiento dentro del contexto de los elementos del modelo de objetos.

1.3.1 El panorama del Análisis Orientado a Objetos

Las tecnologías de objetos ha generado docenas de métodos de Análisis Orientado a Objetos desde finales de los 80 y durante los 90, cada uno de ellos introduce un proceso para el análisis de un producto o sistema, un conjunto de modelos que evoluciona fuera del proceso y una notación que posibilita crear cada modelo de un amanera consistente, entre los más utilizados se encuentran:

- **El método de Booch:** Abarca un microproceso de desarrollo y un macroproceso de desarrollo, el nivel micro define un conjunto de tareas de análisis que se reaplican en cada etapa en el macroproceso. El microproceso de desarrollo identifica clases y objetos y la semántica de dichas clases y objetos, define las relaciones entre clases y objetos y realiza una serie de refinamientos para elaborar el modelo del análisis.

- **El método de Rumbaugh:** Desarrollo la Técnica de Modelado de Objetos (OMT) para el análisis, diseño del sistema y diseño a nivel de objetos, la actividad de análisis crea tres modelos: el modelo de objetos (clases, objetos, jerarquías y relaciones), el modelo dinámico (comportamiento del sistema y los objetos) y el modelo funcional (flujo de información del sistema).
- **El método de Coad y Yourdon:** Se considera con frecuencia como uno de los métodos del Análisis Orientado a Objetos más sencillos de aprender, la notación es relativamente simple y las reglas para desarrollar el modelo de análisis son evidentes, a continuación se mencionan los pasos del proceso de Análisis Orientado a objetos de Coad y Yourdon:
  - Identificar objetos usando el criterio de qué buscar.
  - Definir una estructura de generalización-especificación.
  - Definir una estructura de todo-parte.
  - Identificar temas.
  - Definir atributos.
  - Definir servicios.
- **El método de Wirfs-Brock:** No hace una distinción clara entre las tareas de análisis y diseño, sino que propone un proceso continuo que comienza con la valoración de una especificación del cliente y termina con el diseño.

Aunque la terminología y etapas del proceso para cada uno de estos métodos de Análisis Orientado a Objetos difieren, los procesos generales de Análisis Orientado a Objetos son en realidad muy similares, para realizar un análisis orientado a objetos se deberían ejecutar las siguientes etapas genéricas:

1. Obtener los requisitos del cliente para el sistema.
2. Identificar escenarios o casos de uso.
3. Seleccionar clases y objetos usando los requisitos básicos.
4. Identificar atributos y operaciones para cada objeto del sistema.
5. Definir estructuras y jerarquías que organicen las clases.
6. Construir un modelo Objeto-Relación.
7. Construir un modelo Objeto-Comportamiento.

8. Revisar el modelo de análisis orientado a objetos con relación a los casos de uso/escenarios.

1.3.2. Análisis del dominio

El análisis en sistemas orientados a objetos puede ocurrir a muchos niveles diferentes de abstracción. A nivel de negocios o empresas se hace un esfuerzo por definir clases, objetos, relaciones y comportamientos que modelen el negocio por completo. A nivel de áreas de negocios, puede definirse un modelo de objetos que describa el trabajo de un área específica de negocios, a nivel de las aplicaciones el modelo de objetos se centra en los requisitos específicos del cliente. El análisis del dominio tiene lugar cuando se quiere crear una biblioteca de clases reutilizables aplicables a una categoría completa de aplicaciones.

1.3.2.1. El proceso de análisis del dominio: El análisis del dominio es la actividad de cobertura para el proceso del software, es una actividad no ligada a ningún proyecto de software. El papel del analista del dominio es diseñar y construir componentes reutilizables que puedan ser reutilizados por diferentes personas que trabajan en aplicaciones similares pero no necesariamente iguales, en esencia el análisis del dominio es muy similar a la ingeniería del conocimiento, durante el análisis del dominio ocurre la extracción de objetos.

1.3.2.1.1. Definir el dominio a investigar: Primero se debe aislar el área de negocio, tipo de sistema o categoría del producto de interés, luego se extraen los elementos tanto orientados a objetos como no orientados a objetos, los elementos orientados a objetos incluyen especificaciones, diseños y código para clases de aplicaciones orientados a objetos ya existentes, clases de soporte, bibliotecas de componentes y casos de prueba.
Los elementos no orientados a objetos abarcan políticas, procedimientos, planes, estándares y guías, partes de aplicaciones no orientadas a objetos y métricas.

1.3.2.1.2. Clasificar los elementos extraídos del dominio: Los elementos se organizan en categorías y se establecen las características generales que definen la categoría, se propone un esquema de clasificación para las categorías y se definen

convenciones para la nomenclatura de cada elemento, se establecen las jerarquías de clasificación en caso de ser apropiado.

1.3.2.1.3. Recolectar una muestra representativa de aplicaciones en el dominio: La aplicación en cuestión tiene elementos que cae dentro de las categorías ya definidas, el analista del dominio debe identificar los objetos conceptuales en cada aplicación.

1.3.2.1.4. Analizar cada aplicación dentro de la muestra: Se deben seguir las siguientes etapas:

- Identificar objetos candidatos reutilizables.
- Indicar las razones que hacen que el objeto haya sido identificado como reutilizable.
- Definir adaptaciones al objeto que también pueden ser reutilizables.
- Estimar el porcentaje de aplicaciones en el dominio que pueden reutilizar el objeto.
- Identificar los objetos por nombre y usar técnicas de gestión de configuración para controlarlos.

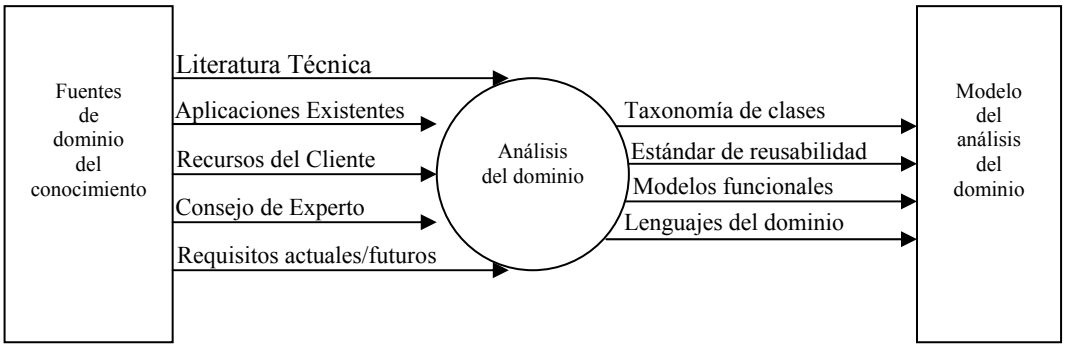


Figura 1.8. Entrada y Salida para el análisis del dominio

1.3.3. Componentes genéricos del modelo de análisis orientado a objetos.

El proceso de análisis orientado a objetos se adapta a conceptos y principios básicos de análisis, aunque la terminología, notación y actividades difieren respecto de los

usados en métodos convencionales, el análisis orientado a objetos resuelve los mismos objetivos subyacentes.

Para desarrollar un modelo preciso, conciso, comprensible y correcto del mundo real se debe seleccionar una notación que se soporte a un conjunto de componentes genéricos de análisis orientado a objetos, a continuación se mencionan los componentes genéricos del modelo de análisis orientado a objetos:

- **Vista estática de clases semánticas:** Se imponen los requisitos y se extraen las clases como parte del modelo de análisis, las clases persisten a través de todo el período de vida de la aplicación.

- **Vista estática de los atributos:** Toda clase debe describirse explícitamente, los atributos asociados con la clase aportan una descripción de la clase, así como una indicación inicial de las operaciones de la clase.

- **Vista estática de las relaciones:** Los objetos están conectados unos a otros de varias formas, se debe representar las relaciones de manera tal que puedan identificarse las operaciones y que pueda desarrollarse un buen diseño de intercambio de mensajes.

- **Vista estática de los comportamientos:** Las relaciones definen un conjunto de comportamientos que se adaptan al escenario utilizado del sistema.

- **Vista dinámica de la comunicación:** Los objetos deben comunicarse unos con otros y hacerlo basándose en una serie de mensajes que provoquen transiciones de un estado a otro del sistema.

- **Vista dinámica del control y manejo del tiempo:** Debe describirse la naturaleza y duración de los sucesos que provocan transiciones de estados.

1.3.4. El Proceso del Análisis Orientado a Objetos.

1.3.4.1. Casos de Uso.

Los casos de uso modelan el sistema desde el punto de vista del usuario, creados durante la obtención de los requisitos, deben cumplir con los siguientes objetivos:

- Definir los requisitos funcionales y operativos del sistema.
- Proporcionar una base para la validación de las pruebas.

Durante el análisis orientado a objetos los casos de uso sirven como base para los primeros elementos del modelo de análisis, los casos de uso pueden representarse a diferentes niveles de abstracción. Los diagramas de casos de uso contienen casos de uso y actores, siendo los actores las entidades que interactúan con el sistema.

1.3.4.2. Modelado de Clases, Responsabilidades y Colaboraciones.

El modelado de clases, relaciones y colaboraciones (CRC) aporta un medio sencillo de identificar y organizar las clases que resulten relevantes al sistema, el modelo CRC puede hacer uso de tarjetas índices virtuales o reales, el caso es desarrollar una representación organizada de las clases. Las responsabilidades son los atributos y operaciones relevantes para la clase. Los colaboradores son aquellas clases necesarias para proveer a una clase con la información necesaria para completar una responsabilidad.

1.3.4.2.1. Clases.

Los objetos se manifiestan en una variedad de formas: entidades externas, cosas, ocurrencias o sucesos, roles, unidades organizativas, lugares o estructuras. Una técnica para identificarlos en el contexto de un problema del software es realizar un análisis gramatical con la narrativa de procesamiento para el sistema, todos los nombres se transforman en objetos potenciales, sin embargo no todo objeto potencial podrá incluirse en el modelo, un objeto potencial debe satisfacer seis características para poder ser considerado como posible miembro del modelo:

1. **Retener información:** El objeto potencial será útil durante el análisis si la información sobre el mismo debe guardarse para que el sistema funcione.
2. **Servicios necesarios:** El potencial objeto debe tener un conjunto de operaciones identificables que permitan cambiar los valores de los resultados.
3. **Múltiples atributos:** Durante el análisis de requisitos nos centramos más en la información más importante.
4. **Atributos comunes:** El conjunto de atributos definido para la clase debe ser aplicable a todas la ocurrencias.
5. **Operaciones comunes:** El objeto potencial debe definir un conjunto de operaciones aplicables a todos los objetos de la clase.
6. **Requisitos esenciales:** Las entidades externas que aparecen en el espacio del problema y producen información esencial para la operación de una solución para el sistema casi siempre se definen como objetos en el modelo de requisitos.

Firesmith extiende las características de clasificación sugiriendo además de las mencionadas las siguientes:

- **Clases dispositivo:** Modelan entidades externas.
- **Clases propiedad:** Representan alguna propiedad importante del entorno del problema.
- **Clases interacción:** Modelan interacciones que ocurren entre otros objetos.

Adicionalmente los objetos y las clases pueden clasificarse por un conjunto de características:

- **Tangibilidad:** Representa la clase algo tangible o palpable (teclado) o representa información más abstracta (una salida prevista).
- **Inclusividad:** Es la clase atómica (no incluye otras clases) o es agregada (incluye al menos un objeto anidado).
- **Secuencia:** Es la clase concurrente (posee su propio hilo de control) o secuencial (controlada por recursos externos).

- **Persistencia:** La clase es temporal (se crea durante la ejecución del programa y se elimina cuando este termina) o permanente (almacenada en una base de datos).
- **Integridad:** Es la clase corrompible (no protege sus recursos de influencias externas) o es segura (la clase refuerza los controles de accesos a sus recursos).

1.3.4.2.2. Responsabilidades.

Los atributos representan características estables de una clase, esto es información sobre la clase que debe retenerse para llevar a cabo los objetivos del software. Los atributos pueden a menudo extraerse del planteamiento de alance o discernirse a partir de la comprensión de la naturaleza de la clase. Las operaciones pueden extraerse desarrollando un análisis gramatical sobre la narrativa de procesamiento del sistema.
Los verbos se transforman en candidatos a operaciones, cada operación elegida para una clase exhibe un comportamiento de la clase.

Wirfs-Brock y sus colegas sugieren cinco pautas para especificar responsabilidades para las clases:

1. La inteligencia del sistema debe distribuirse de manera igualitaria, toda aplicación encierra un grado de inteligencia, esa inteligencia puede distribuirse entre las clases, las clases tontas pueden modelarse de manera que actúen como sirvientes de las clases listas, este enfoque hace que el flujo de control dentro de un sistema sea claro, posee algunas desventajas:
    a. Concentra toda la inteligencia en pocas clases.
    b. Tiende a necesitar más clases.

    Por esta razón la inteligencia del sistema debe distribuirse de manera igualitaria entre las clases de una aplicación.

    Para determinar si la inteligencia del sistema está distribuida equitativamente las responsabilidades definidas en cada tarjeta índice del modelo CRC deben ser evaluadas para determinar si cada clase posee una lista de responsabilidades extraordinariamente grande, las responsabilidades de cada clase deben mostrar el mismo nivel de abstracción.

2. Cada responsabilidad debe establecerse lo más general posible, la responsabilidades generales (atributos y operaciones) deben residir en la parte alta de la jerarquía de clases, adicionalmente debe usarse el polimorfismo para definir las operaciones que generalmente se aplica a la superclase.
3. La información y el comportamiento asociado a ella debe encontrarse dentro de la misma, los datos y procesos que manipulan estos datos deben empaquetarse como una unidad cohesionada.
4. La información sobre un elemento debe estar localizada dentro de una clase, no distribuida a través de varias clases, una clase simple debe asumir la responsabilidad de almacenamiento y manipulación de un tipo específico de información.
5. Compartir responsabilidades entre las clases relacionadas cuando sea apropiado, colaborar con los otros objetos.

1.3.4.2.3. Colaboradores.

Las colaboraciones identifican relaciones entre clases, las colaboraciones se identifican determinando si una clase puede satisfacer cada responsabilidad, si no puede, entonces necesita interactuar con otras clase, de aquí surge lo que se llama una colaboración. Para ayudar en la identificación de colaboradores se puede analizar tres relaciones genéricas diferentes entre clases:

1. La relación **es parte de.**
2. La relación **tiene conocimiento sobre.**
3. La relación **depende de.**

El nombre de la clase colaboradora se registra en la tarjeta índice del modelo CRC a lado de la responsabilidad que ha generado dicha colaboración.

1.3.4.3. Definición de estructuras y jerarquías.

Una vez que se han identificado las clases y objetos usando el modelo CRC, se comienza a centrar en la estructura del modelo de clases y las jerarquías resultantes

que surgen al emerger clases y subclases, usando la notación UML debe crearse una estructura de **generalización-especialización** para las clases identificadas.
En otros casos un objeto representado según el modelo inicial puede estar compuesto realmente de un número de partes las cuales pueden definirse a su vez como objetos. Estos objetos agregados pueden representarse como una estructura **componente-agregado.**
Las representaciones estructurales proveen los medios para particionar el modelo CRC y para representar esta partición gráficamente.

1.3.4.4. Definición de subsistemas.

Los subconjuntos de clases que colaboran entre sí para llevar a cabo un conjunto de responsabilidades cohesionadas, se les llama normalmente subsistemas o paquetes, estos son abstracciones que aportan una referencia o puntero a los detalles en el modelo de análisis, un subsistema puede tratarse como una caja negra que contiene un conjunto de responsabilidades y que posee sus propios colaboradores. Un subsistema implementa uno o más contratos con sus colaboradores, un contrato es una lista específica de solicitudes que los colaboradores pueden hacer a un subsistema.
Los subsistemas pueden representarse dentro del contexto del modelo CRC creando una tarjeta índice del subsistema, la tarjeta índice del subsistema indica el nombre del subsistema, los contratos que debe cumplir y las clases u otros subsistemas que soportan el contrato.
Los paquetes son idénticos a los subsistemas en intención y contenido, pero se representan gráficamente en UML

1.3.5. El Modelo Objeto Relación.

El primer paso en el establecimiento de las relaciones es comprender las responsabilidades de cada clase, la tarjeta índice del modelo CRC contiene una lista de responsabilidades, el siguiente paso es definir aquellas clases colaboradoras que ayudan en la realización de cada responsabilidad, esto establece la conexión entre clases.
Entre dos clases cualesquiera que estén conectadas existe una relación, debido a esto los colaboradores siempre están relacionados de alguna manera. El tipo de relación

más común es la binaria la misma que posee una dirección específica que se define a partir de qué clase desempeña el papel del cliente y cuál actúa como servidor.
La notación del lenguaje unificado de modelado para el modelo objeto relación utiliza una simbología adaptada de las técnicas del modelo entidad relación, los objetos se conectan con otros objetos utilizando relaciones con nombre, se especifica la cardinalidad de la conexión y se establece toda una red de relaciones, el modelo objeto relación puede obtenerse en tres pasos o etapas:

1. Usando las tarjetas índice CRC, puede dibujarse una red de objetos colaboradores.
2. Revisando el modelo CRC de tarjetas índice se evalúan las responsabilidades y colaboradores y cada línea de conexión sin etiquetar recibe un nombre, para evitar ambigüedad una punta de flecha indica la dirección de la relación.
3. Una vez que se han establecido y nombrado las relaciones se evalúa cada extremo para determinar la cardinalidad, la misma que puede ser:
    a. 0 a 1.
    b. 1 a 1.
    c. 0 a muchos.
    d. 1 a muchos.

Los pasos anteriores continúan hasta que se produzca un modelo objeto relación completo.

1.3.6. El Modelo Objeto Comportamiento.

El modelo objeto comportamiento indica cómo responderá un sistema orientado a objetos sucesos externos o estímulos, para crear el modelo se deben ejecutar los siguientes pasos:

1. Evaluar todos los casos de uso para comprender totalmente la secuencia de interacción dentro del sistema.
2. Identificar sucesos que dirigen la secuencia de interacción y comprender cómo estos sucesos se relacionan con objetos específicos.
3. Crear una traza de sucesos para cada caso de uso.

4. Construir un diagrama de transición de estados para el sistema.
5. Revisar el modelo objeto comportamiento para verificar la exactitud y consistencia.

1.3.6.1. Identificación de sucesos con casos de uso.

El caso de uso representa una secuencia de actividades que incluyen a actores y al sistema, en general un suceso ocurre cada vez que un sistema orientado a objetos y un actor intercambian información, los sucesos son booleanos.
Un suceso no es la información que se intercambia, es el hecho de que la información ha sido intercambiada, un caso de uso se examina por puntos de intercambio de información, deberá identificarse un actor para cada suceso, la información que se intercambia debe anotarse y deberán indicarse otras condiciones o restricciones.
Una vez que todos los sucesos han sido identificados se asocian a los objetos incluidos, los actores y los objetos pueden responsabilizarse de la generación de sucesos o reconociendo sucesos que han ocurrido en otra parte.

1.3.6.2. Representación de estados.

En el contexto de sistemas orientados a objetos deben considerarse dos caracterizaciones de estados:

1. El estado de cada objeto cuando el sistema ejecuta su función.
2. El estado del sistema observado desde el exterior cuando éste ejecuta su función.

El estado de un objeto adquiere en ambos casos características pasivas y activas, un estado pasivo es simplemente el estado actual de todos los atributos de un objeto.
El estado activo de un objeto indica el estado actual cuando éste entra en una transformación continua o proceso, para forzar la transición de un objeto de un estado activo a otro debe ocurrir un suceso (disparador), un componente de un modelo objeto comportamiento es una representación simple de los estados activos de cada objeto y los sucesos que producen los cambios entre estos estados activos. El modelo de estado activo aporta una visión interna muy útil de la historia de vida de

un objeto, es posible especificar información adicional para aportar más profundidad en la comprensión del comportamiento de un objeto, adicionalmente a especificar el suceso que provoca la ocurrencia de la transición el análisis puede especificar una guarda y una acción, una guarda es una condición booleana que debe satisfacerse para posibilitar la ocurrencia de una transición.
Una acción ocurre concurrentemente con la transición o como una consecuencia de ella y generalmente implica una o más operaciones del objeto.

El segundo tipo de representación de comportamiento para el análisis orientado a objetos considera una representación de estados para el sistema, esta representación abarca un modelo simple de traza de sucesos que indica cómo los sucesos causan las transiciones de objeto a objeto y un diagrama de transición de estados que ilustra el comportamiento de cada objeto durante el procesamiento, una vez que han sido identificados los sucesos para un caso de uso, se crea una representación acerca de cómo los sucesos provocan el flujo desde un objeto a otro, esta representación llamada traza de sucesos es una versión abreviada del caso de uso que representa objetos clave y los sucesos que provocan el comportamiento de pasar de objeto a objeto.

1.4. Conceptos de Programación Orientada a Objetos.

La simple mención de la palabra objetos puede provocar excesiva ansiedad en muchos programadores. No se preocupe, quizá sin saberlo, ha estado trabajando con objetos la mayor parte de su vida. Una vez que haya comprendido algunos conceptos básicos, los objetos le facilitarán la programación más que nunca.

Las clases y los objetos están estrechamente relacionados, pero no son lo mismo.
Una clase contiene información sobre cuál debe ser la apariencia y el comportamiento de un objeto. Una clase es el plano o esquema de un objeto. Por ejemplo, el esquema eléctrico y de diseño de un teléfono sería algo similar a una clase. El objeto o una instancia de la clase sería el teléfono.

1.4.1. La programación orientada a objetos.

La aparición en la programación del concepto de objetos, revolucionó la historia de esta, se introdujeron al lenguaje diario de los profesionales del área, conceptos tales como herencia, polimorfismo, encapsulamiento, etc., que hicieron que el modo de pensar y concebir el desarrollo de una aplicación fuera muy diferente a lo que había sido hasta ese momento con la programación lineal o estructurada.

En la programación orientada a objetos se tiene dos elementos fundamentales, las clases y los objetos, los mismos que están estrechamente relacionados pero no son lo mismo. El objeto o una instancia de la clase sería el teléfono.

A las clases se las puede definir como la generalización de los objetos, una clase contiene información sobre cuál debe ser la apariencia y el comportamiento de un objeto. Una clase es el plano o esquema de un objeto y a los objetos se los define como la concreción de la clase.

La metodología de la Programación Orientada a Objetos (POO) gira en torno a términos tales como abstracción, encapsulamiento, herencia y polimorfismo; para poder comprender lo que significa la Programación Orientada a Objetos debemos tener muy claros los términos mencionados.

Se ha mencionado las cuatro características básicas que debe cumplir un objeto para denominarse como tal, un objetos es el pilar fundamental de la programación orientada a objetos.

Cuando se habla de objetos, se habla de pequeños elementos bien definidos, representaciones verdaderas de objetos que tenemos en la vida real, mediante el uso e implementación de la metodología de “Programación Orientada a Objetos” se intenta crear objetos de software que tengan correlación con los objetos del mundo real en la solución que intentamos implementar, por ejemplo si estamos desarrollando un módulo de facturación de un sistema de gestión comercial, debemos crear el objeto factura el mismo que tendrá relaciones con los objetos clientes y artículos, la aplicación de esta concepción de trabajo es la que nos permiten aislar cada componente del resto de la aplicación y de esa forma aprovechar en mayor medida nuestro esfuerzo, nuestra concentración en el buen funcionamiento de dichos

componentes, nuestro control sobre ellos, más prolijidad en la codificación y por sobre todas las cosas, poder reutilizar el código que ya ha sido escrito.

Una vez que se ha definido el concepto de lo que es un objeto y sabiendo cuáles son sus cuatro características principales y necesarias para denominarse como tal, detallaremos cada una de esas cuatro características.

1.4.2. Abstracción.

La abstracción es la capacidad de un objeto para cumplir sus funciones independientemente del contexto en el que se lo utilice es decir el objeto "cliente" siempre expondrá sus mismas propiedades y dará los mismos resultados a través de sus usos sin importar el ámbito en el cual se lo haya creado.

1.4.3. Encapsulamiento.

El encapsulamiento es la característica que denota la capacidad del objeto de responder a peticiones a través de sus métodos sin la necesidad de exponer los medios utilizados para llegar a brindar estos resultados.

Por Ejemplo el método **MostrarSaldo()** del objeto "cliente" mencionado con anterioridad siempre va a dar el saldo de la cuenta de un cliente, sin necesidad de tener conocimiento de cuáles son los recursos que ejecuta para llegar a brindar este resultado.

1.4.4. Herencia.

La herencia es la característica por la cual los objetos para su creación se basan en una superclase (clase base), de la cual heredan todas sus atributos y métodos, los cuales a su vez pueden o no ser implementados y/o modificados.

1.4.5. Polimorfismo.

El término de polimorfismo define la capacidad de que más de un objeto pueda crearse usando la misma superclase para lograr dos conceptos de objetos diferentes, por ejemplo los teléfonos, los cuales se basan en un teléfono base, con la capacidad de hacer **ring** y tener un auricular, para luego obtener un teléfono digital,

inalámbrico, con botonera de marcado y también, tomando la misma base, construir un teléfono analógico y con disco de marcado.

Figura 1.9. **Ilustración de Polimorfismo**

1.4.6. Ventajas y Desventajas de la Programación Orientada a Objetos.

No todo es tan bonito como suena en la programación orientada a objetos; si bien se han mencionado ventajas en cuanto a la utilización de la programación orientada a objetos, esta metodología de trabajo cambia completamente el enfoque del desarrollo de los procedimientos de codificación hacia el diseño y definición de clases.

Debido a lo que mencionado recientemente, el hecho de que los objetos son módulos completos de código, pequeños, de alta mantenibilidad y absolutamente independientes del contexto en el que se utilicen, nos brinda la posibilidad de poder trabajar en un equipo de desarrolladores que solo trabajen sobre clases de objetos, sobre las cuales los desarrolladores de aplicaciones se basarán para personalizar sus funcionalidades heredando las bases y luego reunirlas todas en una solución final.

Como podemos apreciar éste fue un claro ejemplo de lo que es la reutilización de código, la programación orientada a objetos nos permite trabajar sobre una biblioteca de clases personalizadas que luego se pueden utilizar como base para futuros desarrollos.

Todas estas ventajas al momento de codificar y producir, tienen en contrapartida un gran esfuerzo al momento del diseño de los mencionados objetos, esto requiere de un

exhaustivo trabajo de análisis que requiere también de mucha disciplina para que los resultados sean exitosos y no una completa catástrofe.

También debemos destacar que la depuración de código orientado a objeto es algo más compleja que la depuración de código estructurado, esto no quiere decir que nuestro código vaya a ser malo o tener más errores, pero sí es cierto que en el caso de producirse un error deberemos recorrer todo el árbol de herencia para encontrarlo y poderlo corregir algo que en programación estructurada no tenemos.

1.5 Introducción a la arquitectura de n-Capas

Las primeras aplicaciones que se construían se denominaban de una sola capa o "Single Tier", en esta arquitectura, la base de datos, las reglas de negocio y la interfaz del usuario estaban embebidas dentro de la misma aplicación, posteriormente aparecieron los servidores de bases de datos, y se hizo por primera vez una distinción entre los datos y la interfaz del cliente, aunque dentro de esta última seguía intrínseca las reglas de negocio, a esta arquitectura se le denominó Cliente-Servidor. Algunos años atrás apareció la arquitectura de tres capas, que propone convertir la interfaz de usuario, las reglas de negocio y la base de datos en entidades separadas y definir correctamente las interfaces que cada una exponga para comunicarse la una con la otra. La idea principal detrás de esta división consiste en aprovechar la escalabilidad que dicha arquitectura nos ofrece, esto es: si tenemos una aplicación que aplique correctamente las reglas de diseño de tres capas, y necesitamos migrar dicha aplicación a un motor de base de datos diferente al original, nada más tendríamos que modificar la capa de datos para que se ajuste a los requerimientos del nuevo motor, quedando intacta la capa de la interfaz de usuario, y como mucho se debería aplicar alguna leve modificación en la capa de negocios.

Finalmente, hoy en día ha aparecido una nueva concepción en arquitectura de aplicaciones, denominada "n-Tier" o de n-capas, la idea consiste en separar definitivamente las reglas de negocio de las de acceso a datos, permitiendo que se incremente la escalabilidad de la aplicación al reducir el impacto de los cambios en algún módulo específico sobre el sistema global.

1.5.1 Objetivos de una aplicación en n-Capas

Básicamente se trata de lograr los siguientes objetivos:

- Si se producen cambios en los métodos base de acceso a datos, el código del lado del cliente debe permanecer intacto.
- Todas las rutinas de acceso a datos deben ser expuestas como objetos en lugar de llamadas a función.
- El lenguaje SQL del lado del cliente debe ser eliminado completamente, en su lugar deben existir métodos y propiedades.
- Los nombres de tablas y de columnas deben ser eliminados en el código del lado del cliente, en su lugar se deben utilizar **conjuntos de datos** que expongan dicha información como propiedades.
- El código del lado del cliente debe ser simplificado, al reducir las llamadas a funciones, por la utilización de métodos y propiedades.

1.5.2 Ventajas de la arquitectura de n-Capas

La aplicación de la arquitectura de n-capas proporciona las siguientes ventajas:

- Abstracción total a cerca del origen de datos
- Bajo costo de desarrollo y mantenimiento de las aplicaciones
- Estandarización de las reglas de negocios
- Mejor calidad de software
- Reutilización del código
- Escalabilidad de aplicaciones

1.5.3 Desventajas de la arquitectura de n-Capas

Aunque las ventajas de construir una aplicación en n-capas son amplias, existen ciertas desventajas:

- Se puede terminar creando demasiadas clases, lo que repercute en el mantenimiento y en el rendimiento, debido a la dificultad de crear nuevas clases en tiempo de ejecución.
- Es necesario conocer muy bien la estructura de las tablas de las que se obtendrán datos.

- La creación de reportes no es una tarea sencilla debido a que los generadores de reportes generalmente no utilizan clases para abastecerse de los datos necesarios.

1.5.4 Arquitectura de Aplicaciones para E-Commerce B2C y .NET

La construcción del la aplicación E-Commerce B2C para la empresa Plantaciones El Trébol contempla la utilización de una arquitectura de tres capas:

1. **La capa de presentación.-** Está conformada por los componentes de IU que básicamente son los elementos con los que puede interactuar el usuario, y los componentes de proceso de IU, que encapsulan la lógica de navegación y el control de eventos de la interfase.
2. **La capa de negocio.-**Encapsula la lógica de negocios, los servicios de esta capa estarán divididos en dos partes: las entidades empresariales que representan objetos que van a ser manejados o consumidos por toda la aplicación y los componentes empresariales, que contienen la lógica de negocio.
3. **La capa de acceso a datos**.- Contiene clases altamente especializadas que interactúan con la base de datos.

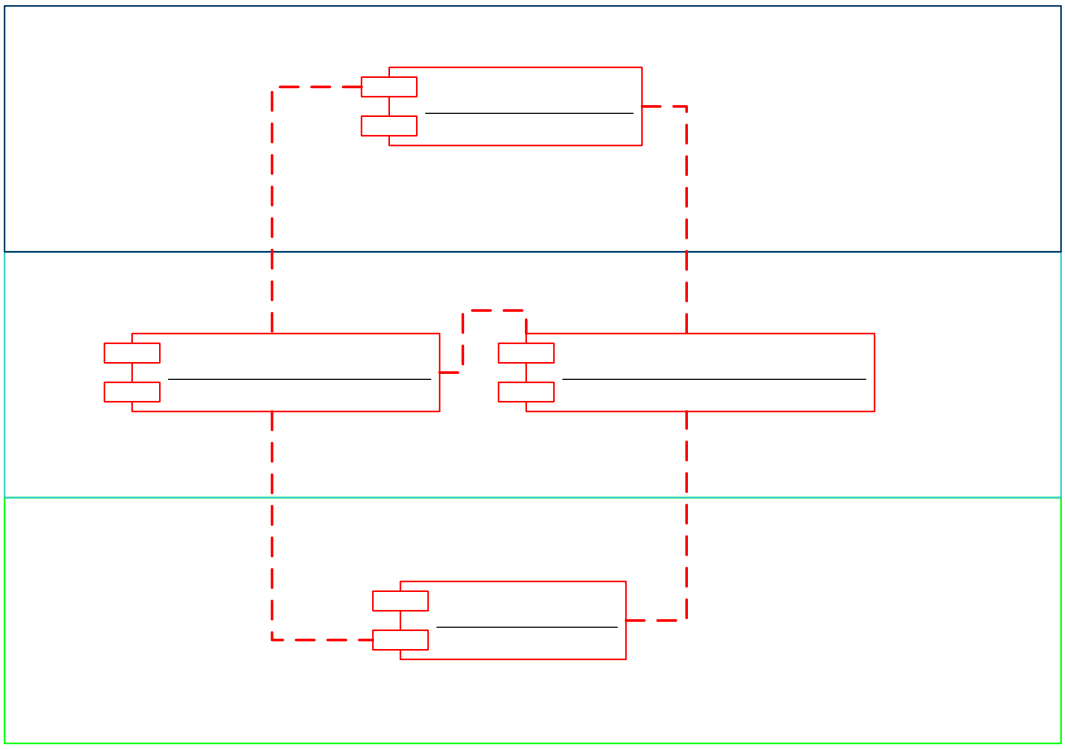


Figura 1.10. Arquitectura de la aplicación Ecommerce B2C

1.6 Introducción a la plataforma de desarrollo .NET

Nos encontramos en un momento especial en la industria de computación ya que estamos en el inicio de una nueva manera de hacer y de integrar las aplicaciones, la necesidad de permitir a las computadoras conectadas a Internet comunicarse entre ellas ha dado forma al nuevo modelo de computación distribuida llamada servicios Web basados en XML. El objetivo es permitir comunicarse entre si a sistemas heterogéneos dentro y fuera de la empresa. Esta comunicación es independiente del sistema operativo, lenguaje o modelo de programación. Para conseguir esto el consorcio de Internet http://www.w3c.org ha desarrollado los siguientes estándares que permiten hacer uso de lo servicios Web:

- XML: (Lenguaje de Marcado eXtensible) Es un formato universal para representar los datos.

- SOAP: (Protocolo Simple de Acceso a Objetos) Es un protocolo que permite mover los datos entre aplicaciones y sistemas. Es el mecanismo por medio del cual los servicios Web son invocados e interactúan.

- UDDI: (Descubrimiento, Descripción e Integración Universal) Lenguaje que permite publicar, encontrar y usar los Servicios Web basados en XML. Es la 'Página Amarilla' de los servicios Web es decir un directorio para poder encontrarlos. Puede ser accedido con un explorador en http://www.uddi.org o programáticamente ya que UDDI es también un servicio Web.

- WSDL: (Lenguaje de Descripción de Servicios Web) Lenguaje por medio del cual un servicio Web describe entre otras cosas qué hace o qué funcionalidad implementa.

La plataforma .NET provee los cimientos para la nueva generación de software. Utiliza los Servicios Web como un medio para poder interoperar a distintas tecnologías. Permite conectar distintos sistemas operativos, dispositivos físicos, información y usuarios. Le da a los desarrolladores las herramientas y tecnologías para hacer rápidamente soluciones de negocios que involucran distintas aplicaciones, dispositivos físicos y organizaciones

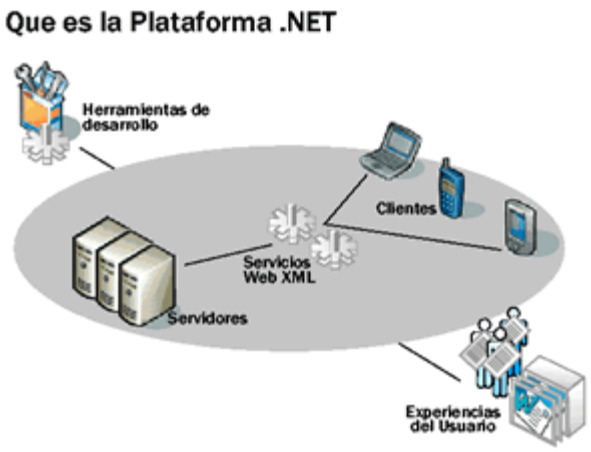


Figura 1.11. La plataforma .NET

La idea central detrás de la plataforma .NET es la construir software como servicio y de cómo instalar, consumir, integrar o agregar estos servicios para que puedan ser accedidos mediante Internet, permitiendo que los usuarios accedan desde cualquier dispositivo, en cualquier sistema operativo y lugar a la funcionalidad que los servicios Web proveen.

1.6.1 Componentes de la plataforma .NET

La plataforma .Net esta formada por los siguientes elementos:

- Clientes inteligentes.- Permiten acceder a la información en el formato apropiado, en cualquier momento y lugar haciendo uso de los Servicios Web, dentro de esta categoría tenemos dispositivos que van desde un teléfono inteligente, hasta los equipos portátiles y personales.

- **Servidores:** Proveen de la infraestructura para implementar el modelo de computación distribuida en Internet. Son sistemas operativos y de aplicación.
- **Servicios Web basados en XML.-** Son componentes que permiten a las aplicaciones compartir datos y pueden ser llamados desde distintos sistemas operativos, plataformas de hardware y lenguajes de programación.

- **Experiencias del usuario:** Es decir, soluciones que implementen los servicios Web para hacer la experiencia de uso de una aplicación más sencilla y centrada en el usuario.

- **Herramientas de desarrollo:** Visual Studio .NET y el .NET Framework. Ambos permiten al desarrollador hacer servicios Web basados en XML además de otro tipo de aplicaciones.

1.6.2 El .NET Framework

Es un conjunto de servicios de programación diseñados para simplificar el desarrollo de aplicaciones en el entorno altamente distribuido de Internet. Los componentes del .NET Framework proveen la base necesaria para construir las aplicaciones Web, los servicios Web y cualquier otra aplicación dentro de Visual Studio .NET.

El **.NET Compact Framework** permite hacer uso de los servicios Web en dispositivos móviles. Debido a que es un subconjunto del .NET Framework comparte el mismo modelo de programación y herramientas de desarrollo de aplicaciones haciendo posible que los desarrolladores transfieran sus conocimientos existentes al desarrollo de aplicaciones móviles.

1.6.3 Visual Studio .NET

Visual Studio .NET es un conjunto de herramientas de desarrollo para la construcción de aplicaciones Web ASP, servicios Web XML, aplicaciones para escritorio y aplicaciones móviles. Todas las herramientas que conforman Visual Studio (Visual Basic .NET, Visual C++ .NET, Visual C# .NET y Visual J# .NET) utilizan el mismo entorno de desarrollo integrado (IDE), que les permite compartir herramientas y facilita la creación de soluciones en varios lenguajes. Así mismo, dichos lenguajes aprovechan las funciones de .NET Framework, que ofrece acceso a tecnologías clave para simplificar el desarrollo de aplicaciones Web ASP y servicios Web XML.

1.6.4 Microsoft Visio para Arquitectos Empresariales

Microsoft Visio para Arquitectos Empresariales es una herramienta complementaria para Visual Studio .NET que aumenta la interoperatividad y la compatibilidad con las herramientas para programadores de Microsoft. Entre otros aspectos, Visio proporciona también herramientas para mejorar la productividad en la creación de diagramas de software y bases de datos.

Entre las múltiples ventajas Visio Ofrece a los desarrolladores de software tenemos:

- Productividad de interfaz.- Permite agregar campos nuevos consecutivamente a las clases de software o las tablas de bases de datos, con lo que se agiliza la entrada de datos.

- Modelado de bases de datos.- Visio es compatible con Microsoft SQL Server 2000, además de con otros varios sistemas de administración de bases de datos (DBMS). En Visio puede generar automáticamente un diagrama completo de bases de datos desde el Asistente para ingeniería inversa, lo que elimina la necesidad de arrastrar tablas desde la ventana Tablas.

- Modelado de software.- En Visio para arquitectos empresariales se han incluido nuevas capacidades para el modelado de software, cuya principal novedad es el soporte para el Lenguaje de Modelado Unificado UML que permite el desarrollo de Diagramas de Actividades, Casos de Uso, etc.

1.6.5 Microsoft SQL Server 2000

Microsoft SQL Server 2000 es un sistema manejador de bases de datos relacional RDBMS que brinda una excelente plataforma para el procesamiento transaccional a gran escala, soporte para data warehousing y aplicaciones de comercio electrónico.

Características como el soporte para XML, soporte para memoria física de hasta 64 GB, mecanismos de respaldo y recuperación, entre otras hacen de SQL Server 2000 el sistema de bases de datos predilecto para la implementación de los sistemas de comercio electrónico.

1.7 Funciones de la empresa Plantaciones El Trébol

Plantaciones El Trébol es una empresa especializada en la producción de rosas de alta calidad, con tallos largos, botones grandes y colores brillantes.

La plantación está ubicada en el valle de Burgay, provincia del Cañar, 40 kilómetros al norte de Cuenca – Ecuador, cuya ubicación ofrece las condiciones ideales para la siembra debido a su suelo fértil, la ausencia de vientos fuertes, y el contar con doce

horas diarias de luz natural durante todo el año, componen el clima ideal para la actividad productiva de las rosas.

Mediante la aplicación de la norma ISO 9002, se mantiene un control continuo de todo el proceso de producción, que contempla tanto la siembra, la cosecha, hasta el embarque de la flor para sus clientes, en diversas partes del mundo como son: Alemania, Rusia, Estados Unidos de América, entre otras.

La aplicación de la norma ISO 9002 ha servido para:

- Garantizar la uniformidad, el brillo de los colores, la frescura y la longitud de sus tallos, especialmente en 60, 70, 80 y 90 cm., así como también el tamaño de sus botones de 6 o más centímetros.
- Mejorar los procesos a través de la documentación y el análisis estadístico
- Fortalecer la relación con el cliente, a través de un compromiso a largo plazo que garantiza el profesionalismo, la honestidad y un adecuado reparto de sus órdenes.

El proceso de empaque de la flor, sigue un riguroso procedimiento de selección, que garantiza la calidad, posteriormente dicha flor es cuidadosamente transportada en vehículos especializados que cuentan con acondicionadores de temperatura, hasta las diversas agencias de envío las cuales se encargan de hacer llegar la mercadería a su destino.

1.7.1 Necesidades

El continuo proceso de mejora de la empresa y el gran desarrollo de las tecnologías de la información como el Internet, exigen a la compañía la implantación de nuevas formas de hacer negocios y que permitan solventar ciertas inconsistencias en el flujo de los datos, que tienen que ver sobre todo con los procedimientos actuales para la recepción de las órdenes de los clientes y su procesamiento, lo cual nos plantea la necesidad de establecer un mecanismo eficiente que nos permita mantener un alto grado de interacción con nuestros clientes, de forma que sea posible:

- Presentar las variedades de nuestro producto al mundo de forma que nos permita la captación de nuevos mercados y clientes.
- Publicar nuestro stock en las diferentes variedades.
- Dar a conocer las novedades que acontecen en la empresa, así como los planes a futuro en la producción de nuevas variedades.
- Fortalecer nuestra relación con los clientes, proporcionándoles nuevos e innovadores recursos que les permitan trabajar con mayor comodidad y que les incentive a utilizar nuestros servicios.
- Realizar pedidos en línea a cualquier hora del día y desde cualquier parte del mundo.
- Mantener informados a nuestros clientes a cerca del estado de sus órdenes, durante todo el proceso que va desde la recepción de dicha orden, hasta el empaque y el envío de la mercadería a su destino.
- Proporcionar a nuestros clientes un mecanismo que les permita realizar el seguimiento permanente de sus pedidos, para de esta forma reducir la constante dependencia del agente de ventas, con la dificultad agregada de las diferencias horarias.
- Garantizar que la información tanto emitida como la recibida sea confiable, consistente y libre de amenazas informáticas.
- Solventar los problemas lingüísticos existentes entre las distintas partes del mundo de la que provienen nuestros clientes, mediante la utilización del idioma más utilizado en el mundo de los negocios, como es el inglés.
- Aprovechar al máximo los recursos tecnológicos con los que cuenta la empresa, como es el caso de nuestro servicio de Internet, el espacio Web contratado, etc.

Capítulo 2
Análisis del Sistema

El objetivo de esta especificación es definir de manera clara los requisitos empresariales que orientarán el diseño de la aplicación Web E-Commerce B2C, así como las funcionalidades y condicionantes técnicas del sistema.

Se analizan los aspectos importantes del negocio, así como las exigencias de los usuarios finales de la aplicación, de manera que se cumplan con sus expectativas y se logre un sistema eficiente y completamente funcional.

2.1 Conocimiento de los requerimientos

En un mundo de continuo y rápido movimiento como es el de los negocios en Internet, un sitio de comercio electrónico significa mucho más que una tienda basada en el Web. El usuario es cada vez más exigente con este tipo de sitios y si uno de ellos no se corresponde con sus expectativas, optará por visitar cualquier otro. Por lo tanto, para llevar a cabo un buen diseño y construcción de una aplicación de comercio electrónico, debemos considerar, tanto los requerimientos empresariales; así como lo que demanda el usuario en un sitio de comercio electrónico. En la siguiente lista se incluyen algunos de los principales aspectos que afectarán al diseño de la aplicación:

- Facilidad de uso y exploración
- Rápido rendimiento
- Compras anónimas
- Perfil de usuario
- Seguridad eficaz
- Competitividad gracias a la facilidad de uso

2.2 Especificación de los requisitos

2.2.1 Ámbito del sistema

La aplicación E-Commerce B2C para Plantaciones El Trébol, se diseñará para ejecutarse en un servidor de hosting compartido, y permitirá el registro de clientes, disponibilidad de productos, colocación de pedidos por parte de los clientes, procesamiento de dichos pedidos por los agentes vendedores; y entre otras cosas, permitirá hacer un seguimiento continuo de las ventas, tanto por parte del cliente, como del agente vendedor.

El objetivo es definir y analizar el sistema contemplando su funcionalidad y necesidades fundamentales.

2.2.2 Definiciones y acrónimos

2.2.2.1 Definiciones

Cliente	Persona que utiliza un servicio que presta la empresa.
Agente Vendedor	Persona encargada de revisar, procesar y confirmar la información generada por el cliente.
Administrador del Sistema	Persona encargada del mantenimiento de la aplicación y de la información que en ella se genera.

Tabla 2.1. Definiciones.

2.3 Requisitos Específicos

2.3.1 Gestión de Agentes Vendedores.

R2.3.1.1 El sistema deberá permitir el ingreso de nuevos agentes vendedores, y registrar información como el número de cédula, la dirección, y el teléfono.

R2.3.1.2 El sistema deberá permitir la modificación de la información del agente vendedor.

R2.3.1.3 El sistema permitirá dar de baja los agentes vendedores registrados, manteniendo un historial de sus ventas.

Tabla 2.2. Gestión de Agentes Vendedores.

2.3.2 Gestión de Clientes.

R2.3.2.1 El sistema deberá permitir el registro de nuevos clientes, registrando información como su número de documento de identidad (Código), sus nombres completos, su dirección, teléfono, país de origen y e-mail.

R2.3.2.2 Deberá permitir la modificación de la información del cliente.

R2.3.2.3 Deberá permitir especificar el tipo de mercado al que pertenece el cliente, que puede ser mercado ruso y otros.

R2.3.2.3 Deberá solicitar el idioma predeterminado de la interfaz que se le presentará al usuario.

Tabla 2.3. Gestión de Clientes.

2.3.3 Gestión de Productos.

R2.3.3.1 El sistema deberá permitir el ingreso de la información de los productos, registrando información como la variedad, el tamaño del botón, el color, si es bicolor, etc.

R2.3.3.2 Deberá permitir la modificación de la información ingresada

R2.3.3.3 Deberá permitir dar de baja las variedades de flor ingresada, manteniendo un historial de las ventas.

Tabla 2.4. Gestión de Productos.

2.3.4 Autenticación de Clientes.

2.3.4.1 El sistema deberá pedir a los clientes el nombre de usuario y su contraseña, previa la colocación de un pedido.

Tabla 2.5. Autenticación de Clientes.

2.3.5 Mantenimiento de la disponibilidad de los Productos.

2.3.5.1 El sistema deberá permitir registrar la disponibilidad de los productos

2.3.5.2 Permitirá la modificación de la disponibilidad ingresada

2.3.5.2 Permitirá la eliminación de la disponibilidad ingresada

Tabla 2.6. Mantenimiento de la disponibilidad de los Productos.

2.3.6 Colocación de Pedidos.

2.3.6.1 El sistema deberá permitir la colocación de pedidos por parte del cliente previamente autenticado.

2.3.6.2 Deberá permitir cambiar la información del pedido antes de confirmar su colocación.

2.3.6.3 Solicitará que el cliente confirme su decisión de colocar el pedido antes de guardarlo.

2.3.6.4 Permitirá al cliente colocar información adicional que desee que aparezca en su pedido, como la marcación de las cajas, y el punto de apertura del producto.

2.3.6.5 El sistema permitirá al agente vendedor revisar los pedidos colocados por los clientes, y realizar la confirmación de cantidades solicitadas y precios.

2.3.6.6 Deberá permitir la colocación de observaciones, información de vuelo y novedades en los pedidos.

2.3.6.7 Permitirá a los clientes consultar el estado de sus pedidos, para saber si se confirmaron, la información del vuelo, etc.

Tabla 2.7. Colocación de Pedidos.

2.3.7 Perfil de Cliente.

2.3.4.1 El sistema deberá permitir guardar el perfil del cliente, como el idioma de la interfaz, y la información comercial y de contacto del mismo.

2.3.4.2 Permitirá modificar las preferencias del cliente, permitiéndole cambiar la información comercial y de contacto previamente almacenado y el idioma de la interfaz.

2.3.4.3 No permitirá cambiar las contraseñas de acceso por parte del cliente, puesto que dichas claves serán asignadas por el agente vendedor asignado.

2.3.4.4 Permitirá al cliente consultar el estado de su cuenta.

Tabla 2.8. Perfil de Cliente.

2.3.8 Interfaz de Navegación.

2.3.5.1 El sistema deberá permitir seleccionar el idioma de la interfaz de navegación, la misma que estará disponible en español e inglés.

2.3.5.2 Dispondrá de un buscador, que acepte un criterio de búsqueda y devuelva los resultados obtenidos dentro del sitio.

2.3.5.3 Deberá mostrar un menú de navegación fácil e intuitivo, que permita localizar las distintas opciones que ofrece la aplicación.

2.3.5.4 Dispondrá de un área en donde se pueda resaltar los últimos cambios en el sitio, y las novedades de la empresa.

2.3.5.5 Dispondrá de un área que publique aspectos afines a la empresa, como su historia, y las certificaciones de calidad, información de contacto, entre otros.

2.3.5.6 Permitirá al administrador del sistema el diseño de formularios de encuestas, y su publicación.

Tabla 2.9. Interfaz de Navegación.

2.3.9 Requerimientos no funcionales.

2.3.6.1	El sistema deberá ser compatible con los navegadores Internet Explorer, Mozila, y Mozila Firefox.
2.3.6.2	Deberá manejar un algoritmo de encriptamiento de la información importante, como las contraseñas.
2.3.6.3	Se adaptará al esquema de seguridad manejado por plantaciones El Trébol.

Tabla 2.10. Requerimientos no funcionales.

2.4 Casos de Uso: Descripción de Procesos.

2.4.1 Identificación de Actores.

Los actores que intervendrán en el sistema serán los siguientes:

Clientes	Son las personas que generan información referente a las órdenes de ventas que pueden ser realizadas por la empresa.
Agente Vendedor	Son las personas encargadas de realizar la confirmación de las órdenes de ventas realizadas por los clientes, para que las mismas puedan ser despachadas a los respectivos clientes.
Administrador del Sistema	Es la persona encargada de manejar el sistema, realizará todos los mantenimientos del sistema, como: Productos, Clientes, Agentes Vendedores, además será el encargado de subir al sistema la información referente a las novedades, historia de la empresa, etc.

Tabla 2.11. Actores del Sistema.

2.4.2 Identificación de los casos de uso.

2.4.2.1 Casos de Uso Primarios.

Caso de Uso:	Ingreso de Agentes Vendedores.
Actores:	Administrador del Sistema.
Tipo:	Primario

Descripción:	El Administrador del Sistema deberá ingresar los datos de los agentes vendedores dentro de los cuales se incluyen los siguientes: ➢ Número de Cédula. ➢ Nombre. ➢ Dirección.

Tabla 2.12. Caso de Uso Ingreso de Agentes Vendedores.

Caso de Uso:	Modificación de Agentes Vendedores.
Actores:	Administrador del Sistema.
Tipo:	Primario
Descripción:	El Administrador del Sistema deberá poder actualizar los datos de los Agentes Vendedores para poder contar con la información actualizada de cada uno de ellos, entre los datos que pueden ser actualizados se encuentran los siguientes: ➢ Nombre. ➢ Dirección. El número de cédula del Agente Vendedor no podrá ser modificado, ya que está es la clave principal de los mismos y por lo tanto deben ser únicos.

Tabla 2.13. Caso de Uso Modificación de Agentes Vendedores.

Caso de Uso:	Eliminación de Agentes Vendedores.
Actores:	Administrador del Sistema.
Tipo:	Primario.
Descripción:	El Administrador del Sistema tendrá la capacidad de dar de baja los datos de los Agentes Vendedores, pero deberá mantener un registro histórico de las ventas realizadas por el Agente Vendedor en cuestión.

Tabla 2.14. Caso de Uso Eliminación de Agentes Vendedores.

Caso de Uso:	Ingreso de Clientes.
Actores:	Administrador del Sistema.
Tipo:	Primario.
Descripción:	El Administrador del Sistema deberá ingresar los datos de los clientes de la empresa que serán los que generaran la información

referente a las órdenes de ventas, entre los datos de los clientes se registrarán los siguientes:

- Código (Número de documento de identificación).
- Nombres.
- Dirección.
- Teléfono.
- País de origen.
- E-mail.

Tabla 2.15. Caso de Uso Ingreso de Clientes.

Caso de Uso:	Modificación de Clientes.
Actores:	Administrador del Sistema, Clientes.
Tipo:	Primario.
Descripción:	El Administrador del Sistema deberá poder actualizar los datos de los clientes de la empresa para poder contar con la información actualizada de cada uno de ellos, entre los datos que pueden ser actualizados se encuentran los siguientes: ➢ Nombre. ➢ Dirección. ➢ Teléfono. ➢ País de origen. ➢ E-mail. El código (Número de documento de identificación) del Cliente no podrá ser modificado, ya que está es la clave principal de los mismos y por lo tanto deben ser únicos.

Tabla 2.16. Caso de Uso Modificación de Clientes.

Caso de Uso:	Especificar tipo de mercado.
Actores:	Administrador del Sistema.
Tipo:	Primario**.**
Descripción:	El Administrador del Sistema deberá especificar el tipo de mercado al que pertenece el Cliente que puede ser mercado ruso u otros.

Tabla 2.17. Caso de Uso Especificar tipo de mercado.

Caso de Uso:	Ingreso de Productos.
Actores:	Administrador del Sistema.
Tipo:	Primario**.**
Descripción:	El Administrador del Sistema deberá ingresar los datos de los productos que la empresa ofrece para la venta a sus clientes, entre los datos que se deberán ingresar reencuentran los siguientes: ➢ Código. ➢ Variedad. ➢ Tamaño. ➢ Color. ➢ Bicolor. El campo bicolor se lo podrá tratar como un campo de tipo lógico dentro del sistema.

Tabla 2.18. Caso de Uso Ingreso de Productos.

Caso de Uso:	Modificación de Productos.
Actores:	Administrador del Sistema.
Tipo:	Primario**.**
Descripción:	El Administrador del Sistema deberá poder actualizar los datos de los Productos para poder contar con la información actualizada de cada uno de ellos, entre los datos que pueden ser actualizados se encuentran los siguientes: ➢ Variedad. ➢ Tamaño. ➢ Color. ➢ Bicolor. El Código del Producto no podrá ser modificado, ya que está es la clave principal de los mismos y por lo tanto deben ser únicos.

Tabla 2.19. Caso de Uso Modificación de Productos.

Caso de Uso:	Eliminación de Productos.
Actores:	Administrador del Sistema.
Tipo:	Primario.
Descripción:	El Administrador del Sistema deberá poder dar de baja los datos de

los Productos, pero deberá mantener un registro histórico de las ventas realizadas del producto en cuestión.

Tabla 2.20. Caso de Uso Eliminación de Productos.

Caso de Uso:	Autentificación de Usuarios.
Actores:	Clientes.
Tipo:	Primario.
Descripción:	El Cliente deberá registrarse en el sistema para poder realizar las órdenes de ventas, para lo cual el Cliente deberá ingresar los siguientes datos: ➢ Nombre de Usuario. ➢ Contraseña. Para que el cliente pueda autentificarse deberá estar previamente registrado como cliente de la empresa y se le deberá asignar un nombre de usuario y contraseña por parte del administrador del sistema.

Tabla 2.21. Caso de Uso Autenticación de Usuarios.

Caso de Uso:	Registro de la disponibilidad del producto.
Actores:	Administrador del Sistema.
Tipo:	Primario.
Descripción:	El sistema deberá permitir al Administrador, registrar la disponibilidad del producto o existencia.

Tabla 2.22. Caso de Uso Registro de la disponibilidad del producto.

Caso de Uso:	Modificación de la disponibilidad del producto.
Actores:	Administrador del Sistema.
Tipo:	Primario.
Descripción:	El sistema deberá permitir al Administrador del mismo modificar la información pertinente a la disponibilidad del producto.

Tabla 2.23. Caso de Uso Modificación de la disponibilidad del producto.

Caso de Uso:	Eliminación de la disponibilidad del producto.
Actores:	Administrador del Sistema.
Tipo:	Primario.

Descripción:	El sistema deberá permitir al Administrador del mismo eliminar la información de la disponibilidad del producto.

Tabla 2.24. Caso de Uso eliminación de la disponibilidad del producto.

Caso de Uso:	Colocación de Pedidos.
Actores:	Clientes, Agentes Vendedores.
Tipo:	Primario.
Descripción:	Para la colocación del pedido el sistema deberá permitir la autentificación del cliente para que pueda realizar las órdenes de venta, el Cliente podrá escoger los productos que desea ponerlos dentro de la orden de venta, el Cliente también podrá realizar la modificación de los datos del pedido antes de la confirmación de los mismos, el Sistema deberá pedir al Cliente la confirmación de la colocación del pedido antes de guardar los datos del mismo, el Cliente podrá introducir información adicional en su pedido como por ejemplo el marcado de las cajas, punto de apertura, etc., el Sistema le permitirá al Agente Vendedor revisar los datos de los pedidos colocados por parte de los Clientes y además el Agente Vendedor deberá realizar la confirmación de los precios y cantidades de los productos solicitados por parte del cliente, el Sistema deberá permitir colocar información adicional al pedido como por ejemplo observaciones, información del vuelo, novedades del producto, etc., además el Sistema permitirá al Cliente consultar el estado de sus pedidos para poder constatar la confirmación de los mismos y en caso de que hayan sido confirmados consultar la información del vuelo, etc.

Tabla 2.25. Caso de Uso Colocación de Pedidos.

Caso de Uso:	Estado de Cuenta.
Actores:	Administrador del Sistema, Cliente, Agentes Vendedores.
Tipo:	Primario.
Descripción:	El Administrador del Sistema, el Cliente y el Agente Vendedor podrán consultar el estado de cuenta de los clientes, el Administrador del Sistema podrá consultar el Estado de Cuenta de cualquier Cliente

de la empresa, mientras que los Clientes solo podrán consultar su respectivo estado de cuentas.

Tabla 2.26. Caso de Uso Estado de Cuenta.

2.4.2.2 Casos de Uso Secundarios

Caso de Uso:	Seleccionar el Idioma de la Interfaz.
Actores:	Cliente.
Tipo:	Secundario.
Descripción:	El Cliente deberá especificar el idioma de la interfaz, ya que de esto dependerá el idioma en el cual se le mostrara el sistema al Cliente.

Tabla 2.27. Caso de Uso Seleccionar Idioma de la Interfaz.

Caso de Uso:	Ingreso de Perfil de Cliente.
Actores:	Administrador del Sistema.
Tipo:	Secundario.
Descripción:	El Administrador del Sistema deberá ingresar los datos de los Perfiles de Clientes entre los cuales se incluyen los siguientes:

- Idioma de la interfaz.
- Información comercial.
- Nombre de usuario.
- Contraseña

Tabla 2.28. Caso de Uso Ingreso de Perfil de Cliente.

Caso de Uso:	Modificación de Perfil de Cliente.
Actores:	Administrador del Sistema, Cliente.
Tipo:	Secundario.
Descripción:	El Administrador del Sistema y el Cliente deberán poder actualizar los datos de los Perfiles de Clientes para poder contar con la información actualizada de cada uno de ellos, el cliente tendrá algunos limitantes para realizar la modificación de los datos de los Perfiles de Clientes, entre los datos que pueden ser actualizados se encuentran los siguientes:

- Cliente.
  - Idioma de la interfaz.
- Administrador del Sistema.
  - Idioma de la interfaz.
  - Información comercial.
  - Nombre de usuario.
  - Contraseña.

Tabla 2.29. Caso de Uso Modificación de Perfil del Cliente.

Caso de Uso:	Interfaz de Navegación.
Actores:	Administrador del Sistema, Cliente.
Tipo:	Secundario.
Descripción:	El sistema permitirá seleccionar el idioma de la interfaz de navegación, la misma que podrá ser en Español e Inglés, el sistema dispondrá de un buscador en el cual el usuario del mismo podrá introducir el criterio de búsqueda y el sistema le devolverá los resultados obtenidos de la búsqueda dentro del sitio, el sistema dispondrá de un menú de navegación fácil e intuitivo que permita a los diferentes usuarios localizar rápidamente las opciones que ofrezca el sistema, además dispondrá de un área en la cual se podrán colocar los cambios efectuados al sitio y las novedades de la empresa, además se podrá publicar información de la empresa como por ejemplo la historia de la misma, certificaciones de calidad obtenidas por la empresa, información de contactos entre otras, por último el sistema permitirá al Administrador del mismo el diseño de en cuentas para poder aplicarlas a los clientes y la publicación de las mismas.

Tabla 2.30. Caso de Uso Interfaz de Navegación...

2.5 Diagramas de los Casos de Uso.

2.5.1 Caso de Uso Gestión de Agentes Vendedores.

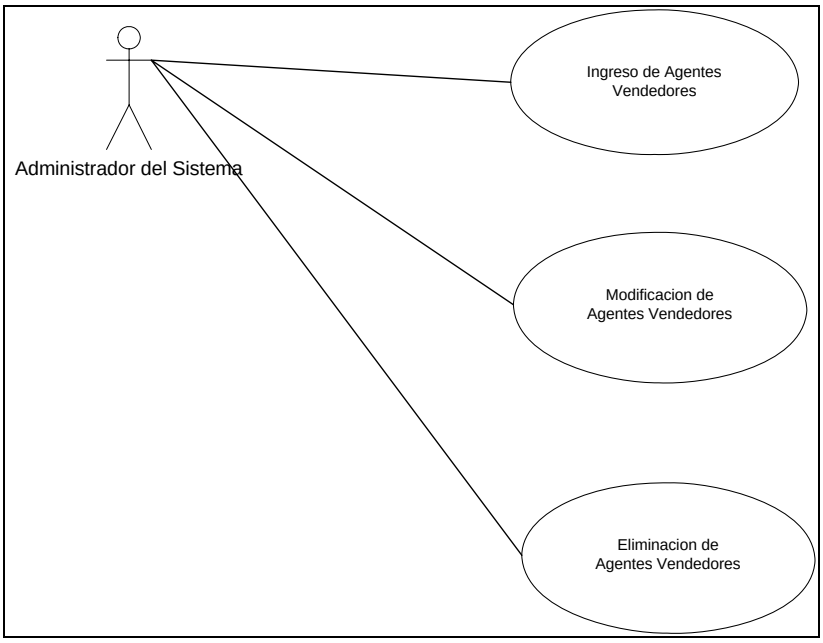


Figura 2.1. Caso de Uso Gestión de Agentes Vendedores.

2.5.2 Caso de Uso Gestión de Clientes.

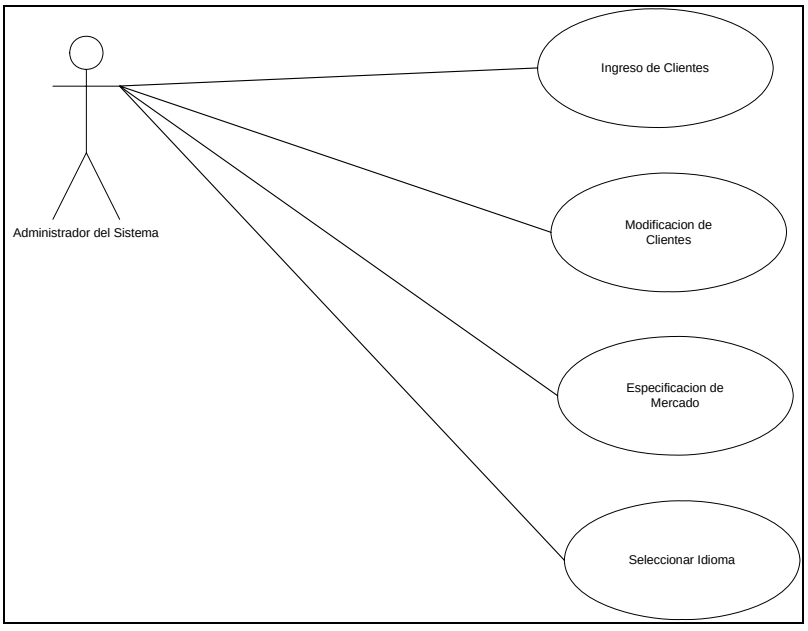


Figura 2.2. Caso de Uso Gestión de Clientes.

2.5.3 Caso de Uso Gestión de Productos.

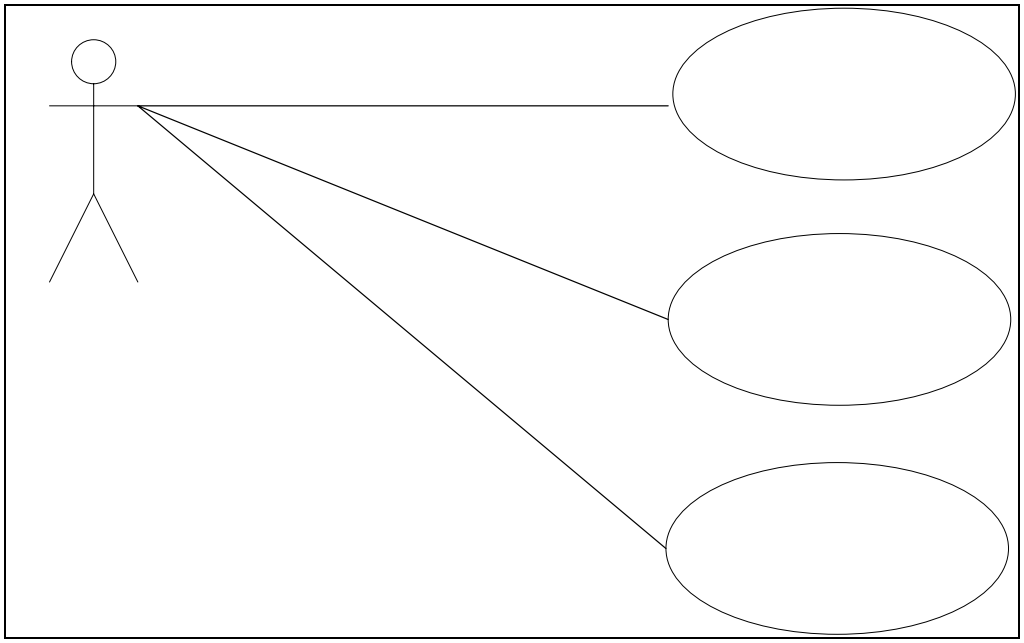


Figura 2.3. Caso de Uso Gestión de Productos.

2.5.4 Caso de Uso Autentificación de Usuarios.

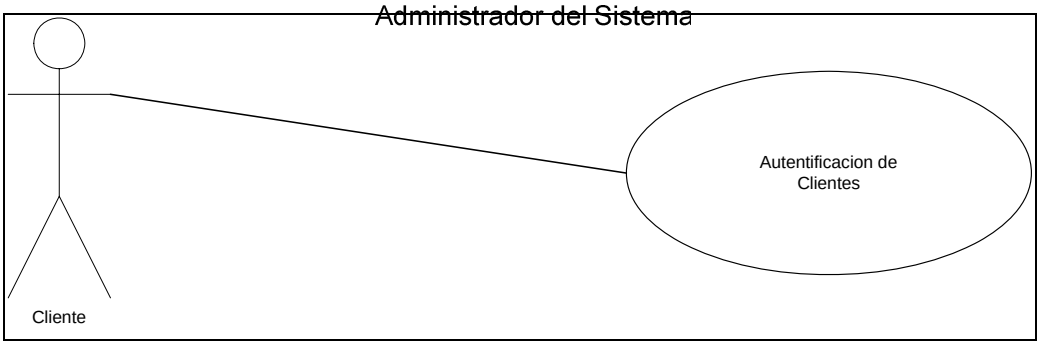


Figura 2.4. Caso de Uso Autenticación de Usuarios.

2.5.5 Caso de Uso Registro de Disponibilidad.

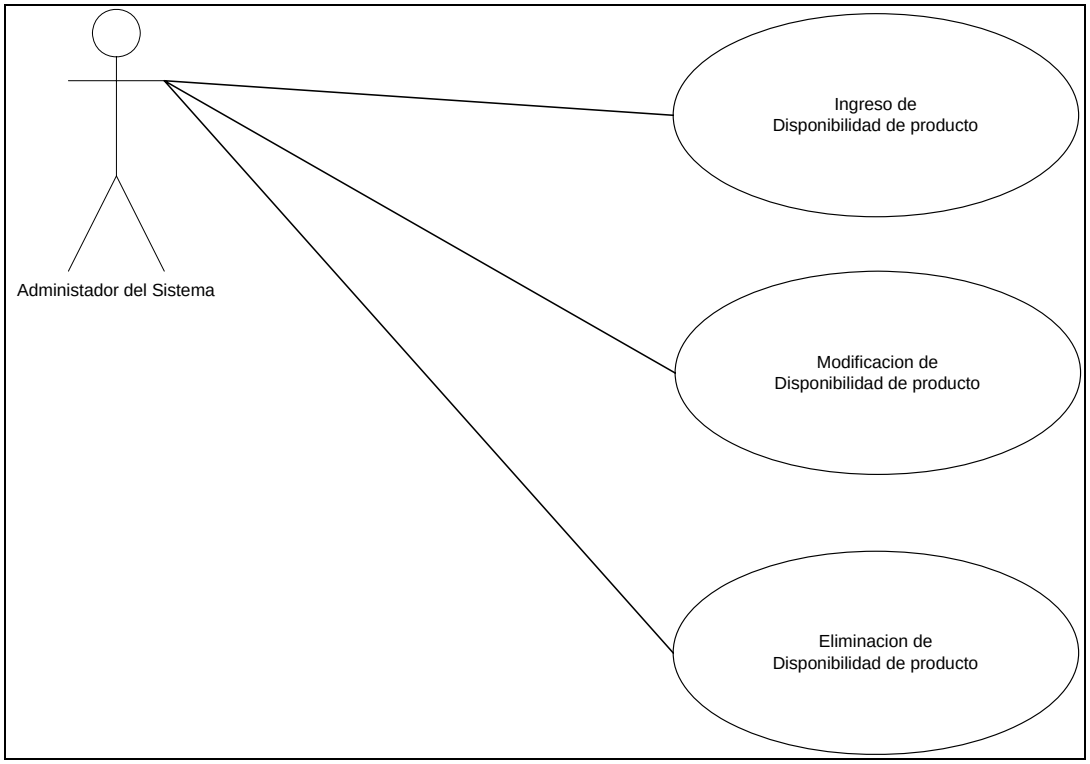


Figura 2.5. Caso de Uso Registro de la Disponibilidad.

2.5.6 Caso de Uso Colocación de Pedidos.

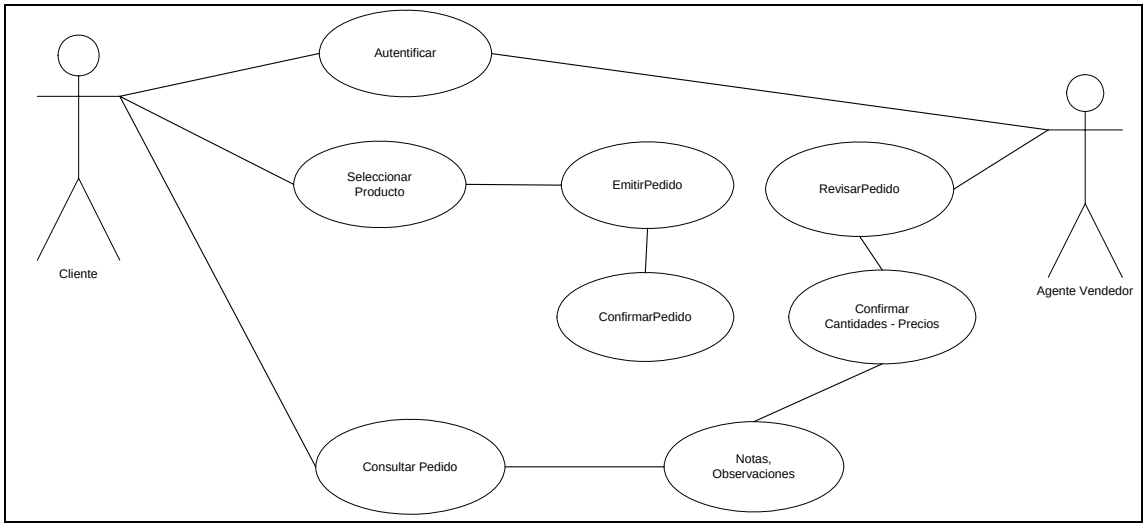


Figura 2.6. Caso de Uso Colocación de Pedidos.

2.5.7 Caso de Uso Perfil del Cliente (Administrador).

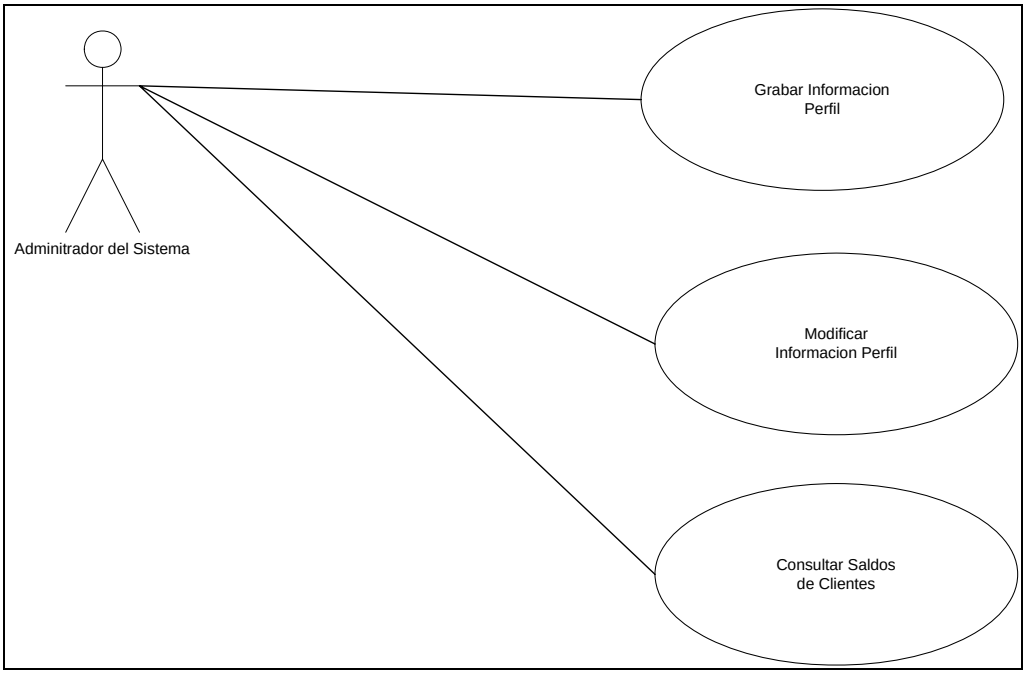


Figura 2.7. Caso de Uso Perfil del Cliente (Administrador).

2.5.8 Caso de Uso Perfil del Cliente (Cliente).

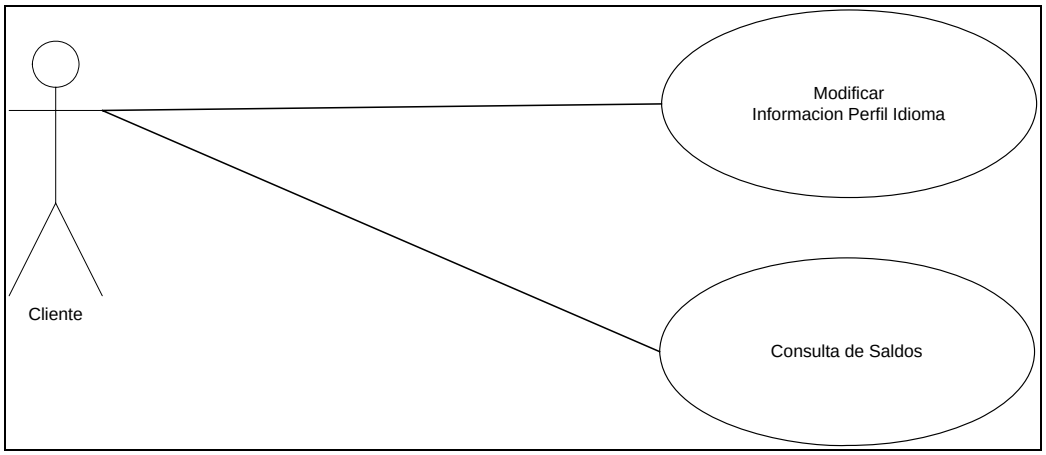


Figura 2.8. Caso de Uso Perfil del Cliente (Cliente).

2.5.9 Caso de Uso Interfaz de Navegación (Administrador).

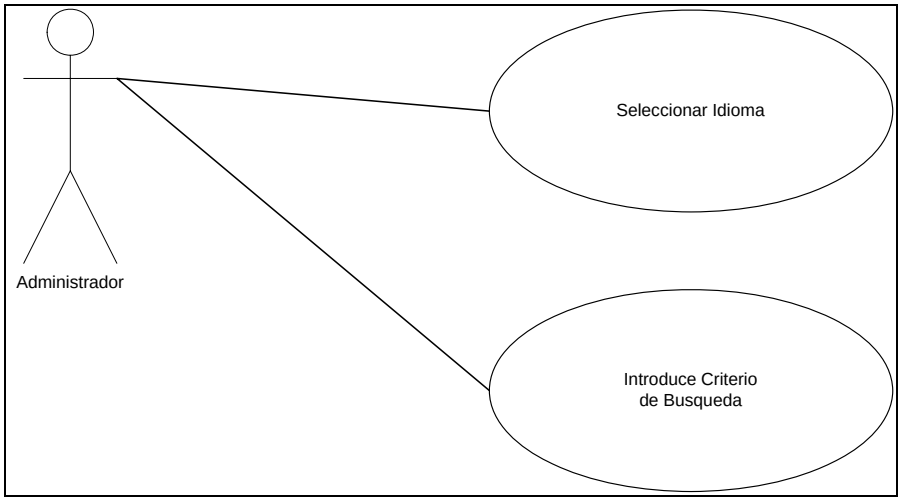


Figura 2.9. Caso de Uso Interfaz de Navegación (Administrador).

2.5.10 Caso de Uso Interfaz de Navegación (Cliente).

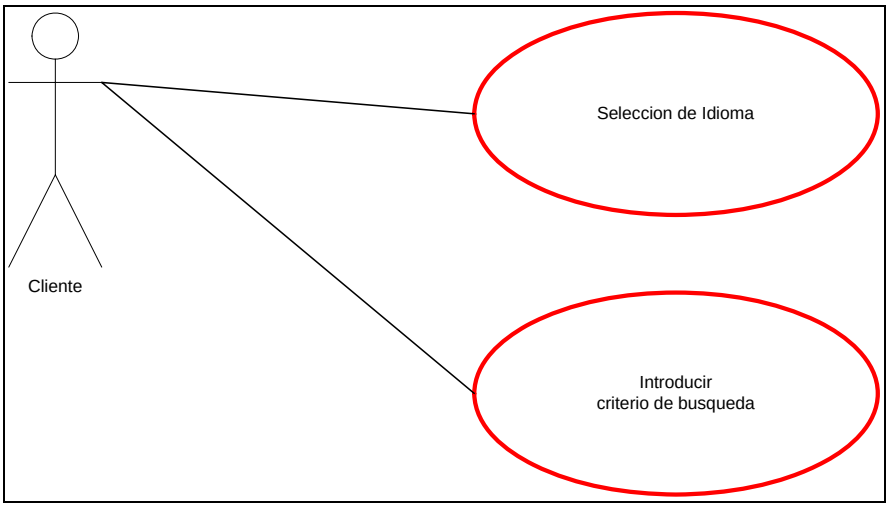


Figura 2.10. Caso de Uso Interfaz de Navegación (Cliente).

Capítulo 3
Diseño del Sistema

En este capítulo se mostrará la interacción entre los objetos del sistema, a través de los diagramas de secuencia. Mientras que los diagramas de caso de uso permiten el modelado de una vista global del escenario, el diagrama de secuencia contiene detalles de implementación del escenario, incluyendo los objetos y clases que se usan para implementar el escenario, y mensajes pasados entre los objetos.

Mediante el diagrama de despliegue se mostrarán las relaciones físicas entre los componentes hardware y software en el sistema final.

Mediante el diagrama de clases se presentarán las clases del sistema con sus relaciones estructurales y de herencia.

Mediante el Modelo Entidad Relación se ilustra el esquema físico que tendrá la Base de Datos.

3.1. Diagramas de Secuencia.

3.1.1 Escenario para el ingreso de agentes vendedores.

- El administrador del sistema ingresa su nombre de usuario y contraseña para poder ingresar al sistema.
- El administrador del sistema ingresa los datos del agente vendedor.
- El sistema pide la confirmación de los datos del agente vendedor.
- El administrador del sistema confirma los datos del agente vendedor.
- El sistema devuelve mensaje de datos guardados.

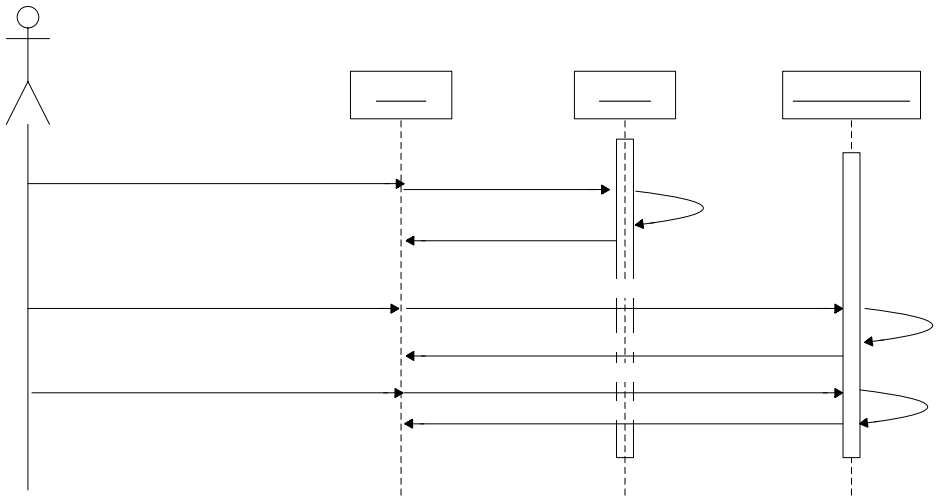


Figura 3.1. Diagrama de Secuencia Ingreso de Agentes Vendedores.

Inte

Paquete superior::Administrador

* Ingresa nombre de usuario y contraseña

Ingresa los datos del Agente Vendedor

3.1.2. Escenario para la modificación de agentes vendedores.

- El administrador del sistema ingresa nombre de usuario y contraseña
- El administrador del sistema ingresa el nombre o código del agente vendedor a modificar.
- El sistema devuelve los datos del agente vendedor a modificar.
- El administrador del sistema ingresa los nuevos datos del agente vendedor.
- El sistema pide confirmación de los datos.
- El administrador del sistema confirma los datos.
- El sistema devuelve mensaje de datos guardados.

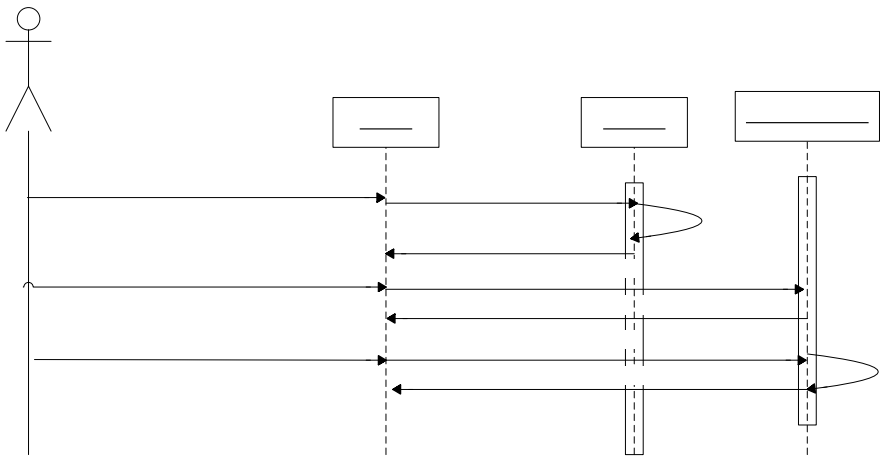


Figura 3.2. Diagrama de Secuencia Modificación de Agentes Vendedores.

3.1.3. Escenario para la eliminación de agentes vendedores.

- El administrador del sistema ingresa nombre de usuario y contraseña.
- El administrador del sistema ingresa el código o nombre del agente vendedor a ser eliminado.
- El sistema devuelve los datos del agente vendedor a ser eliminado.
- El sistema pide confirmación de eliminación de los datos del agente vendedor.
- El administrador del sistema confirma la eliminación de los datos del agente vendedor.

Inter

Paquete superior::Administrador

*

Ingresa nombre de usuario y contrasena

Ingresa nombre o codigo del agente vendedor

- El sistema devuelve mensaje de datos eliminados.

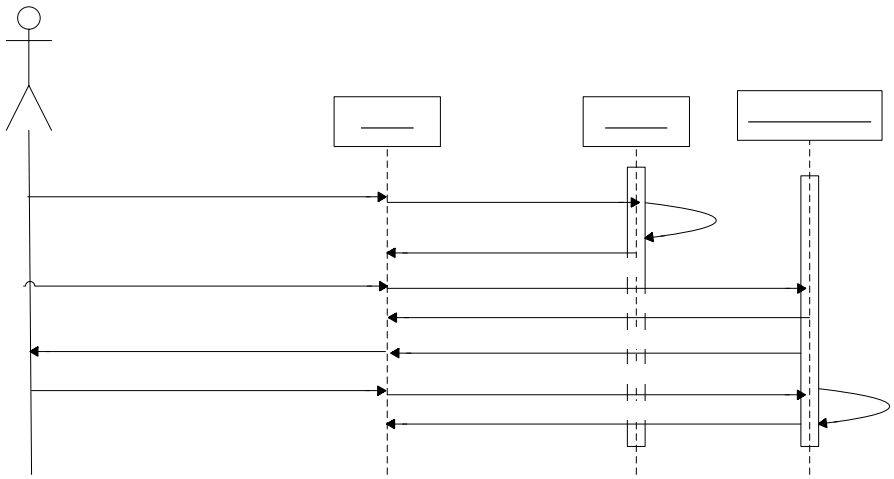


Figura 3.3. Diagrama de Secuencia Eliminación de Agentes Vendedores.

3.1.4. Escenario para el ingreso de clientes.

- El administrador del sistema ingresa su nombre de usuario y contraseña para poder ingresar al sistema.
- El administrador del sistema ingresa los datos del cliente.
- El sistema pide la confirmación de los datos del cliente.
- El administrador del sistema confirma los datos del cliente.
- El sistema devuelve mensaje de datos guardados.

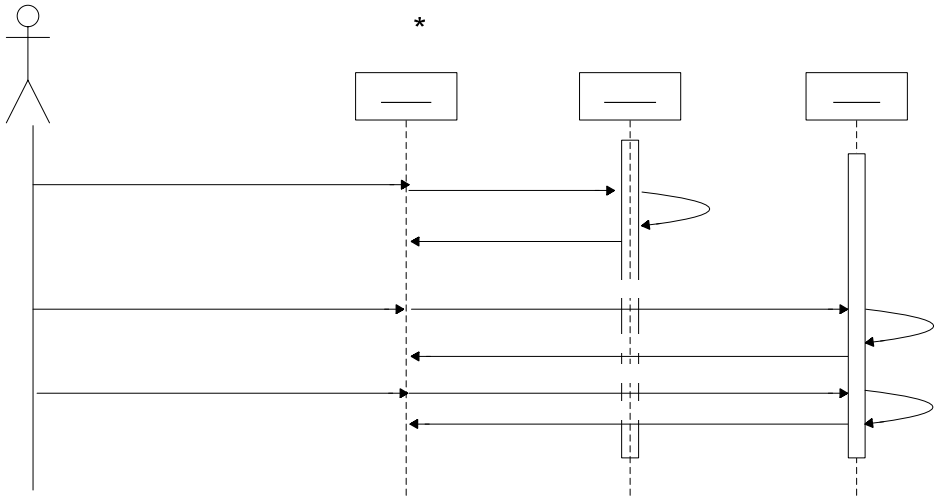


Figura 3.4. Diagrama de Secuencia Ingreso de Clientes.

3.1.5. Escenario para la modificación de clientes.

- El administrador del sistema ingresa nombre de usuario y contraseña
- El administrador del sistema ingresa el nombre o código del cliente a modificar.
- El sistema devuelve los datos del cliente a modificar.
- El administrador del sistema ingresa los nuevos datos del cliente.
- El sistema pide confirmación de los datos.
- El administrador del sistema confirma los datos.
- El sistema devuelve mensaje de datos guardados.

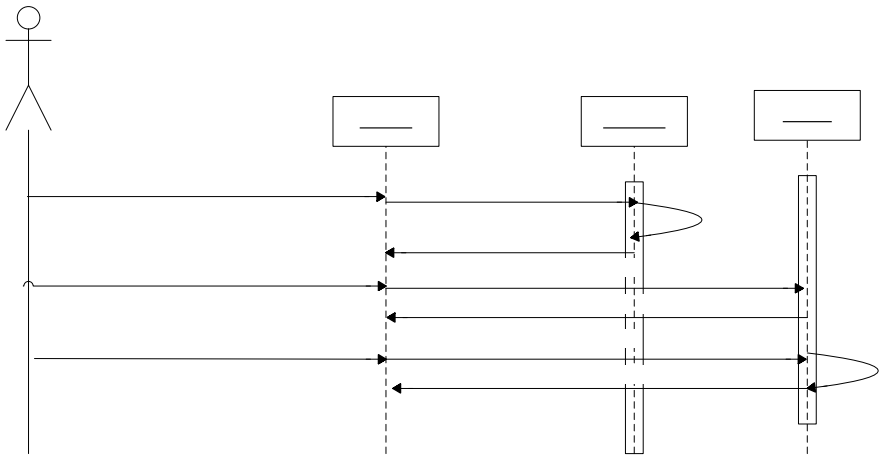


Figura 3.5. Diagrama de Secuencia Modificación de Clientes.

3.1.6. Escenario para la eliminación de clientes.

- El administrador del sistema ingresa nombre de usuario y contraseña.
- El administrador del sistema ingresa el código o nombre del cliente a ser eliminado.
- El sistema devuelve los datos del cliente a ser eliminado.
- El sistema pide confirmación de eliminación de los datos del cliente.
- El administrador del sistema confirma la eliminación de los datos del cliente.
- El sistema devuelve mensaje de datos eliminados.

Interf

Paquete superior::Administrador

*

Ingresa nombre de usuario y contrasena

Ingresa nombre o codigo del Cliente

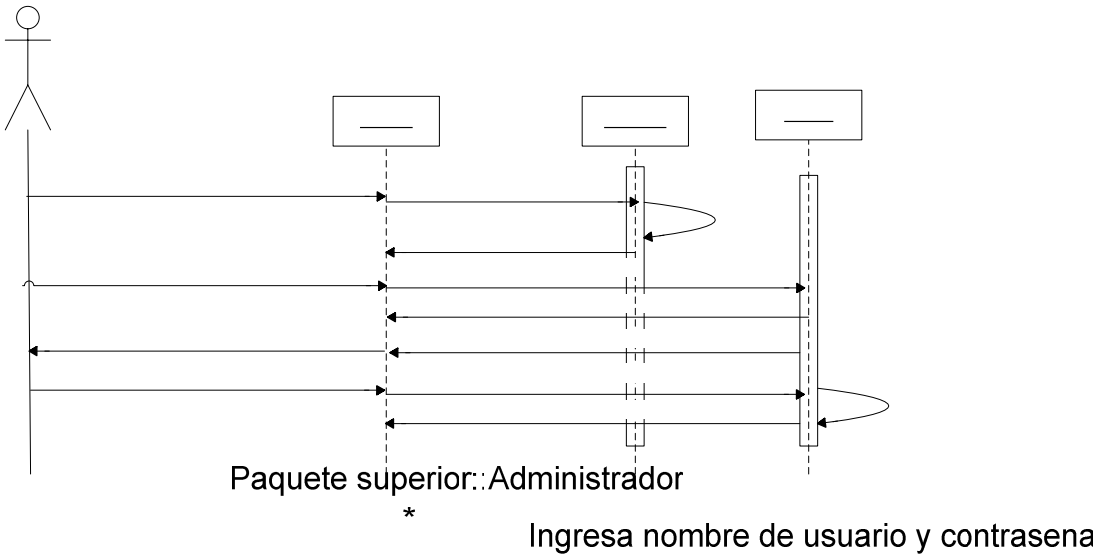


Figura 3.6. Diagrama de Secuencia Eliminación de Clientes.

3.1.7. Escenario para la especificación del tipo de mercado.

Ingresa nombre o codigo del Cliente

- El administrador del sistema ingresa nombre de usuario y contraseña.
- El administrador del sistema ingresa código o nombre del cliente para especificar el tipo de mercado.

Pide Confirmar Eliminacion

Confirma Eliminacion

- El sistema devuelve los datos del cliente.
- El administrador especifica el tipo de mercado.
- El sistema pide confirmación de los datos.
- El administrador del sistema confirma los datos.
- El sistema devuelve mensaje de datos guardados.

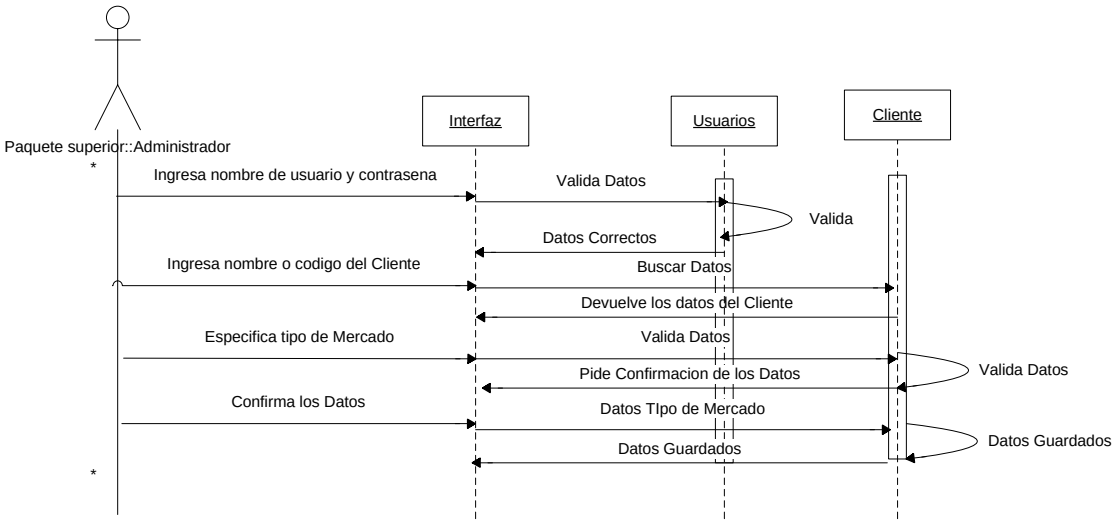


Figura 3.7. Diagrama de Secuencia Especificación del tipo de mercado.

3.1.8. Escenario para el ingreso de productos.

- El administrador del sistema ingresa su nombre de usuario y contraseña para poder ingresar al sistema.
- El administrador del sistema ingresa los datos del producto.
- El sistema pide la confirmación de los datos del producto.
- El administrador del sistema confirma los datos del producto.
- El sistema devuelve mensaje de datos guardados.

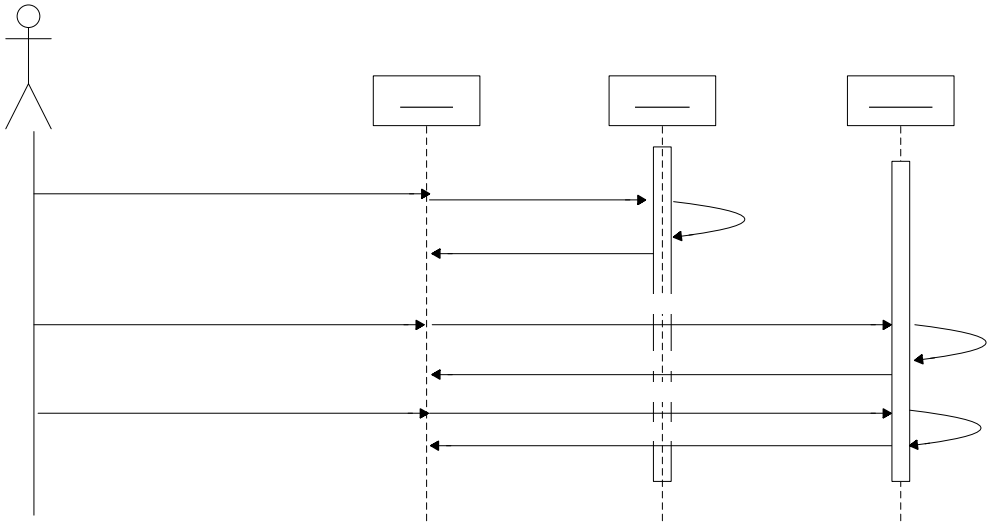


Figura 3.8. Diagrama de Secuencia Ingreso de Productos.

3.1.9. Escenario para la modificación de productos.

- El administrador del sistema ingresa nombre de usuario y contraseña
- El administrador del sistema ingresa el nombre o código del producto a modificar.
- El sistema devuelve los datos del producto a modificar.
- El administrador del sistema ingresa los nuevos datos del producto.
- El sistema pide confirmación de los datos.
- El administrador del sistema confirma los datos.
- El sistema devuelve mensaje de datos guardados.

Interfaz

Administrador

*

Ingresa nombre de usuario y contraseña

Ingresa los datos del Producto

Confirma los datos

*

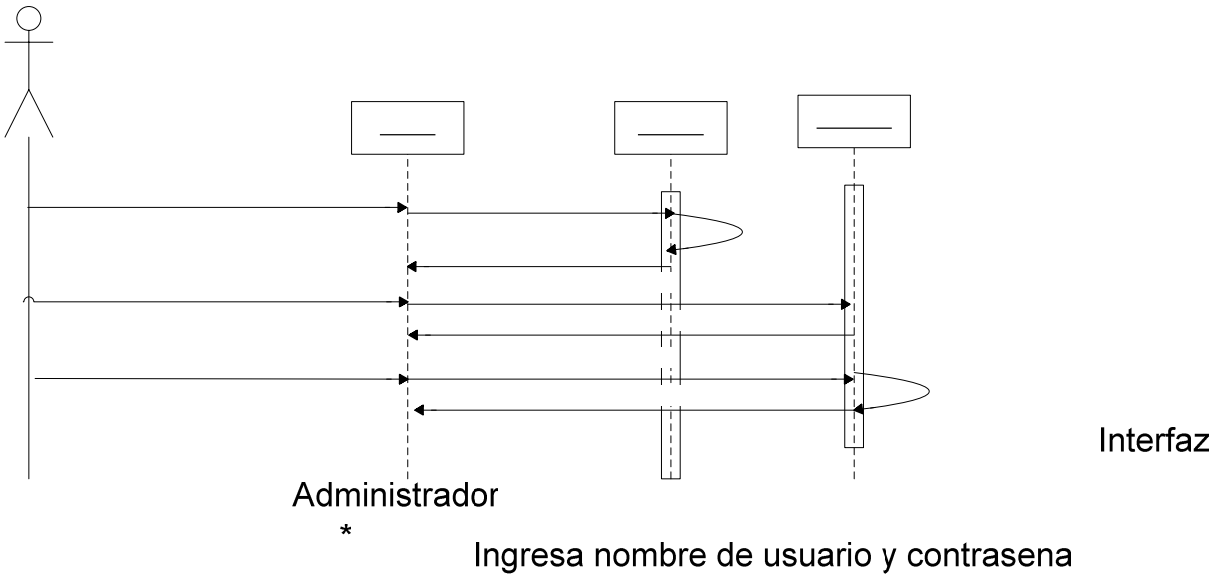


Figura 3.9. Diagrama de Secuencia Modificación de productos.

3.1.10. Escenario para la eliminación de productos.

Ingresa descripcion o codigo del Producto

- El administrador del sistema ingresa nombre de usuario y contraseña.
- El administrador del sistema ingresa el código o nombre del producto a ser eliminado.

Ingresa Nuevos Datos

- El sistema devuelve los datos del producto a ser eliminado.
- El sistema pide confirmación de eliminación de los datos del producto.
- El administrador del sistema confirma la eliminación de los datos del producto.
- El sistema devuelve mensaje de datos eliminados.

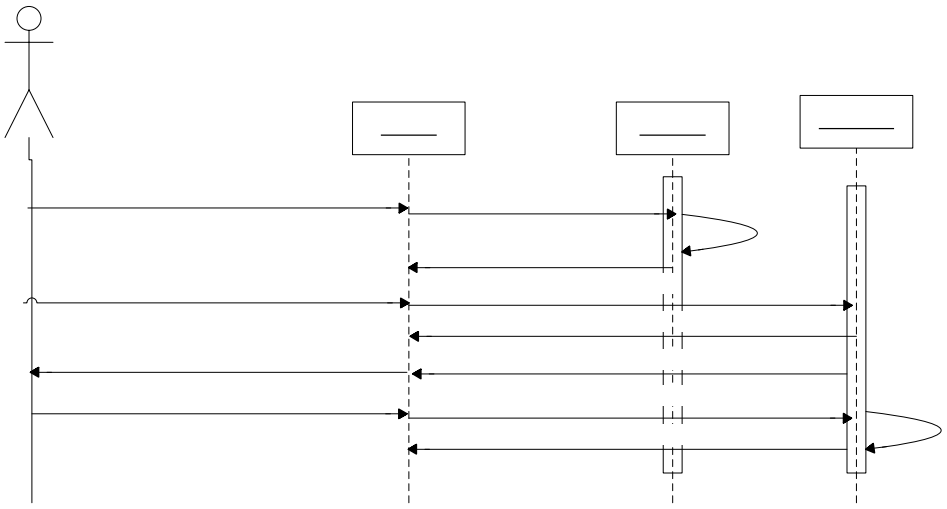


Figura 3.10. Diagrama de Secuencia Eliminación de productos.

3.1.11. Escenario para la autentificación de clientes.

- El cliente ingresa su nombre de usuario y contraseña.
- El sistema pide confirmación de los datos.
- El cliente confirma los datos.
- El sistema devuelve mensaje de datos guardados.

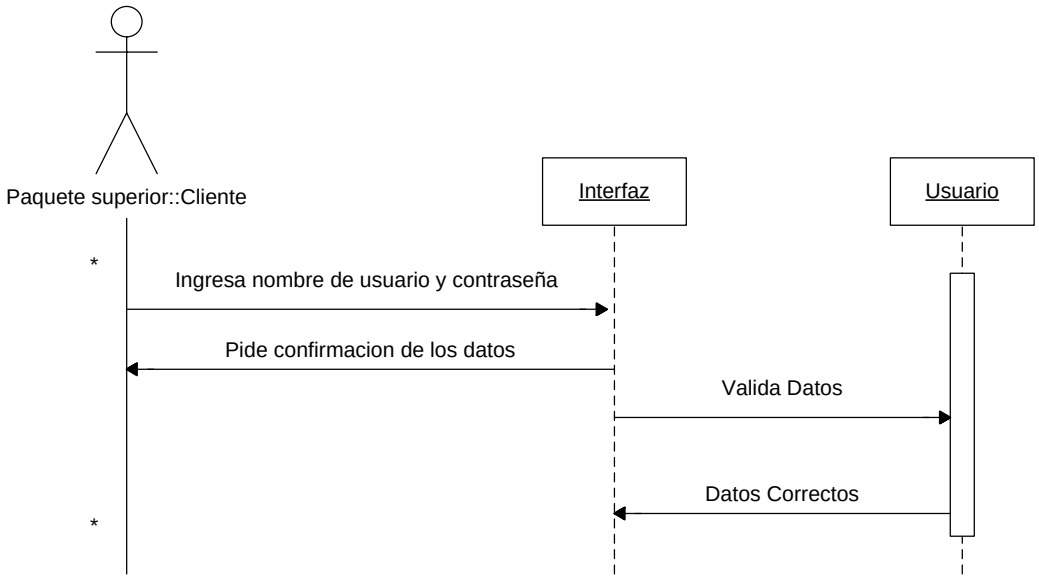


Figura 3.11. Diagrama de Secuencia Autenticación de Clientes.

3.1.12. Escenario para el registro de la disponibilidad del producto.

- El administrador del sistema ingresa su nombre de usuario y contraseña.
- El administrador del sistema ingresa el código o descripción del producto para registrar la disponibilidad.
- El sistema devuelve los datos del producto.
- El administrador del sistema ingresa la disponibilidad del producto.
- El sistema pide la confirmación de los datos.
- El administrador del sistema confirma los datos.
- El sistema devuelve mensaje de datos guardados.

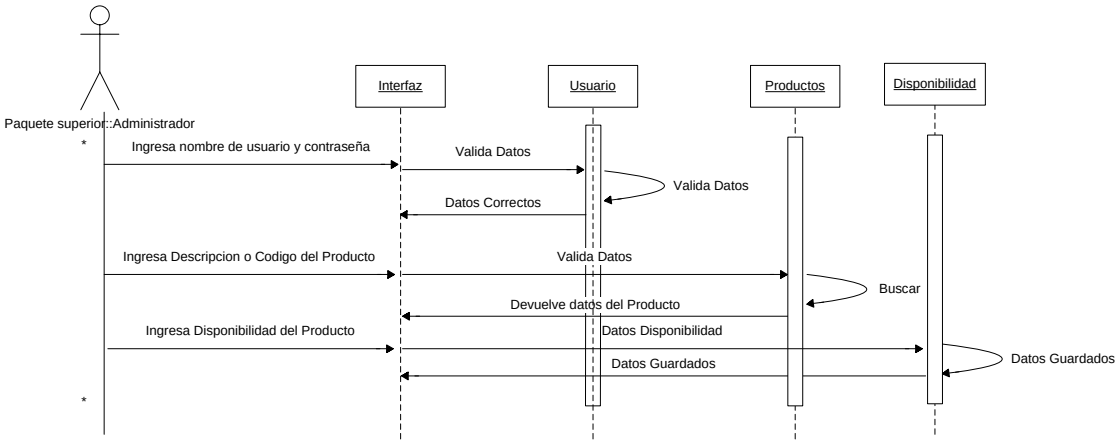


Figura 3.12. Diagrama de Secuencia Registro de la disponibilidad del producto.

3.1.13. Modificación de la disponibilidad del producto.

- El administrador del sistema ingresa su nombre de usuario y contraseña.
- El administrador del sistema ingresa el código o descripción del producto para modificar la disponibilidad.
- El sistema devuelve los datos del producto.
- El administrador del sistema ingresa los nuevos datos.
- El sistema pide la confirmación de los datos.
- El administrador del sistema confirma los datos.
- El sistema devuelve mensaje de datos guardados.

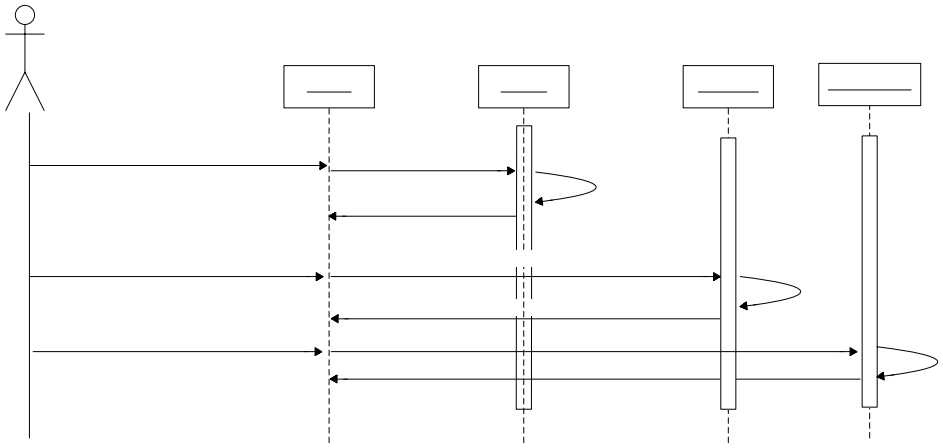


Figura 3.13. Diagrama de Secuencia Modificación de la disponibilidad del producto.

3.1.14. Escenario para la eliminación de la disponibilidad del producto.

- El administrador del sistema ingresa su nombre de usuario y contraseña.
- El administrador del sistema ingresa el código o descripción del producto para modificar la disponibilidad.
- El sistema devuelve los datos del producto.
- El sistema pide la confirmación de eliminación.
- El administrador del sistema confirma la eliminación.
- El sistema devuelve mensaje de datos eliminados.

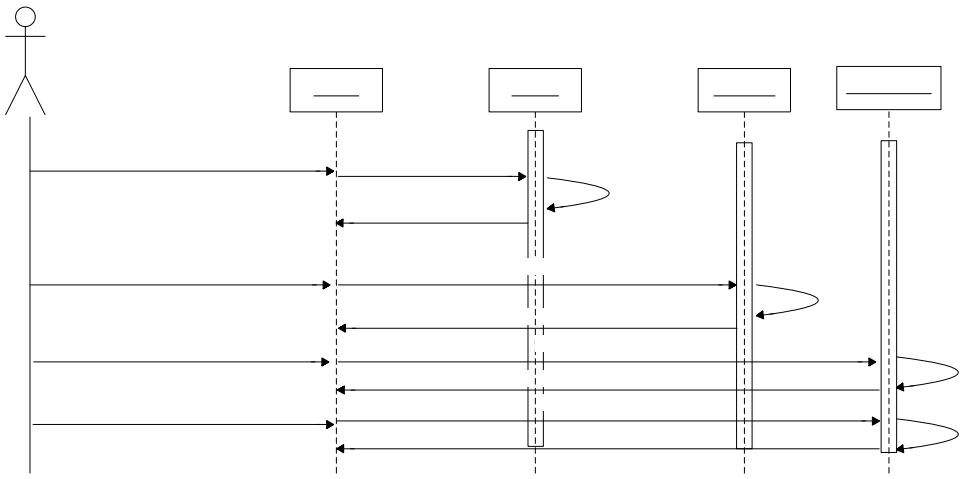


Figura 3.14. Diagrama de Secuencia Eliminación de la disponibilidad del producto.

3.1.15. Escenario para la colocación de pedidos.

- El cliente ingresa su nombre de usuario y contraseña.
- El cliente escoge los productos a comprar.
- El cliente ingresa las cantidades de los productos a comprar.
- El cliente escoge el producto el cual desea modificar los datos en el pedido.
- El sistema devuelve los datos del producto.
- El cliente ingresa los nuevos datos del pedido.
- El cliente ingresa información adicional del pedido.
- El sistema pide confirmación de los datos.
- El cliente confirma los datos.
- El sistema devuelve mensaje de datos guardados.

Interfaz

Paquete superior::Administrador

Ingresa nombre de usuario y contraseña

Va

Dat

Ingresa Descripcion o Codigo del Producto

Pide Eliminacion de Disponibilidad

- El agente vendedor ingresa su nombre de usuario y contraseña.
- El agente vendedor ingresa el número de pedido del cual desea verificar los datos.
- El sistema devuelve los datos del pedido.
- El agente vendedor confirma las cantidades y el precio de los productos del pedido.
- El agente vendedor ingresa información adicional del pedido.
- El sistema pide la confirmación de los datos.
- El agente vendedor confirma los datos.
- El sistema devuelve mensaje de datos guardados.

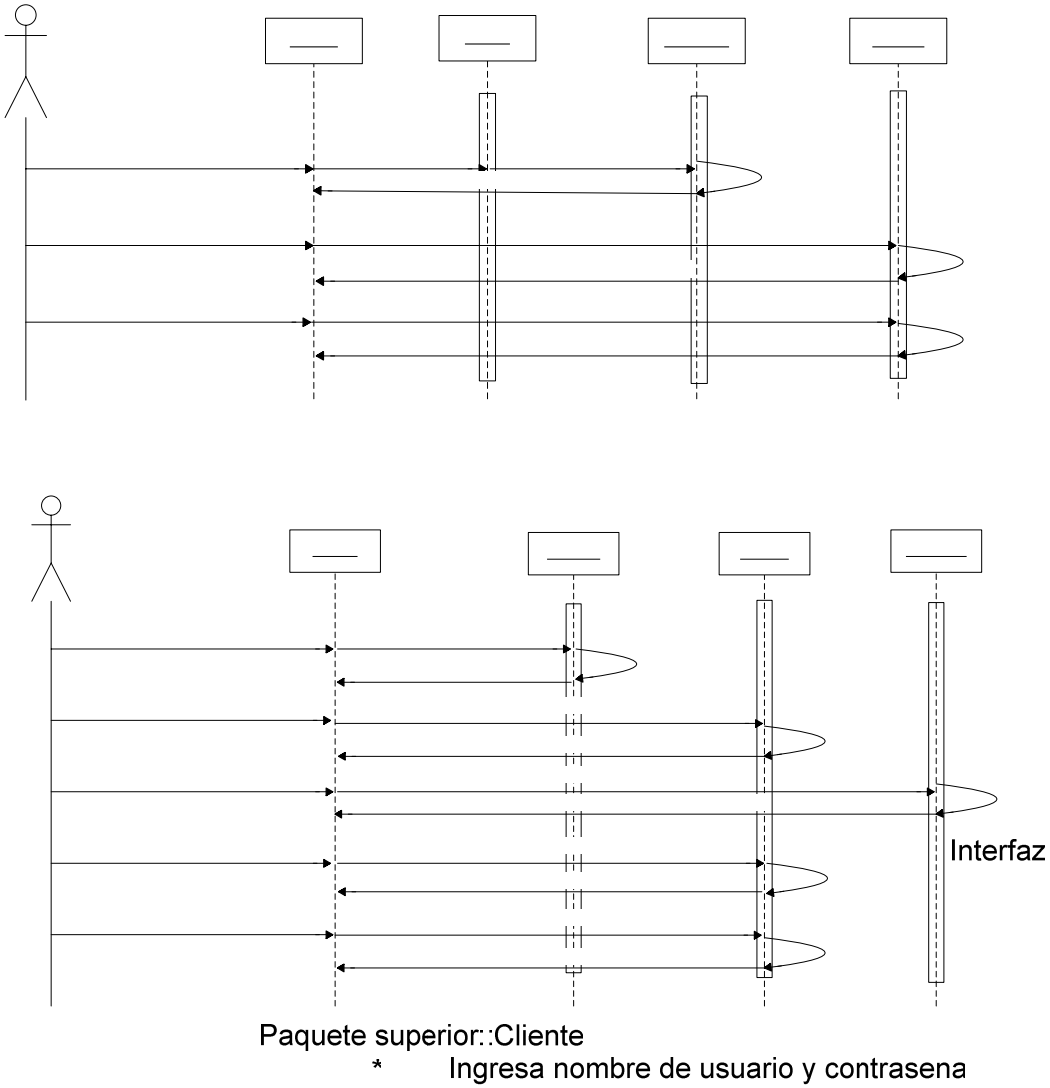


Figura 3.15. Diagrama de Secuencia Colocación de pedidos.

3.1.16. Escenario para la consulta del Estado de Cuenta de los Clientes (Administrador del Sistema).

- El administrador del sistema ingresa su nombre de usuario y contraseña.
- El administrador del sistema ingresa el código o nombre del cliente del cual se desea consultar el estado de cuenta.
- El sistema devuelve los datos del Estado de Cuenta del Cliente.

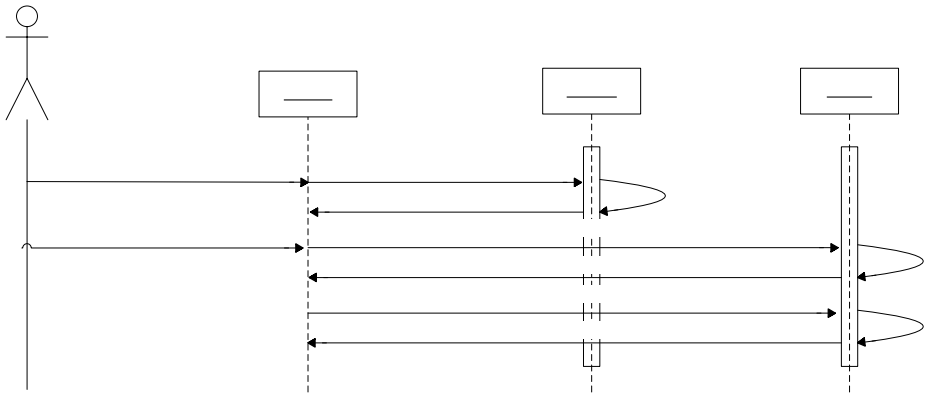


Figura 3.16. Diagrama de Secuencia Consulta del estado de cuenta de los clientes (Administrador).

3.1.17. Escenario para la consulta del Estado de Cuenta de los Clientes (Cliente).

- El cliente ingresa su nombre de usuario y contraseña.
- El sistema devuelve los datos de su Estado de Cuenta.

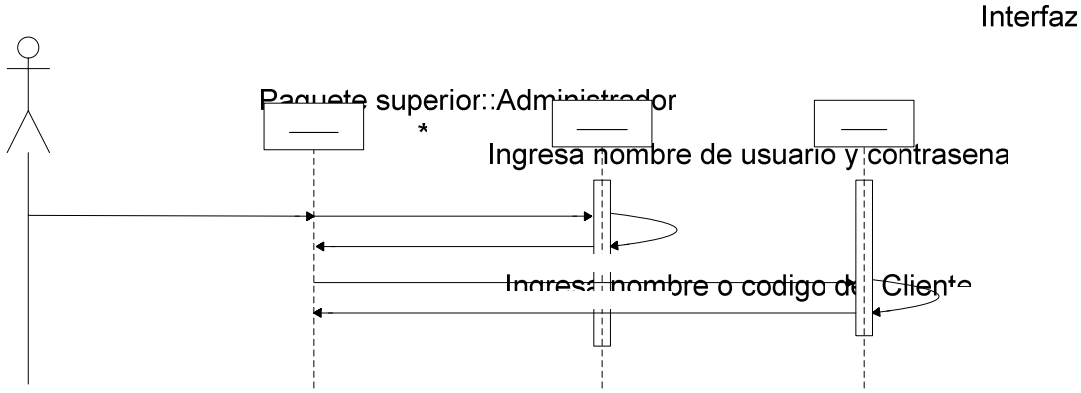


Figura 3.17. Diagrama de Secuencia Consulta del estado de cuenta de los clientes (Cliente).

*

3.1.18. Escenario para el ingreso del perfil de cliente.

- El administrador del sistema ingresa su nombre de usuario y contraseña.
- El administrador del sistema ingresa el código o nombre del cliente del cual se desea ingresar el perfil.
- El sistema devuelve la información del cliente.
- El administrador del sistema ingresa la información del perfil del cliente.
- El sistema pide la confirmación de los datos.
- El administrador del sistema confirma los datos.
- El sistema devuelve mensaje de datos guardados.

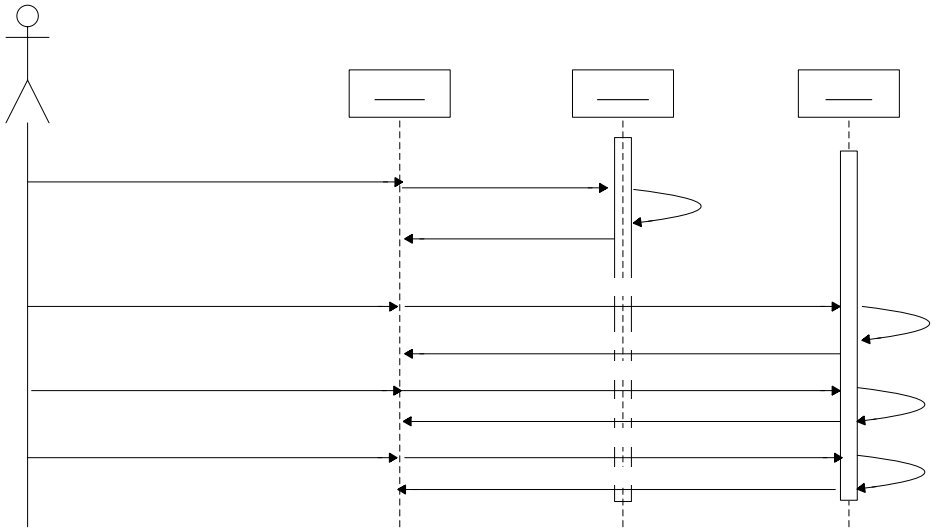


Figura 3.18. Diagrama de Secuencia Ingreso del Perfil del cliente.

3.1.19. Escenario para la modificación del Perfil de Cliente (Administrador del Sistema).

- El administrador del sistema ingresa su nombre de usuario y contraseña.
- El administrador del sistema ingresa el código o nombre del cliente del cual desea modificar el perfil.
- El sistema devuelve la información del perfil del cliente.
- El administrador del sistema ingresa los nuevos datos del perfil del cliente.
- El sistema pide la confirmación de los datos.
- El administrador del sistema confirma los datos.
- El sistema devuelve mensaje de datos guardados.

Inte

Paquete superior::Administrador

Ingresa nombre de usuario y contraseña

Ingresa los datos del Cliente

Ingresa datos de Perfil de Cliente

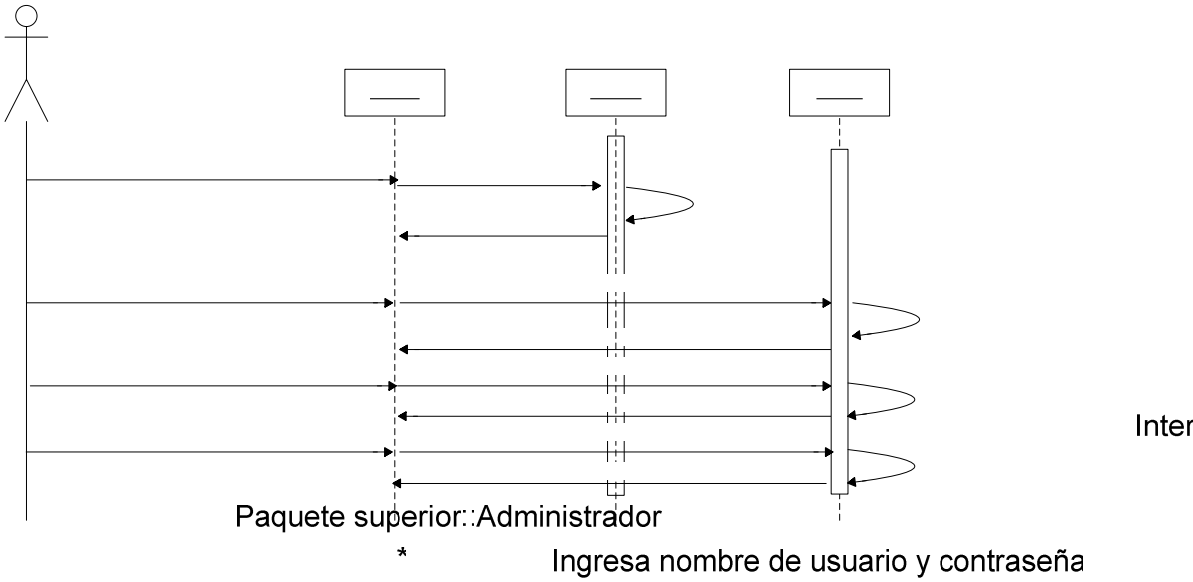


Figura 3.19. Diagrama de Secuencia Modificación del Perfil del cliente (Administrador).

3.1.20. Escenario para la modificación del Perfil del Cliente (Clientes).

- El cliente ingresa su nombre de usuario y contraseña.
- El sistema devuelve el idioma del perfil del cliente.
- El cliente ingresa el nuevo idioma del perfil.
- El sistema pide confirmación de los datos.
- El cliente confirma los datos.
- El sistema devuelve mensaje de datos guardados.

Ingresa los datos del Cliente

Ingresa nuevos datos de Perfil de Cliente

Confirma los Datos

*

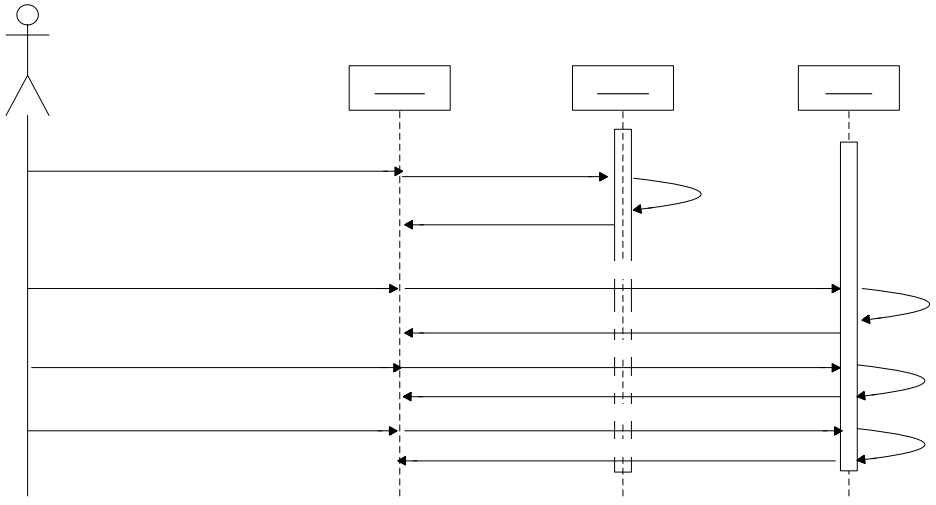


Figura 3.20. Diagrama de Secuencia Modificación del Perfil del cliente (Cliente).

3.2.Diagrama de Despliegue.

Nuestra plataforma de despliegue consta de un servidor central, corriendo ya sea Windows Server 2003 o en su defecto Windows XP, dicho servidor prestará los siguientes servicios:

- Base de datos SQL Server
- Servidor Web Internet Information Services
- Aplicación Web E-Commerce(nodo)

En este caso el nodo a instalarse será distribuido en el equipo que presta los servicios de Internet Information Services, junto con la base de datos.

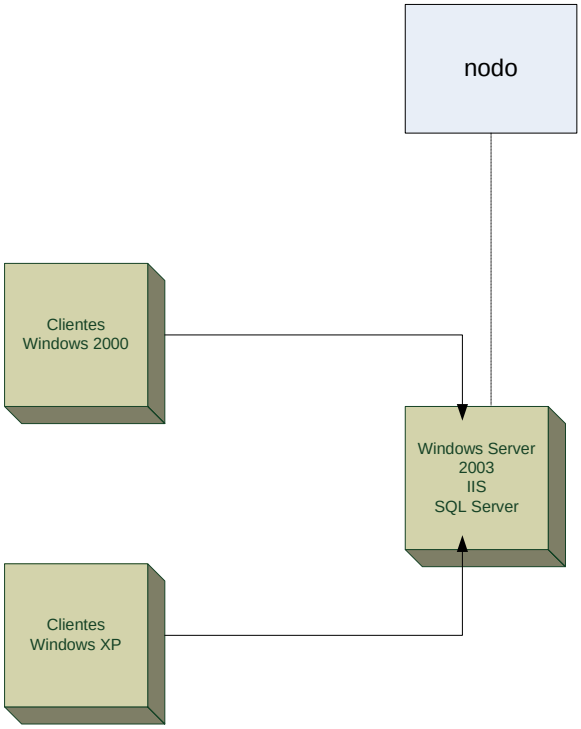


Figura 3.21. Diagrama de Despliegue de la Aplicación E-Commerce.

3.3.Diagrama de Clases.

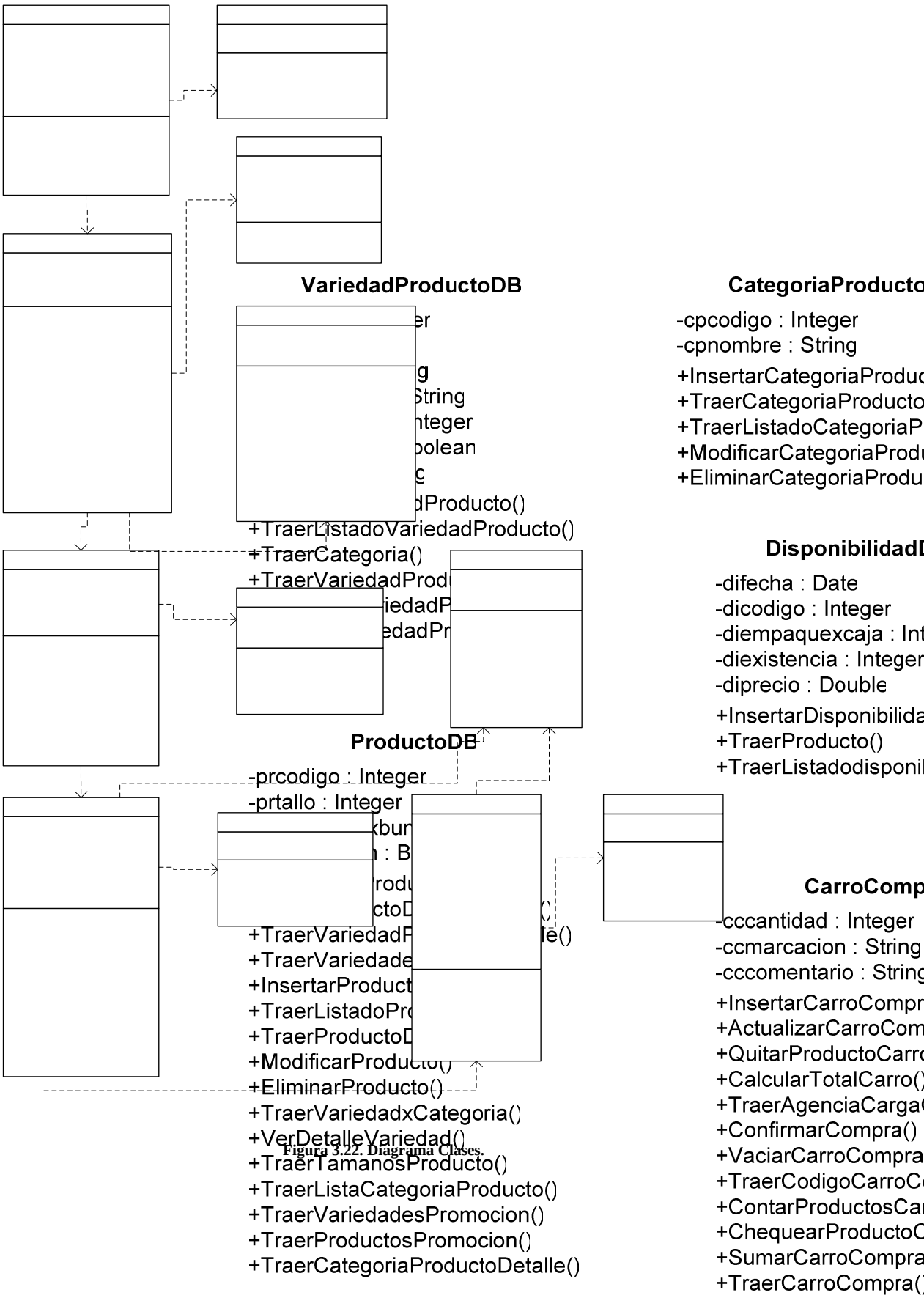


Figura 3.22. Diagrama Clases.

3.4. Modelo de Datos.

3.4.1. Modelo Entidad – Relación (MER).

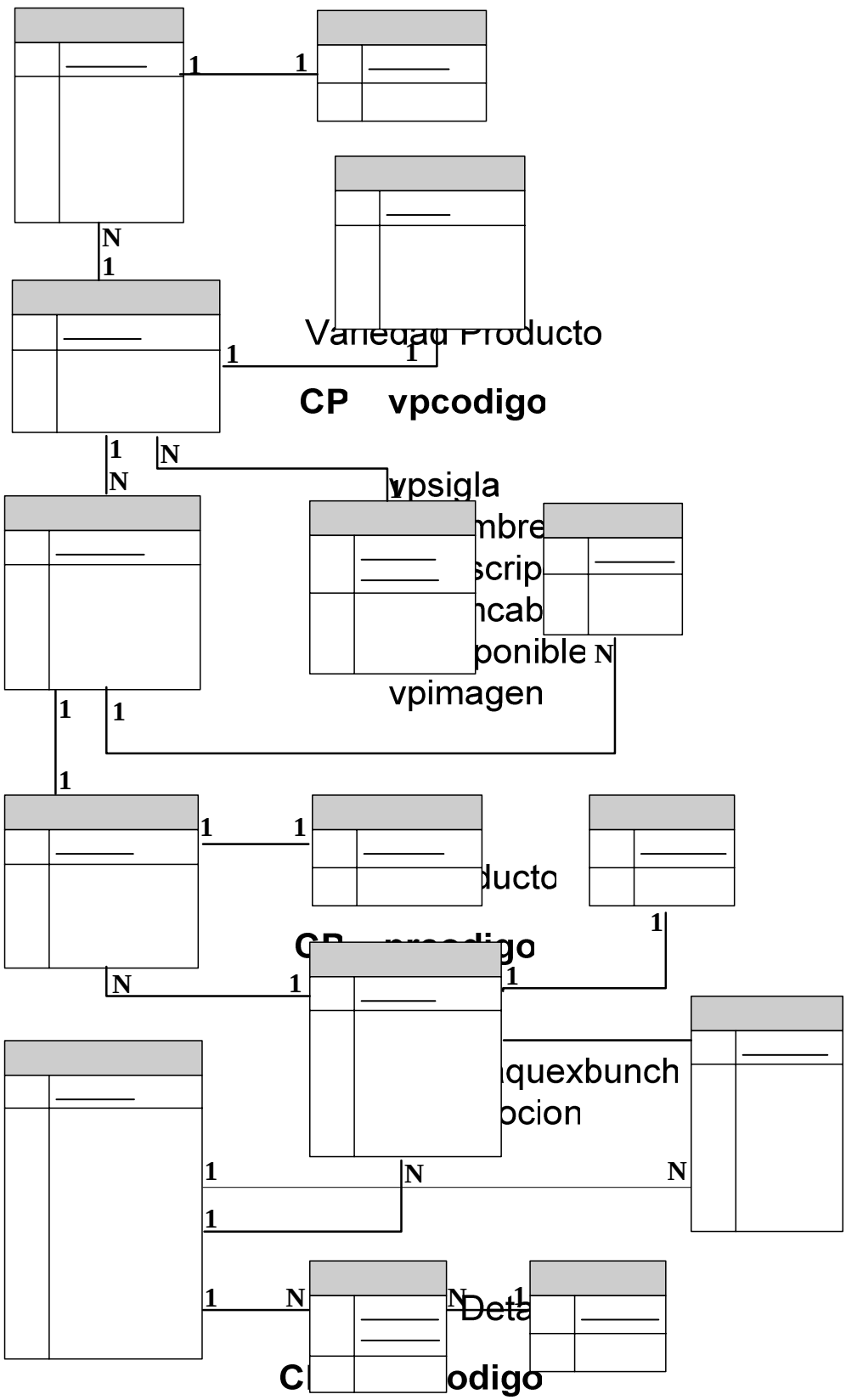


Figura 3.23. Modelo Entidad Relación (MER).

ordcantidad
ordmarcacion
ordnroguiamadre
ordnroguiahija

3.4.2. Diccionario de Datos.

DICCIONARIO DE DATOS.
COMERCIO ELECTONICO E-COMMERCE PLANTACIONES EL TREBOL.

ENTIDAD	ATRIBUTO	DESCRIPCION	TIPO	TAMAÑO	L.P.	L.F.	RELACION
AGENCIACARGA	ACCODIGO	CODIGO DE LA AGENCIA DE CARGA	INTEGER	4	√		
AGENCIACARGA	ACRUC	RUC DE LA AGENCIA DE CARGA	VARCHAR	13			
AGENCIACARGA	ACNOMBRE	NOMBRE DE LA AGENCIA DE CARGA	VARCHAR	40			
AGENCIACARGADIRECCION	ACDCODIGO	CODIGO DE LA DIRECCION DE LA AGENCIA DE CARGA	INTEGER	4	√		
AGENCIACARGADIRECCION	ACCODIGO	CODIGO DE LA AGENCIA DE CARGA	INTEGER	4		√	AGENCIACARGA
AGENCIACARGADIRECCION	ACDCIUDAD	CIUDAD EN LA QUE SE ENCUENTRA LA AGENCIA DE CARGA	VARCHAR	40			
AGENCIACARGADIRECCION	ACDDIRECCION	DIRECCION EN LA QUE SE ENCUENTRA LA AGENCIA DE CARGA	VARCHAR	255			
AGENCIACARGADIRECCION	ACDTELEFONO1	TELEFONO DE LA AGENCIA DE CARGA	VARCHAR	15			
AGENCIACARGADIRECCION	ACDTELEFONO2	TELEFONO DE LA AGENCIA DE CARGA	VARCHAR	15			
AGENCIACARGADIRECCION	ACDCONTACTO	PERSONA ENCARGADA DE LA AGENCIA DE CARGA	VARCHAR	80			
CARROCOMPRA	CLCODIGO	CODIGO DEL CLIENTE	INTEGER	4	√	√	CLIENTE
CARROCOMPRA	PRCODIGO	CODIGO DEL PRODUCTO	INTEGER	4	√	√	PRODUCTO
CARROCOMPRA	CCCANTIDAD	CANTIDAD A COMPRAR	SMALLINT	2			
CARROCOMPRA	CCMARCACION	MARCACION DE LAS CAJAS	VARCHAR	50			
CARROCOMPRA	ACCODIGO	CODIGO DE LA AGENCIA DE CARGA	INTEGER	4		√	AGENCIACARGA
CARROCOMPRA	CCCOMENTARIO	COMENTARIO DE LA COMPRA	VARCHAR	100			
CATEGORIAPRODUCTO	CPCODIGO	CODIGO DE LA CATEGORIA DEL PRODUCTO	INTEGER	4	√		
CATEGORIAPRODUCTO	CPNOMBRE	NOMBRE DE LA CATEGORIA DEL PRODUCTO	VARCHAR	40			
CLIENTE	CLCODIGO	CODIGO DEL CLIENTE	INTEGER	4	√		
CLIENTE	CLNOMBRE	NOMBRE DEL CLIENTE	VARCHAR	50			
CLIENTE	CLNOMBREUSUARIO	NOMBRE DE USUARIO DEL SISTEMA PARA EL CLIENTE	VARCHAR	30			

CLIENTE	CLEMAIL	DIRECCION DE CORREO ELECTRONICO DEL CLIENTE	VARCHAR	100			
CLIENTE	CLTAMDISPOMIN	DISPONIBILIDAD MINIMA DE PRODUCTOS PARA EL CLIENTE	SMALLINT	2			
CLIENTE	CLTAMDISPOMAX	DISPONIBILIDAD MAXIMA DE PRODUCTOS PARA EL CLIENTE	SMALLINT	2			
CLIENTE	MCCODIGO	CODIGO DEL MERCADO AL CUAL PERTENECE EL CLIENTE	INTEGER	4		√	MERCADOCLIENTE
CLIENTE	USCEDULA	CODIGO DEL AGENTE VENDEDOR ASIGNADO AL CLIENTE	VARCHAR	10			
CLIENTE	CLCONTRASEÑA	CONTRASEÑA DEL CLIENTE	VARCHAR	255			
DISPONIBILIDAD	DIFECHA	FECHA DE LA DISPONIBILIDAD DE LOS PRODUCTOS	DATETIME	8	√		
DISPONIBILIDAD	PRCODIGO	CODIGO DEL PRODUCTO	INTEGER	4	√	√	PRODUCTO
DISPONIBILIDAD	DIEMPAQUEXBUNCH		SMALLINT	2			
DISPONIBILIDAD	DIEXISTENCIA	EXISTENCIA DEL PRODUCTO EN LAS FINCAS	SMALLINT	2			
DISPONIBILIDAD	DIPRECIO	PRECIO DE LOS PRODUCTOS	MONEY	8			
ESTADOORDEN	EOCODIGO	CODIGO DEL ESTADO DE LA ORDEN	INTEGER	4	√		
ESTADOORDEN	EODESCRIPCION	DESCRIPCION DEL ESTADO DE LA ORDEN	VARCHAR	40			
MERCADOCLIENTE	MCCODIGO	CODIGO DEL MERCADO DEL CLIENTE	INTEGER	4	√		
MERCADOCLIENTE	MCNOMBRE	NOMBRE DEL MERCADO DEL CLIENTE	VARCHAR	40			
ORDEN	ORCODIGO	CODIGO DE LA ORDEN DE VENTA	INTEGER	4	√		
ORDEN	CLCODIGO	CODIGO DEL CLIENTE	INTEGER	4		√	CLIENTE
ORDEN	ORFECHA	FECHA EN LA QUE SE REALIZA LA VENTA	DATETIME	8			
ORDEN	ORFECHASALIDAFINCA	FECHA EN LA QUE SALE LOS PRODUCTOS DE LA FINCA	DATETIME	8			
ORDEN	ORFECHAVUELO	FECHA EN LA QUE LOS PRODUCTOS SON ENVIADOS	DATETIME	8			
ORDEN	EOCODIGO	CODIGO DEL ESTADO DE LA ORDEN	INTEGER	4			
ORDEN	OROBSERVACIONES	OBSERVACIONES REFERENTES A LA VENTA	VARCAR	100			
ORDENDETALLE	ORCODIGO	CODIGO DE LA ORDEN DE VENTA	INTEGER	4	√		
ORDENDETALLE	PRCODIGO	CODIGO DEL PRODUCTO	INTEGER	4	√	√	PRODUCTO
ORDENDETALLE	ORDCANTIDAD	CANTIDAD HA SER VENDIDA	SMALLINT	2			
ORDENDETALLE	ORDMARCACION	MARCACION DE LAS CAJAS DE PRODUCTOS	VARCHAR	50			
ORDENDETALLE	ACCODIGO	CODIGO DE LA AGENCIA DE CARGA	INTEGER	4	√		AGENCIACARGA
ORDENDETALLE	ORDNROGUIAMADRE	NUMERO DE GUIA	VARCHAR	40			
ORDENDETALLE	ORDNROGUIAHIJA	NUMERO DE GUIA	VARCHAR	40			

ORDENDETALLE	ORDCOMENTARIO	COMENTARIO DE LA ORDEN DE VENTA	VARCHAR	100			
PRODUCTO	PRCODIGO	CODIGO DEL PRODUCTO	INTEGER	4	√		
PRODUCTO	PRTALLO	TAMAÑO DEL TALLO DEL PRODUCTO	INTEGER	4			
PRODUCTO	PREMPAQUEXBUNCH		INTEGER	4			
PRODUCTO	VPCODIGO	CODIGO DE LA VARIEDAD A LA CUAL PERTENECE EL PRODUCTO	INTEGER	4		√	VARIEDADPRODUCTO
PRODUCTO	PRPROMOCION	IDENTIFICA SI EL PRODUCTO ESTA O NO EN PROMOCION	BIT	1			
USUARIO	USCEDULA	CEDULA DEL USUARIO DEL SISTEMA	VARCHAR	10	√		
USUARIO	USNOMBRE	NOMBRES DEL USUARIO	VARCHAR	40			
USUARIO	USAPELLIDO	APELLIDOS DEL USUARIO	VARCHAR	40			
USUARIO	USNOMBREUSARIO	NOMBRE DE USUARIO PARA EL INGRESO AL SISTEMA	VARCHAR	40			
USUARIO	USEMAIL	EMAIL DEL USUARIO DEL SISTEMA	VARCHAR	100			
USUARIO	USTELEFONO	TELEFONO DEL USUARIO DEL SISTEMA	VARCHAR	15			
USUARIO	USEXTENSION	EXTENSION DEL TELEFONO DEL USUARIO DEL SISTEMA	VARCHAR	5			
USUARIO	USTELMOVIL	TELEFONO MOVIL DEL USUARIO DEL SISTEMA	VARCHAR	15			
USUARIO	USFOTO	FOTO DEL USUARIO DEL SISTEMA	VARCHAR	255			
USUARIO	USVENDEDOR	IDENTIFICA SI EL USUARIO DEL SISTEMA ES O NO VENDEDOR	BIT	1			
USUARIO	USCONTRASEÑA	CONTRASEÑA DEL USUARIO DEL SISTEMA	VARCHAR	255			
USUARIO	USACTIVO	IDENTIFICA SI EL USUARIO DEL SISTEMA ESTA O NO ACTIVO	BIT	1			
VARIEDADPRODUCTO	VPCODIGO	CODIGO DE LA VARIEDAD DEL PRODUCTO	INTEGER	4	√		
VARIEDADPRODUCTO	VPSIGLA	SIGLAS DE LA VARIEDAD DEL PRODUCTO	VARCHAR	10			
VARIEDADPRODUCTO	VPNOMBRE	NOMBRE DE LA VARIEDAD DEL PRODUCTO	VARCHAR	50			
VARIEDADPRODUCTO	VPDESCRIPCION	DESCRIPCION DE LA VARIEDAD DEL PRODUCTO	VARCHAR	255			
VARIEDADPRODUCTO	VPTAMCABEZA	TAMAÑO DE LA CABEZA DE LA VARIEDAD DEL PRODUCTO	SMALLINT	2			
VARIEDADPRODUCTO	CPCODIGO	CODIGO DE LA CATEGORIA DEL PRODUCTO	INTEGER	4		√	CATEGORIAPRODUCTO
VARIEDADPRODUCTO	VPDISPONIBLE	IDENTIFICA SI LA VARIEDAD DEL PRODUCTO ESTA O NO DISPONIBLE	BIT	1			
VARIEDADPRODUCTO	VPIMAGEN	IMAGEN DE LA VARIEDAD DEL PRODUCTO	VARCHAR	255			
ROL	ROCODIGO	CODIGO DEL ROL	INTEGER	4	√		
ROL	RONOMBRE	NOMBRE DEL ROL	VARCHAR	100			
ROLUSUARIO	USCEDULA	CEDULA DEL USUARIO DEL SISTEMA	VARCHAR	10		√	USUARIO

ROLUSUARIO	ROCODIGO	CODIGO DEL ROL	INTEGER	4		√	ROL
MENSAJE	MECODIGO	CODIGO DEL MENSAJE	INTEGER	4	√		
MENSAJE	USCEDULA	CEDULA DEL USUARIO DEL SISTEMA	VARCHAR	10		√	USUARIO
MENSAJE	CLCODIGO	CODIGO DEL CLIENTE	INTEGER	4		√	CLIENTE
MENSAJE	MEASUNTO	ASUNTO POR EL CUAL EL MENSAJE ES ENVIADO	VARCHAR	255			
MENSAJE	MEMENSAJE	TEXTO DE REDACCION DEL MENSAJE	VARCHAR	5000			
MENSAJE	MELEIDO	IDENTIFICA SI EL MENSAJE HA SIDO LEIDO O NO	BIT	1			
MENSAJE	MEPARA	IDENTIFICA EL DESTINATARIO DEL MENSAJE	BIT	1			
MENSAJE	MEFECHA	FECHA EN LA QUE SE HA ENVIADO EL MENSAJE	DATETIME	8			

Tabla 3.1. Diccionario de Datos

3.4.3. Esquema Físico de la Base de Datos.

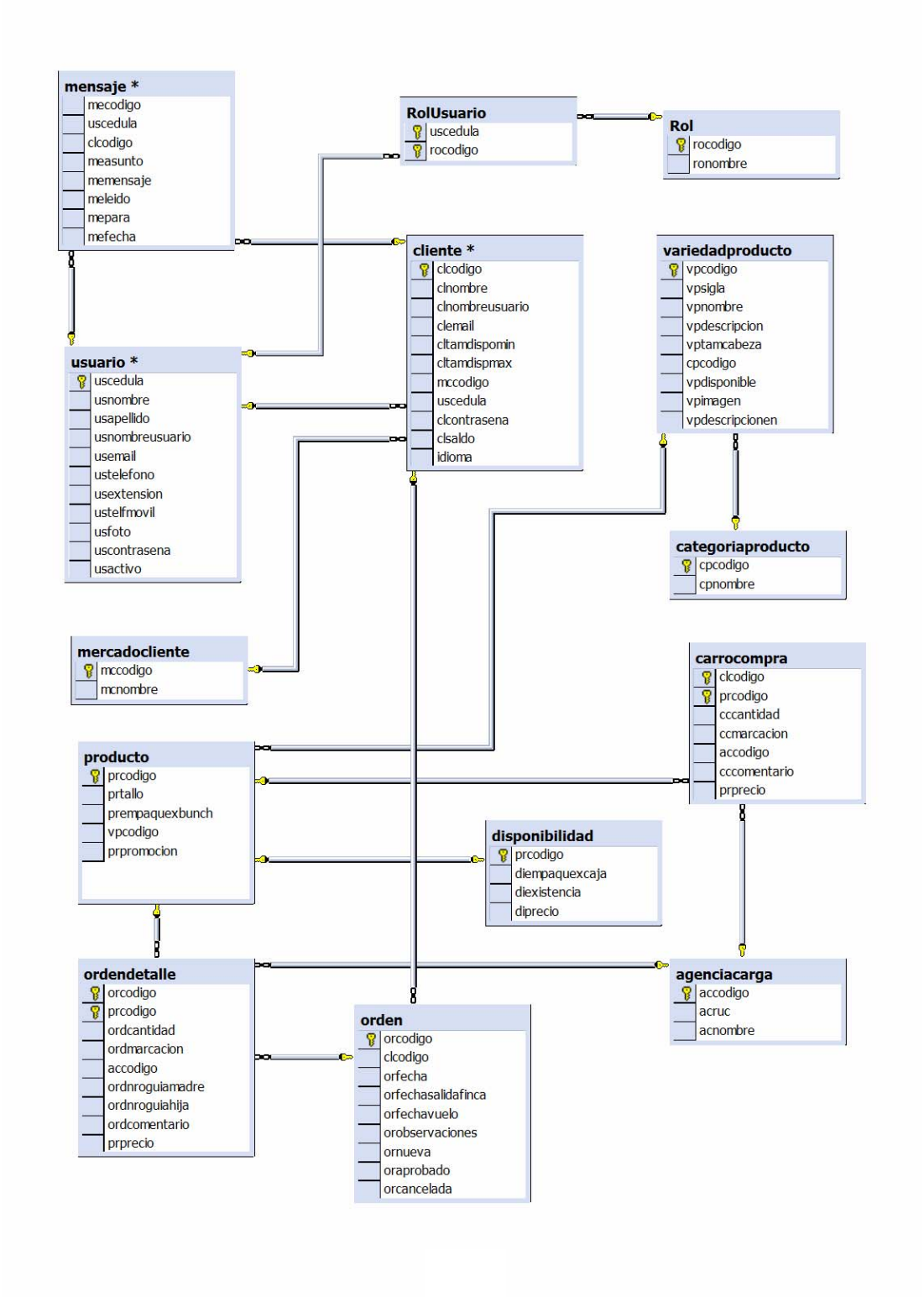


Figura 3.24. Esquema Físico de la Base de Datos.

Capítulo 4
Desarrollo de la Aplicación

En este capítulo se presentan algunos modelos lingüísticos, a los que se han denominado "Estándares de Programación", y que se han utilizado para la construcción de E-Commerce.

Se presentan también las definiciones de las clases, métodos y atributos utilizados para la codificación del sistema mediante Microsoft Visual Studio .NET.

4.1. Estándares de programación.

Para el desarrollo de la aplicación se utilizaron los siguientes modelos lingüísticos, manteniendo siempre el carácter nemotécnico de forma que el código y las expresiones resultantes sean fáciles de comprender:

- Nomenclatura de Clases: Nombre de la Clase junto más el sufijo DB.

  Ejemplo: AgenciaCargaDB

- Nomenclatura de Parámetros: Se utilizo el prefijo prm_ seguido del nombre del parámetro.

  Ejemplo: prm_ruc.

- Nomenclatura de Tablas: El nombre de cada una de las tablas esta definido de acuerdo a la información que almacena.

  Ejemplo: Tabla Cliente, almacena información de los clientes de la empresa.

- Nomenclatura de los campos de las Tablas: Se utilizaron las dos primeras letras del nombre de la tabla seguido del identificador del campo de acuerdo a la información que almacene.

  Ejemplo: Tabla Usuario, campo uscedula.

- Nomenclatura de Formularios: Se utilizo el prefijo frm seguido del nombre del formulario.

  Ejemplo: frmAdmin.aspx

- Nomenclatura de Controles de Usuario: Se utilizo el prefijo Ctrl seguido del nombre del control de usuario.

Ejemplo: CtrlClienteHistorial.ascx

- Nomenclatura de Cuadros de Texto: Se utilizó el prefijo txt_ seguido del nombre del cuadro de texto.

  Ejemplo: txt_cedula.

- Nomenclatura de Listas Desplegables: Se utilizó el prefijo cmb_ seguido del nombre de la lista desplegable.

  Ejemplo: cmb_clientes.

- Nomenclatura de Etiquetas: Se utilizó el prefijo lbl_ seguido del nombre de la etiqueta.

  Ejemplo: lbl_cedula.

- Nomenclatura de Cuadriculas: Se utilizó el prefijo grd_ seguido del nombre de la cuadricula.

  Ejemplo: grd_clientes.

- Nomenclatura de Imágenes: Se utilizó el prefijo img_ seguido del nombre de la imagen.
  Ejemplo: img_rosa.

- Nomenclatura de Botones: Se utilizó el prefijo cmd_ seguido del nombre del botón.
  Ejemplo: txt_cedula.

4.2 Creación de las definiciones de clase a partir de los diagramas de clases del diseño.

4.2.1 Definición de las clases con métodos y atributos.

Agencias de Carga.

```
Public Class AgenciaCargaDB
    Public ruc As String
    Public nombre As String
    Public Function InsertarAgenciaCarga(ByVal ruc As String, ByVal
    nombre As String) As Integer
    End Function

    Public Function TraerListadoAgenciaCarga() As DataSet
    End Function

    Public Function TraerAgenciaCargaDetalle(ByVal codigo As String)
    As AgenciaCargaDetalle
    End Function

    Public Function ModificarAgenciaCarga(ByVal codigo As Integer,
    ByVal ruc As String, ByVal nombre As String) As Integer
End Function

    Public Function EliminarAgenciaCarga(ByVal codigo As Integer) As
    Integer
End Function
End Class
```

Clientes.

```
Public Class ClienteDB
    Public nombre As String
    Public nombreusuario As String
    Public contrasena As String
    Public email As String
```

```
Public tamdispomin As Integer
Public tamdispomax As Integer
Public mercado As Integer
Public agente As String

Public Function TraerMercadoCliente() As SqlDataReader
End Function

Public Function TraerAgenteVendedor() As SqlDataReader
End Function

Public Function InsertarCliente(ByVal nombre As String, ByVal
nombreusuario As String, ByVal contrasena As String, ByVal email
As String, ByVal dispomin As Double, ByVal dispomax As Double,
ByVal mercado As Integer, ByVal agente As String) As Integer
End Function
Public Function TraerListadoCliente() As DataSet
End Function

Public Function TraerClienteDetalle(ByVal codigo As Integer) As
ClienteDetalle
End Function

Public Function ModificarCliente(ByVal codigo As Integer, ByVal
nombre As String, ByVal nombreusuario As String, ByVal contrasena
As String, ByVal email As String, ByVal dispomin As Double, ByVal
dispomax As Double, ByVal mercado As Integer, ByVal agente As
String) As Integer
End Function

Public Function EliminarCliente(ByVal codigo As Integer) As Integer
End Function
```

```
        Public Function LoginCliente(ByVal NombreUsuario As String,
        ByVal Contrasena As String) As Integer
        End Function

        Public Function TraerHistorialCliente(ByVal clcodigo As Integer) As
        DataSet
        End Function
    End Class
```

Carro Compra.

```
    Public Class CarroCompraDB
        Public Function TraerCarroCompra(ByVal clcodigo As Integer) As
        DataSet
        End Function

        Public Function CalcularTotalCarro(ByVal clcodigo As Integer) As
        Single
        End Function

        Public Function InsertarCarroCompra(ByVal cliente As Integer,
        ByVal producto As Integer, ByVal cantidad As Integer, ByVal
        accodigo As Integer) As Integer
        End Function
        Public Function ActualizarCarroCompra(ByVal cliente As Integer,
        ByVal producto As Integer, ByVal cantidad As Integer, ByVal
        marcacion As String, ByVal agencia As Integer, ByVal comentario As
        String) As Integer
        End Function

        Public Function TraerAgenciaCargaCarroCompra(ByVal clcodigo As
        Integer, ByVal prcodigo As Integer) As Integer
        End Function
```

```
        Public Function ConfirmarCompra(ByVal clcodigo As String) As
        Integer
        End Function

        Public Function VaciarCarroCompra(ByVal clcodigo As Integer) As
        Integer
        End Function

        Public Function TraerCodigoCarroCompra()
        End Function

        Public Function ContarProductosCarro(ByVal clcodigo As Integer)
        As Integer
        End Function

        Public Function QuitarProductoCarro(ByVal clcodigo As Integer,
        ByVal prcodigo As Integer) As Integer
        End Function

        Public Function ChequearProductoCarro(ByVal clcodigo As Integer,
        ByVal prcodigo As Integer) As Integer
        End Function

        Public Function SumarCarroCompra(ByVal clcodigo As Integer,
        ByVal prcodigo As Integer, ByVal cantidad As Integer, ByVal
        accodigo As Integer) As Integer
        End Function
    End Class
```

Criptografia.

```
    Public Class CriptoDB
        Public Shared Function Encriptar(ByVal TextoPlano As String) As
        String
        End Function
    End Class
```

Disponibilidad.

```
Public Class DisponibilidadDB
    Public Function InsertarDisponibiidad(ByVal producto As Integer,
    ByVal empaque As Integer, ByVal existencia As Integer, ByVal
    precio As Double) As Integer
    End Function

    Public Function TraerProducto() As SqlDataReader
    End Function

    Public Function TraerListadoDisponibilidad() As DataSet
    End Function
End Class
```

Mail.

```
Public Class mailDB
    Public Function EnviarMail(ByVal destinatario As String, ByVal
    remitente As String, ByVal asunto As String, ByVal mensaje As
    String) As Integer
    End Function

    Public Function SolicitarSuscripcion(ByVal destinatario As String) As
    Integer
    End Function

    Public Function InsertarSuscripcion(ByVal destinatario As String,
    ByVal activa As Boolean) As Integer
    End Function

    Public Function ConfirmarSuscripcion(ByVal destinatario As String)
    As Integer
    End Function
```

```
        Public Function EliminarSuscripcion(ByVal destinatario As String)
        As Integer
        End Function
    End Class
```

Mercado Cliente.

```
    Public Class MercadoDB
        Public nombre As String

        Public Function InsertarMercadoCliente(ByVal nombre As String) As
        Integer
        End Function

        Public Function TraerListadoMercadoCliente() As DataSet
        End Function

        Public Function TraerMercadoClienteDetalle(ByVal codigo As
        String) As MercadoClienteDetalle
        End Function

        Public Function ModificarMercadoCliente(ByVal codigo As Integer,
        ByVal nombre As String) As Integer
        End Function

        Public Function EliminarMercadoCliente(ByVal codigo As Integer)
        As Integer
        End Function
    End Class
```

Orden

```
    Public Class OrdenDB
        Public orfecha As DateTime
        Public orfechasalidafinca As DateTime
        Public orfechavuelo As DateTime
        Public oraprobado As Boolean
```

```
Public orobservaciones As String
Public total As Single
Public items As DataSet

Public Function ColocarCompra(ByVal clcodigo As Integer) As
DataSet
End Function

Public Function TraerOrdenDetalle(ByVal orcodigo As Integer,
ByVal clcodigo As Integer) As OrdenDetalle
End Function

Public Function ChequearOrdenCliente(ByVal orcodigo As Integer,
ByVal clcodigo As Integer) As Integer
End Function

Public Function TraerOrdenesNuevas(ByVal uscodigo As String)
End Function

Public Function ActualizarOrden(ByVal orcodigo As Integer, ByVal
orfechasalidafinca As DateTime, ByVal orfechavuelo As DateTime,
ByVal orobservaciones As String, ByVal oraprobado As Boolean) As
Integer
End Function
Public Function ActualizarOrdenDetalle(ByVal orcodigo As Integer,
ByVal prcodigo As Integer, ByVal ordcantidad As Integer, ByVal
ordmarcacion As String, ByVal accodigo As Integer, ByVal
ordcomentario As String, ByVal prprecio As Single) As Integer
End Function
Public Function EliminarProductoPedido(ByVal orcodigo As Integer,
ByVal prcodigo As Integer) As Integer
End Function
```

```
        Public Function CambiarMarcaOrden(ByVal orcodigo As Integer,
        ByVal valor As Boolean) As Integer
        End Function

        Public Function TraerListaOrdenes(ByVal uscodigo As String)
        End Function
    End Class
```

Producto.

```
    Public Class ProductoDB
        Public prtallo As Integer
        Public prempaquexbunch As Integer
        Public vpcodigo As Integer
        Public prpromocion As Boolean

        Public Function TraerProductoDisponibilidad(ByVal vpcodigo As
        Integer) As DataSet
        End Function

        Public Function TraerCategoriaProductoDetalle(ByVal cpcodigo) As
        CategoriaProductoDetalle
        End Function

        Public Function TraerVariedadProductoDetalle(ByVal vpcodigo) As
        VariedadProductoDetalle
        End Function

        Public Function TraerVariedadades() As SqlDataReader
        End Function
        Public Function InsertarProducto(ByVal tallo As Integer, ByVal
        empaque As Integer, ByVal variedad As Integer, ByVal promocion As
        Boolean) As Integer
        End Function
```

```
        Public Function TraerListadoProducto() As DataSet
        End Function

        Public Function TraerProductoDetalle(ByVal codigo As Integer) As
        ProductoDetalle
        End Function

        Public Function ModificarProducto(ByVal codigo As Integer, ByVal
        tallo As Integer, ByVal empaque As Integer, ByVal variedad As
        Integer, ByVal promocion As Boolean) As Integer
        End Function

        Public Function TraerVariedadxCategoria(ByVal cpcodigo As
        Integer) As DataSet
        End Function

        Public Function VerDetalleVariedad(ByVal codigo As String) As
        DataSet
        End Function

        Public Function TraerTamanosProducto(ByVal vpcodigo As Integer)
        As DataSet
        End Function

        Public Function TraerListaCategoriaProducto() As DataSet
        End Function

        Public Function TraerProductosPromocion(ByVal vpcodigo As
        Integer)
        End Function
End Class

Public Class CategoriaProductoDB
        Public nombre As String
```

```
        Public Function InsertarCategoriaProducto(ByVal nombre As String)
        As Integer
        End Function

        Public Function TraerCategoriaProductoDetalle(ByVal codigo As
        String) As CategoriaProductoDetalle
        End Function

        Public Function TraerListadoCategoriaProducto() As DataSet
        End Function

        Public Function ModificarCategoriaProducto(ByVal codigo As
        Integer, ByVal nombre As String) As Integer
        End Function

        Public Function EliminarCategoriaProducto(ByVal codigo As Integer)
        As Integer
        End Function
    End Class
```

Roles

```
    Public Class RolesDB
        Public prnombre As String

        Public Function TraerListadoRol() As DataSet
        End Function

        Public Function TraerUsuarios() As SqlDataReader
        End Function

        Public Function TraerRoles() As SqlDataReader
        End Function

        Public Function InsertarRolesUsuario(ByVal usuario As String,
        ByVal rol As Integer) As Integer
```

```
        End Function

        Public Function TraerListadoUsuario() As DataSet
        End Function

        Public Function TraerRolAsignado(ByVal cedula As String) As
        DataSet
        End Function
    End Class
```

Seguridad.

```
    Public Class SeguridadDB
        Public Function TraerRol()
        End Function

        Public Function ChequearRol(ByVal uscodigo As String, ByVal
        rocodigo As Integer) As Boolean
        End Function
    End Class
```

Suscripcion.

```
    Public Class SuscripcionDB
        Public sumail As String
        Public suactiva As Boolean

        Public Function TraerListadoSuscripcion() As DataSet
        End Function

        Public Function TraerSuscripcionDetalle(ByVal codigo As Integer)
        As SuscripcionDetalle
        End Function
```

```
        Public Function DesactivarSuscripcion(ByVal codigo As Integer,
        ByVal activa As Boolean) As Integer
        End Function
    End Class
```

Usuario.

```
    Public Class UsuarioDetalle
        Public nombre As String
        Public apellido As String
        Public nombreusuario As String
        Public contrasena As String
        Public email As String
        Public telefono As String
        Public extension As String
        Public movil As String
        Public foto As String
        Public activo As Boolean

        Public Function IngresoUsuario(ByVal cedula As String, ByVal
        nombre As String, ByVal apellido As String, ByVal nombreusuario
        As String, ByVal contrasena As String, ByVal email As String, ByVal
        teltrabajo As String, ByVal extension As String, ByVal telmovil As
        String, ByVal foto As String, ByVal activo As Boolean) As Integer
        End Function

        Public Function TraerUsuarioDetalle(ByVal cedula As String) As
        UsuarioDetalle
        End Function

        Public Function ModificarUsuario(ByVal cedula As String, ByVal
        nombre As String, ByVal apellido As String, ByVal nombreusuario
        As String, ByVal contrasena As String, ByVal email As String, ByVal
        teltrabajo As String, ByVal extension As String, ByVal telmovil As
        String, ByVal foto As String, ByVal activo As Boolean) As Integer
```

```
        End Function

        Public Function EliminarUsuario(ByVal cedula As String) As Integer
        End Function

        Public Function TraerListadoUsuario() As DataSet
        End Function

        Public Function TraerRolesUsuario(ByVal uscodigo As String) As
        String()
        End Function

        Public Function LoginUsuario(ByVal nombreusuario As String,
        ByVal contrasena As String) As String
        End Function
    End Class
```

Variedad Producto.

```
    Public Class VariedadProductoDB
        Public vpsigla As String
        Public vpnombre As String
        Public vpdescripcion As String
        Public vptamcabeza As Integer
        Public cpcodigo As Integer
        Public vpdisponible As Boolean
        Public vpimagen As String

        Public Function InsertarVariedadProducto(ByVal sigla As String,
        ByVal nombre As String, ByVal descripcion As String, ByVal
        tamcabeza As Integer, ByVal categoria As Integer, ByVal disponible
        As Boolean, ByVal foto As String) As Integer
        End Function

        Public Function TraerListadoVariedadProducto() As DataSet
```

```
        End Function

        Public Function Traercategoria() As SqlDataReader
        End Function

        Public Function TraerVariedadProductoDetalle(ByVal codigo As
        Integer) As VariedadProductoDetalle
        End Function

        Public Function ModificarVariedadProducto(ByVal codigo As
        Integer, ByVal sigla As String, ByVal nombre As String, ByVal
        descripcion As String, ByVal tamcabeza As Integer, ByVal categoria
        As Integer, ByVal disponible As Boolean, ByVal foto As String) As
        Integer
        End Function


        Public Function EliminarVariedadProducto(ByVal codigo As Integer)
        As Integer
        End Function
    End Class
```

Mensaje.

```
    Public Class MensajeDB
        Public Function InsertarMensaje(ByVal cliente As String, ByVal
        asunto As String, ByVal mensaje As String, ByVal para As Boolean)
        As Integer
        End Function
        Public Function InsertarMensajeCliente(ByVal agente As String,
        ByVal cliente As String, ByVal asunto As String, ByVal mensaje As
        String, ByVal para As Boolean) As Integer
        End Function
        Public Function TraerMensajeCliente(ByVal codigo As Integer) As
        DataSet
        End Function
```

```
        Public Function TraerMensajeAgente(ByVal codigo As String) As
        DataSet
        End Function
        Public Function LeerMensaje(ByVal codigo As Integer, ByVal para
        As Boolean) As MensajeDetalle
        End Function
        Public Function MarcarMensajeLeido(ByVal codigo As Integer) As
        Integer
        End Function
        Public Function ContarMensajesCliente(ByVal codigo As Integer) As
        ContarMensaje
        End Function
        Public Function ContarMensajesAgente(ByVal cedula As Integer) As
        ContarMensaje
        End Function
        Public Function TraerClientesAgente(ByVal cedula As String) As
        SqlDataReader
        End Function
    End Class
```

4.2.2. Adición de los atributos de referencia.

Los atributos de referencia una de una clase están indicados por las asociaciones y la navegabilidad en los diagramas de clases.

A continuación se enumeran los atributos de referencia de E-Commerce:

Producto:

- Asociación a VariedadProducto con navegabilidad a esta última.
- Asociación a Disponibilidad con navegabilidad a esta última.
- Asociación a CarroCompra con navegabilidad a esta última.
- Asociación a OrdenDetalle con navegabilidad a esta última.

VariedadProducto:

- Asociación a CategoriaProducto con navegabilidad a esta última.

Orden:

- Asociación a OrdenDetalle con navegabilidad a esta última.
- Asociación a Cliente con navegabilidad a esta última.

OrdenDetalle:

- Asociación a AgenciaCarga con navegabilidad a esta última.

Cliente:

- Asociación a MercadoCliente con navegabilidad a esta última.

Usuario:

- Asociación a Rol con navegabilidad a esta última.

4.3 Creación de los métodos a través de los diagramas de colaboración.

Agencia de Carga.

```
Public Class AgenciaCargaDB
    Public Function InsertarAgenciaCarga(ByVal ruc As String, ByVal
    nombre As String) As Integer
        Dim conn As SqlConnection = New
        SqlConnection(ConfigurationSettings.AppSettings("conexion"))
        Dim comando As SqlCommand = New
        SqlCommand("Sp_InsertarAgenciaCarga", conn)
        comando.CommandType = CommandType.StoredProcedure
        Dim prm_ruc As SqlParameter = New SqlParameter("@acruc",
        SqlDbType.VarChar, 13)
        Dim prm_nombre As SqlParameter = New
        SqlParameter("@acnombre", SqlDbType.VarChar, 40)
        prm_ruc.Value = ruc.Trim()
        prm_nombre.Value = nombre.Trim()
        With comando.Parameters
            .Add(prm_ruc)
            .Add(prm_nombre)
```

```
        End With
        Try
            conn.Open()
            Dim insertados = comando.ExecuteNonQuery
            conn.Close()
            Return insertados
        Catch ex As Exception
            Return 0
        End Try
    End Function
    Public Function TraerListadoAgenciaCarga() As DataSet
        Dim Conn As SqlConnection = New
        SqlConnection(ConfigurationSettings.AppSettings("Conexion"))
        Dim comando As SqlCommand = New
        SqlCommand("Sp_TraerListadoAgenciaCarga", Conn)
        comando.CommandType = CommandType.StoredProcedure
        Dim da As SqlDataAdapter = New
        SqlDataAdapter("Sp_TraerListadoAgenciaCarga", Conn)
        da.SelectCommand.CommandType =
        CommandType.StoredProcedure
        Dim ds As DataSet = New DataSet
        da.Fill(ds, "ListadoAgenciaCarga")
        Return ds
    End Function
    Public Function TraerAgenciaCargaDetalle(ByVal codigo As String) As
    AgenciaCargaDetalle
        Dim Conn As SqlConnection = New
        SqlConnection(ConfigurationSettings.AppSettings("Conexion"))
        Dim comando As SqlCommand = New
        SqlCommand("Sp_TraerAgenciaCargaDetalle", Conn)
        comando.CommandType = CommandType.StoredProcedure
        Dim prm_codigo As SqlParameter = New
        SqlParameter("@accodigo", SqlDbType.Int)
```

```
        Dim prm_ruc As SqlParameter = New SqlParameter("@acruc",
        SqlDbType.VarChar, 13)
        Dim prm_nombre As SqlParameter = New
        SqlParameter("@acnombre", SqlDbType.VarChar, 40)
        prm_codigo.Value = codigo
        prm_ruc.Direction = ParameterDirection.Output
        prm_nombre.Direction = ParameterDirection.Output
        With comando.Parameters
            .Add(prm_codigo)
            .Add(prm_ruc)
            .Add(prm_nombre)
        End With
        Dim AgenciaCargaDetalle As AgenciaCargaDetalle = New
        AgenciaCargaDetalle
        Try
            Conn.Open()
            comando.ExecuteNonQuery()
            Conn.Close()
            With AgenciaCargaDetalle
                .ruc = prm_ruc.Value
                .nombre = prm_nombre.Value
            End With
        Catch ex As Exception
            With AgenciaCargaDetalle
                .ruc = ""
                .nombre = ""
            End With
        End Try
        Return AgenciaCargaDetalle
    End Function
    Public Function ModificarAgenciaCarga(ByVal codigo As Integer, ByVal
    ruc As String, ByVal nombre As String) As Integer
        Dim conn As SqlConnection = New
        SqlConnection(ConfigurationSettings.AppSettings("conexion"))
```

```
        Dim comando As SqlCommand = New
        SqlCommand("Sp_ModificarAgenciaCarga", conn)
        comando.CommandType = CommandType.StoredProcedure
        Dim prm_codigo As SqlParameter = New
        SqlParameter("@accodigo", SqlDbType.Int)
        Dim prm_ruc As SqlParameter = New SqlParameter("@acruc",
        SqlDbType.VarChar, 13)
        Dim prm_nombre As SqlParameter = New
        SqlParameter("@acnombre", SqlDbType.VarChar, 40)
        prm_codigo.Value = código
        prm_ruc.Value = ruc
        prm_nombre.Value = nombre
        With comando.Parameters
            .Add(prm_codigo)
            .Add(prm_ruc)
            .Add(prm_nombre)
        End With
        Try
            conn.Open()
            Dim modificados = comando.ExecuteNonQuery
            conn.Close()
            Return modificados
        Catch ex As Exception
            Return 0
        End Try
    End Function

    Public Function EliminarAgenciaCarga(ByVal codigo As Integer) As Integer
        Dim conn As SqlConnection = New
        SqlConnection(ConfigurationSettings.AppSettings("conexion"))
        Dim comando As SqlCommand = New
        SqlCommand("Sp_EliminarAgenciaCarga", conn)
        comando.CommandType = CommandType.StoredProcedure
```

```
            Dim prm_codigo As SqlParameter = New
            SqlParameter("@accodigo", SqlDbType.Int)
            prm_codigo.Value = código
            comando.Parameters.Add(prm_codigo)
            Try
                conn.Open()
                Dim eliminados = comando.ExecuteNonQuery
                conn.Close()
                Return eliminados
            Catch ex As Exception
                Return 0
            End Try
        End Function
End Class
```

Carro Compra.

```
Public Class CarroCompraDB
        Public Function TraerCarroCompra(ByVal clcodigo As Integer) As DataSet
            Dim Conn As SqlConnection = New
            SqlConnection(ConfigurationSettings.AppSettings("Conexion"))
            Dim da As SqlDataAdapter = New
            SqlDataAdapter("sp_TraerCarroCompra", Conn)
            Dim prm_clcodigo As New SqlParameter("@clcodigo",
SqlDbType.Int)
            prm_clcodigo.Value = clcodigo
            da.SelectCommand.CommandType =
CommandType.StoredProcedure
            da.SelectCommand.Parameters.Add(prm_clcodigo)
            Dim ds As DataSet = New DataSet
            da.Fill(ds, "CarroCompra")
            Return ds
        End Function
        Public Function CalcularTotalCarro(ByVal clcodigo As Integer) As Single
```

```
        Dim conn As SqlConnection = New
        SqlConnection(ConfigurationSettings.AppSettings("Conexion"))
        Dim comando As SqlCommand = New
        SqlCommand("Sp_CalcularTotalCarro", conn)
        comando.CommandType = CommandType.StoredProcedure
        Dim prm_cliente As SqlParameter = New SqlParameter("@clcodigo",
        SqlDbType.Int)
        Dim prm_total As SqlParameter = New SqlParameter("@total",
        SqlDbType.Money)
        prm_cliente.Value = clcodigo
        prm_total.Direction = ParameterDirection.Output
        With comando.Parameters
            .Add(prm_cliente)
            .Add(prm_total)
        End With
        Try
            conn.Open()
            comando.ExecuteNonQuery()
            conn.Close()
            Return prm_total.Value
        Catch ex As Exception
            Return 0
        End Try
    End Function
    Public Function InsertarCarroCompra(ByVal cliente As Integer, ByVal
    producto As Integer, ByVal cantidad As Integer, ByVal accodigo As Integer)
    As Integer
        Dim Conn As SqlConnection = New
        SqlConnection(ConfigurationSettings.AppSettings("conexion"))
        Dim comando As SqlCommand = New
        SqlCommand("Sp_InsertarCarroCompra", Conn)
        comando.CommandType = CommandType.StoredProcedure
        Dim prm_cliente As SqlParameter = New SqlParameter("@clcodigo",
        SqlDbType.Int)
```

```
        Dim prm_producto As SqlParameter = New
        SqlParameter("@prcodigo", SqlDbType.Int)
        Dim prm_cantidad As SqlParameter = New
        SqlParameter("@cccantidad", SqlDbType.Int)
        Dim prm_accodigo As SqlParameter = New
        SqlParameter("@accodigo", SqlDbType.Int)
        prm_cliente.Value = cliente
        prm_producto.Value = producto
        prm_cantidad.Value = cantidad
        prm_accodigo.Value = accodigo
        With comando.Parameters
            .Add(prm_cliente)
            .Add(prm_producto)
            .Add(prm_cantidad)
            .Add(prm_accodigo)
        End With
        Try
            Conn.Open()
            Dim insertados = comando.ExecuteNonQuery()
            Conn.Close()
            Return insertados
        Catch ex As Exception
            Return 0
        End Try
    End Function
    Public Function ActualizarCarroCompra(ByVal cliente As Integer, ByVal
    producto As Integer, ByVal cantidad As Integer, ByVal marcacion As String,
    ByVal agencia As Integer, ByVal comentario As String) As Integer
        Dim Conn As SqlConnection = New
        SqlConnection(ConfigurationSettings.AppSettings("Conexion"))
        Dim comando As SqlCommand = New
        SqlCommand("Sp_ActualizarCarroCompra", Conn)
        comando.CommandType = CommandType.StoredProcedure
```

```
Dim prm_cliente As SqlParameter = New SqlParameter("@clcodigo",
SqlDbType.Int)
Dim prm_producto As SqlParameter = New
SqlParameter("@prcodigo", SqlDbType.Int)
Dim prm_cantidad As SqlParameter = New
SqlParameter("@cccantidad", SqlDbType.SmallInt)
Dim prm_marcacion As SqlParameter = New
SqlParameter("@ccmarcacion", SqlDbType.VarChar, 50)
Dim prm_agencia As SqlParameter = New
SqlParameter("@accodigo", SqlDbType.Int)
Dim prm_comentario As SqlParameter = New
SqlParameter("@cccomentario", SqlDbType.VarChar, 100)
prm_cliente.Value = cliente
prm_producto.Value = producto
prm_cantidad.Value = cantidad
prm_marcacion.Value = marcacion
prm_agencia.Value = agencia
prm_comentario.Value = comentario
With comando.Parameters
        .Add(prm_cliente)
        .Add(prm_producto)
        .Add(prm_cantidad)
        .Add(prm_marcacion)
        .Add(prm_agencia)
        .Add(prm_comentario)
End With
Try
        Conn.Open()
        Dim modificados = comando.ExecuteNonQuery()
        Conn.Close()
        Return modificados
Catch ex As Exception
        Return 0
End Try
```

```
End Function
Public Function TraerAgenciaCargaCarroCompra(ByVal clcodigo As
Integer, ByVal prcodigo As Integer) As Integer
        Dim Conn As SqlConnection = New
        SqlConnection(ConfigurationSettings.AppSettings("Conexion"))
        Dim comando As SqlCommand = New
        SqlCommand("Sp_TraerAgenciaCargaCarroCompra", Conn)
        comando.CommandType = CommandType.StoredProcedure
        Dim prm_clcodigo As New SqlParameter("@clcodigo",
        SqlDbType.Int)
        Dim prm_prcodigo As New SqlParameter("@prcodigo",
        SqlDbType.Int)
        Dim prm_accodigo As New SqlParameter("@accodigo",
        SqlDbType.Int)
        prm_clcodigo.Value = clcodigo
        prm_prcodigo.Value = prcodigo
        prm_accodigo.Direction = ParameterDirection.Output
        With comando.Parameters
           .Add(prm_clcodigo)
           .Add(prm_prcodigo)
           .Add(prm_accodigo)
        End With
        Try
                Conn.Open()
                comando.ExecuteNonQuery()
                Conn.Close()
                Return prm_prcodigo.Value
        Catch ex As Exception
                Return 0
        End Try
End Function
Public Function ConfirmarCompra(ByVal clcodigo As String) As Integer
        Dim Conn As SqlConnection = New
        SqlConnection(ConfigurationSettings.AppSettings("Conexion"))
```

```
        Dim comando As SqlCommand = New
        SqlCommand("Sp_ConfirmarCompra", Conn)
        comando.CommandType = CommandType.StoredProcedure
        Dim prm_clcodigo As New SqlParameter("@clcodigo",
        SqlDbType.Int)
        Dim prm_CodigoOrden As New SqlParameter("@codigoOrden",
        SqlDbType.Int)
        prm_clcodigo.Value = clcodigo
        prm_CodigoOrden.Direction = ParameterDirection.Output
        With comando.Parameters
            .Add(prm_clcodigo)
            .Add(prm_CodigoOrden)
        End With
        Try
            Conn.Open()
            comando.ExecuteNonQuery()
            Conn.Close()
            Return prm_CodigoOrden.Value
        Catch ex As Exception
            Return 0
        End Try
    End Function
    Public Function VaciarCarroCompra(ByVal clcodigo As Integer) As Integer
        Dim Conn As SqlConnection = New
        SqlConnection(ConfigurationSettings.AppSettings("Conexion"))
        Dim comando As SqlCommand = New
        SqlCommand("Sp_VaciarCarroCompra", Conn)
        comando.CommandType = CommandType.StoredProcedure
        Dim prm_clcodigo As New SqlParameter("@clcodigo",
        SqlDbType.Int)
        prm_clcodigo.Value = clcodigo
        With comando.Parameters
            .Add(prm_clcodigo)
        End With
```

```
            Try
                Conn.Open()
                comando.ExecuteNonQuery()
                Conn.Close()
            Catch ex As Exception
            End Try
    End Function
    Public Function TraerCodigoCarroCompra()
            Dim contexto As HttpContext = HttpContext.Current
            If contexto.User.Identity.Name <> "" Then
                Return contexto.User.Identity.Name
            End If
            if Not contexto.Request.Cookies("Ecommerce_CodigoCarro") Is
            Nothing Then
                Return
                contexto.Request.Cookies("Ecommerce_CodigoCarro").Value
            Else
                Dim CarroCompraTemp As Guid = Guid.NewGuid()
                contexto.Response.Cookies("Ecommerce_CodigoCarro").Valu
                e = CarroCompraTemp.ToString()
                Return CarroCompraTemp.ToString()
            End If
    End Function
    Public Function ContarProductosCarro(ByVal clcodigo As Integer) As
    Integer
            Dim Conn As SqlConnection = New
            SqlConnection(ConfigurationSettings.AppSettings("Conexion"))
            Dim comando As SqlCommand = New
            SqlCommand("Sp_ContarProductosCarro", Conn)
            comando.CommandType = CommandType.StoredProcedure
            Dim prm_clcodigo As New SqlParameter("@clcodigo",
            SqlDbType.Int)
            Dim prm_items As New SqlParameter("@items", SqlDbType.Int)
            prm_clcodigo.Value = clcodigo
```

```
        prm_items.Direction = ParameterDirection.Output
        With comando.Parameters
            .Add(prm_clcodigo)
            .Add(prm_items)
        End With
        Try
            Conn.Open()
            comando.ExecuteNonQuery()
            Conn.Close()
            Return CInt(prm_items.Value)
        Catch ex As Exception
            Return 0
        End Try
    End Function
    Public Function QuitarProductoCarro(ByVal clcodigo As Integer, ByVal
    prcodigo As Integer) As Integer
        Dim Conn As SqlConnection = New
        SqlConnection(ConfigurationSettings.AppSettings("Conexion"))
        Dim comando As SqlCommand = New
        SqlCommand("Sp_QuitarProductoCarro", Conn)
        comando.CommandType = CommandType.StoredProcedure
        Dim prm_clcodigo As New SqlParameter("@clcodigo",
        SqlDbType.Int)
        Dim prm_prcodigo As New SqlParameter("@prcodigo",
        SqlDbType.Int)
        prm_clcodigo.Value = clcodigo
        prm_prcodigo.Value = prcodigo
        With comando.Parameters
            .Add(prm_clcodigo)
            .Add(prm_prcodigo)
        End With
        Try
            Conn.Open()
            Dim r As Integer = comando.ExecuteNonQuery
```

```
            Conn.Close()
            Return r
        Catch ex As Exception
            Return 0
        End Try
    End Function
    Public Function ChequearProductoCarro(ByVal clcodigo As Integer, ByVal
    prcodigo As Integer) As Integer
        Dim Conn As SqlConnection = New
        SqlConnection(ConfigurationSettings.AppSettings("Conexion"))
        Dim comando As SqlCommand = New
        SqlCommand("Sp_ChequearProductoCarro", Conn)
        comando.CommandType = CommandType.StoredProcedure
        Dim prm_clcodigo As New SqlParameter("@clcodigo",
        SqlDbType.Int)
        Dim prm_prcodigo As New SqlParameter("@prcodigo",
        SqlDbType.Int)
        Dim prm_existe As New SqlParameter("@existe", SqlDbType.Int)
        prm_clcodigo.Value = clcodigo
        prm_prcodigo.Value = prcodigo
        prm_existe.Direction = ParameterDirection.Output
        With comando.Parameters
            .Add(prm_clcodigo)
            .Add(prm_prcodigo)
            .Add(prm_existe)
        End With
        Try
            Conn.Open()
            comando.ExecuteNonQuery()
            Conn.Close()
            Return prm_existe.Value
        Catch ex As Exception
            Return 0
        End Try
```

```
End Function
Public Function SumarCarroCompra(ByVal clcodigo As Integer, ByVal
prcodigo As Integer, ByVal cantidad As Integer, ByVal accodigo As Integer)
As Integer
    Dim Conn As SqlConnection = New
    SqlConnection(ConfigurationSettings.AppSettings("Conexion"))
    Dim comando As SqlCommand = New
    SqlCommand("Sp_SumarCarroCompra", Conn)
    comando.CommandType = CommandType.StoredProcedure
    Dim prm_clcodigo As New SqlParameter("@clcodigo",
    SqlDbType.Int)
    Dim prm_prcodigo As New SqlParameter("@prcodigo",
    SqlDbType.Int)
    Dim prm_cantidad As New SqlParameter("@cantidad",
    SqlDbType.Int)
    Dim prm_accodigo As New SqlParameter("@accodigo",
    SqlDbType.Int)
    prm_clcodigo.Value = clcodigo
    prm_prcodigo.Value = prcodigo
    prm_cantidad.Value = cantidad
    prm_accodigo.Value = accodigo
    With comando.Parameters
        .Add(prm_clcodigo)
        .Add(prm_prcodigo)
        .Add(prm_cantidad)
        .Add(prm_accodigo)
    End With
    Try
        Conn.Open()
        Dim r = comando.ExecuteNonQuery()
        Return r
        Conn.Close()
    Catch ex As Exception
        Return 0
```

```
            End Try
        End Function
    End Class
```

Clientes.

```
    Public Class ClienteDB
        Public Function TraerMercadoCliente() As SqlDataReader
            Dim conn As SqlConnection = New
            SqlConnection(ConfigurationSettings.AppSettings("conexion"))
            Dim comando As SqlCommand = New
            SqlCommand("Sp_TraerMercadoCliente", conn)
            comando.CommandType = CommandType.StoredProcedure
            conn.Open()
            Dim mercadocliente As SqlDataReader =
            comando.ExecuteReader(CommandBehavior.CloseConnection)
            Return mercadocliente
        End Function
        Public Function TraerAgenteVendedor() As SqlDataReader
            Dim conn As SqlConnection = New
            SqlConnection(ConfigurationSettings.AppSettings("conexion"))
            Dim comando As SqlCommand = New
            SqlCommand("Sp_TraerAgenteVendedor", conn)
            comando.CommandType = CommandType.StoredProcedure
            conn.Open()
            Dim agente As SqlDataReader =
            comando.ExecuteReader(CommandBehavior.CloseConnection)
            Return agente
        End Function
        Public Function InsertarCliente(ByVal nombre As String, ByVal
        nombreusuario As String, ByVal contrasena As String, ByVal email As
        String, ByVal dispomin As Double, ByVal dispomax As Double, ByVal
        mercado As Integer, ByVal agente As String) As Integer
```

```
Dim conn As SqlConnection = New
SqlConnection(ConfigurationSettings.AppSettings("conexion"))
Dim comando As SqlCommand = New
SqlCommand("Sp_InsertarCliente", conn)
comando.CommandType = CommandType.StoredProcedure
Dim prm_nombre As SqlParameter = New
SqlParameter("@clnombre", SqlDbType.VarChar, 50)
Dim prm_nombreusuario As SqlParameter = New
SqlParameter("@clnombreusuario", SqlDbType.VarChar, 30)
Dim prm_contrasena As SqlParameter = New
SqlParameter("@clcontrasena", SqlDbType.VarChar, 255)
Dim prm_email As SqlParameter = New SqlParameter("@clemail",
SqlDbType.VarChar, 100)
Dim prm_dispomin As SqlParameter = New
SqlParameter("@cltamdispomin", SqlDbType.SmallInt)
Dim prm_dispomax As SqlParameter = New
SqlParameter("@cltamdispmax", SqlDbType.SmallInt)
Dim prm_mercado As SqlParameter = New
SqlParameter("@mccodigo", SqlDbType.Int)
Dim prm_agente As SqlParameter = New SqlParameter("@uscedula",
SqlDbType.VarChar, 10)
Dim critoDB As New Ecommerce.CriptoDB
prm_nombre.Value = nombre
prm_nombreusuario.Value = nombreusuario
prm_contrasena.Value = CriptoDB.Encriptar(contrasena)
prm_email.Value = email
prm_dispomin.Value = dispomin
prm_dispomax.Value = dispomax
prm_mercado.Value = mercado
prm_agente.Value = agente
With comando.Parameters
        .Add(prm_nombre)
        .Add(prm_nombreusuario)
        .Add(prm_contrasena)
```

```
                .Add(prm_email)
                .Add(prm_dispomin)
                .Add(prm_dispomax)
                .Add(prm_mercado)
                .Add(prm_agente)
        End With
        Try
                conn.Open()
                Dim insertados = comando.ExecuteNonQuery()
                conn.Close()
                Return insertados
        Catch ex As Exception
                Return 0
        End Try
End Function
Public Function TraerListadoCliente() As DataSet
        Dim Conn As SqlConnection = New
        SqlConnection(ConfigurationSettings.AppSettings("Conexion"))
        Dim da As SqlDataAdapter = New
        SqlDataAdapter("Sp_TraerListadoCliente", Conn)
        da.SelectCommand.CommandType =
        CommandType.StoredProcedure
        Dim ds As DataSet = New DataSet
        da.Fill(ds, "ListadoCliente")
        Return ds
End Function
Public Function TraerClienteDetalle(ByVal codigo As Integer) As
ClienteDetalle
        Dim Conn As SqlConnection = New
        SqlConnection(ConfigurationSettings.AppSettings("Conexion"))
        Dim comando As SqlCommand = New
        SqlCommand("Sp_TraerClienteDetalle", Conn)
        comando.CommandType = CommandType.StoredProcedure
```

```
Dim prm_codigo As SqlParameter = New SqlParameter("@clcodigo",
SqlDbType.Int)
Dim prm_nombre As SqlParameter = New
SqlParameter("@clnombre", SqlDbType.VarChar, 50)
Dim prm_nombreusuario As SqlParameter = New
SqlParameter("@clnombreusuario", SqlDbType.VarChar, 30)
Dim prm_contrasena As SqlParameter = New
SqlParameter("@clcontrasena", SqlDbType.VarChar, 255)
Dim prm_email As SqlParameter = New SqlParameter("@clemail",
SqlDbType.VarChar, 100)
Dim prm_tamdispomin As SqlParameter = New
SqlParameter("@cltamdispomin", SqlDbType.SmallInt)
Dim prm_tamdispmax As SqlParameter = New
SqlParameter("@cltamdispmax", SqlDbType.SmallInt)
Dim prm_mercado As SqlParameter = New
SqlParameter("@mccodigo", SqlDbType.Int)
Dim prm_agente As SqlParameter = New SqlParameter("@uscedula",
SqlDbType.VarChar, 10)
prm_codigo.Value = codigo
prm_nombre.Direction = ParameterDirection.Output
prm_nombreusuario.Direction = ParameterDirection.Output
prm_contrasena.Direction = ParameterDirection.Output
prm_email.Direction = ParameterDirection.Output
prm_tamdispomin.Direction = ParameterDirection.Output
prm_tamdispmax.Direction = ParameterDirection.Output
prm_mercado.Direction = ParameterDirection.Output
prm_agente.Direction = ParameterDirection.Output
With comando.Parameters
        .Add(prm_codigo)
        .Add(prm_nombre)
        .Add(prm_nombreusuario)
        .Add(prm_contrasena)
        .Add(prm_email)
        .Add(prm_tamdispomin)
```

```
                .Add(prm_tamdispmax)
                .Add(prm_mercado)
                .Add(prm_agente)
            End With
            Dim ClienteDetalle As ClienteDetalle = New ClienteDetalle
            Try
                Conn.Open()
                comando.ExecuteNonQuery()
                Conn.Close()
                With ClienteDetalle
                    .nombre = prm_nombre.Value
                    .nombreusuario = prm_nombreusuario.Value
                    .contrasena = prm_contrasena.Value
                    .email = prm_email.Value
                    .tamdispomin = prm_tamdispomin.Value
                    .tamdispomax = prm_tamdispmax.Value
                    .mercado = prm_mercado.Value
                    .agente = prm_agente.Value
                End With
            Catch ex As Exception
                With ClienteDetalle
                    .nombre = ""
                    .nombreusuario = ""
                    .contrasena = ""
                    .email = ""
                    .tamdispomin = 0
                    .tamdispomax = 0
                    .mercado = 0
                    .agente = ""
                End With
            End Try
            Return ClienteDetalle
        End Function
```

```
Public Function ModificarCliente(ByVal codigo As Integer, ByVal nombre
As String, ByVal nombreusuario As String, ByVal contrasena As String,
ByVal email As String, ByVal dispomin As Double, ByVal dispomax As
Double, ByVal mercado As Integer, ByVal agente As String) As Integer
        Dim conn As SqlConnection = New
        SqlConnection(ConfigurationSettings.AppSettings("conexion"))
        Dim comando As SqlCommand = New
        SqlCommand("Sp_ModificarCliente", conn)
        comando.CommandType = CommandType.StoredProcedure
        Dim prm_codigo As SqlParameter = New SqlParameter("@clcodigo",
        SqlDbType.Int)
        Dim prm_nombre As SqlParameter = New
        SqlParameter("@clnombre", SqlDbType.VarChar, 50)
        Dim prm_nombreusuario As SqlParameter = New
        SqlParameter("@clnombreusuario", SqlDbType.VarChar, 30)
        Dim prm_contrasena As SqlParameter = New
        SqlParameter("@clcontrasena", SqlDbType.VarChar, 255)
        Dim prm_email As SqlParameter = New SqlParameter("@clemail",
        SqlDbType.VarChar, 100)
        Dim prm_dispomin As SqlParameter = New
        SqlParameter("@cltamdispomin", SqlDbType.SmallInt)
        Dim prm_dispomax As SqlParameter = New
        SqlParameter("@cltamdispmax", SqlDbType.SmallInt)
        Dim prm_mercado As SqlParameter = New
        SqlParameter("@mccodigo", SqlDbType.Int)
        Dim prm_agente As SqlParameter = New SqlParameter("@uscedula",
        SqlDbType.VarChar, 10)
        prm_codigo.Value = código
        prm_nombre.Value = nombre
        prm_nombreusuario.Value = nombreusuario
        prm_contrasena.Value = contrasena
        prm_email.Value = email
        prm_dispomin.Value = dispomin
        prm_dispomax.Value = dispomax
```

```
            prm_mercado.Value = mercado
            prm_agente.Value = agente
            With comando.Parameters
               .Add(prm_codigo)
               .Add(prm_nombre)
               .Add(prm_nombreusuario)
               .Add(prm_contrasena)
               .Add(prm_email)
               .Add(prm_dispomin)
               .Add(prm_dispomax)
               .Add(prm_mercado)
               .Add(prm_agente)
            End With
            Try
                  conn.Open()
                  Dim modificados = comando.ExecuteNonQuery()
                  conn.Close()
                  Return modificados
             Catch ex As Exception
                  Return 0
            End Try
      End Function
      Public Function EliminarCliente(ByVal codigo As Integer) As Integer
            Dim Conn As SqlConnection = New
            SqlConnection(ConfigurationSettings.AppSettings("Conexion"))
            Dim comando As SqlCommand = New
            SqlCommand("Sp_EliminarCliente", Conn)
            comando.CommandType = CommandType.StoredProcedure
            Dim prm_codigo As SqlParameter = New SqlParameter("@clcodigo",
            SqlDbType.Int)
            prm_codigo.Value = código
            comando.Parameters.Add(prm_codigo)
            Try
                  Conn.Open()
```

```
                Dim eliminados = comando.ExecuteNonQuery
                Conn.Close()
                Return eliminados
            Catch ex As Exception
                Return 0
            End Try
        End Function
        Public Function LoginCliente(ByVal NombreUsuario As String, ByVal
        Contrasena As String) As Integer
            Dim conn As SqlConnection = New
            SqlConnection(ConfigurationSettings.AppSettings("Conexion"))
            Dim comando As SqlCommand = New
            SqlCommand("Sp_LoginCliente", conn)
            comando.CommandType = CommandType.StoredProcedure
            Dim prm_nombreUsuario As New SqlParameter("@NombreUsuario",
            SqlDbType.VarChar, 30)
            Dim prm_contrasena As New SqlParameter("@Contrasena",
            SqlDbType.VarChar, 255)
            Dim prm_clcodigo As New SqlParameter("@clcodigo",
            SqlDbType.Int)
            prm_nombreUsuario.Value = NombreUsuario
            prm_contrasena.Value = Contrasena
            prm_clcodigo.Direction = ParameterDirection.Output
            With comando.Parameters
                .Add(prm_nombreUsuario)
                .Add(prm_contrasena)
                .Add(prm_clcodigo)
            End With
            Try
                conn.Open()
                comando.ExecuteNonQuery()
                conn.Close()
            Catch ex As Exception
            End Try
```

```
            Dim clcodigo As Integer = CInt(prm_clcodigo.Value)
            If clcodigo = 0 Then
                Return Nothing
            Else
                Return clcodigo
            End If
        End Function
        Public Function TraerHistorialCliente(ByVal clcodigo As Integer) As
        DataSet
            Dim Conn As SqlConnection = New
            SqlConnection(ConfigurationSettings.AppSettings("Conexion"))
            Dim da As SqlDataAdapter = New
            SqlDataAdapter("Sp_TraerHistorialCliente", Conn)
            da.SelectCommand.CommandType =
            CommandType.StoredProcedure
            Dim prm_clcodigo As New SqlParameter("@clcodigo",
            SqlDbType.Int)
            prm_clcodigo.Value = clcodigo
            da.SelectCommand.Parameters.Add(prm_clcodigo)
            Dim ds As New DataSet
            da.Fill(ds, "HistorialCliente")
            Return ds
        End Function
    End Class
```

Criptografía.

```
    Public Class CriptoDB
        Public Shared Function Encriptar(ByVal TextoPlano As String) As String
            Dim BytesPlanos As [Byte]()
            BytesPlanos = New UnicodeEncoding().GetBytes(TextoPlano)
            Dim BytesHash As [Byte]() =
            CType(CryptoConfig.CreateFromName("MD5"),
            HashAlgorithm).ComputeHash(BytesPlanos)
```

```
            Dim TextoEncriptado As String = BitConverter.ToString(BytesHash)
            Return TextoEncriptado
        End Function
    End Class
```

Disponibilidad.

```
    Public Class DisponibilidadDB
        Public Function InsertarDisponibiidad(ByVal producto As Integer, ByVal
        empaque As Integer, ByVal existencia As Integer, ByVal precio As Double)
        As Integer
            Dim conn As SqlConnection = New
            SqlConnection(ConfigurationSettings.AppSettings("conexion"))
            Dim comando As SqlCommand = New
            SqlCommand("Sp_InsertarDisponibilidad", conn)
            comando.CommandType = CommandType.StoredProcedure
            Dim prm_producto As SqlParameter = New
            SqlParameter("@prcodigo", SqlDbType.Int)
            Dim prm_empaque As SqlParameter = New
            SqlParameter("@diempaquexcaja", SqlDbType.SmallInt)
            Dim prm_existencia As SqlParameter = New
            SqlParameter("@diexistencia", SqlDbType.SmallInt)
            Dim prm_precio As SqlParameter = New SqlParameter("@diprecio",
            SqlDbType.Money)
            prm_producto.Value = producto
            prm_empaque.Value = empaque
            prm_existencia.Value = existencia
            prm_precio.Value = precio
            With comando.Parameters
                .Add(prm_producto)
                .Add(prm_empaque)
                .Add(prm_existencia)
                .Add(prm_precio)
            End With
```

```
        Try
            conn.Open()
            Dim insertados = comando.ExecuteNonQuery
            conn.Close()
            Return insertados
        Catch ex As Exception
            Return 0
        End Try
    End Function
    Public Function TraerProducto() As SqlDataReader
        Dim conn As SqlConnection = New
        SqlConnection(ConfigurationSettings.AppSettings("conexion"))
        Dim comando As SqlCommand = New
        SqlCommand("Sp_TraerProducto", conn)
        comando.CommandType = CommandType.StoredProcedure
        conn.Open()
        Dim producto As SqlDataReader =
        comando.ExecuteReader(CommandBehavior.CloseConnection)
        Return producto
    End Function
    Public Function TraerListadoDisponibilidad() As DataSet
        Dim Conn As SqlConnection = New
        SqlConnection(ConfigurationSettings.AppSettings("Conexion"))
        Dim comando As SqlCommand = New
        SqlCommand("Sp_TraerListadoDisponibilidad", Conn)
        comando.CommandType = CommandType.StoredProcedure
        Dim da As SqlDataAdapter = New
        SqlDataAdapter("Sp_TraerListadoDisponibilidad", Conn)
        da.SelectCommand.CommandType =
        CommandType.StoredProcedure
        Dim ds As DataSet = New DataSet
        da.Fill(ds, "ListadoDisponibilidad")
        Return ds
    End Function
```

```
End Class
```

Mail.

```
Public Class mailDB
    Public Function EnviarMail(ByVal destinatario As String, ByVal remitente
    As String, ByVal asunto As String, ByVal mensaje As String) As Integer
        Dim mail As New MailMessage
        With mail
            .To = destinatario
            .From = remitente
            .Subject = asunto
            .Body = mensaje
            .BodyFormat = MailFormat.Text
            .Priority = MailPriority.High
        End With
        Try
            SmtpMail.SmtpServer = "192.168.10.87"
            SmtpMail.Send(mail)
            mail = Nothing
            Return 0
        Catch ex As Exception
            Return 1
        End Try
    End Function
    Public Function SolicitarSuscripcion(ByVal destinatario As String) As
    Integer
        Dim texto = "TrebolRoses" + Chr(13)
        texto += "Always the best Roses" + Chr(13) + Chr(13)
        texto += "Usted o alguien más, ha proporcionado su dirección de
        correo electrónico para suscribirse a nuestra lista." + Chr(13)
        texto += "Para confirmar su suscripción a nuestra lista, haga click (o
        copie y pegue la dirección en un explorador)en el siguiente vínculo
        para activar su suscripción. " + Chr(13)
```

```
            texto +=
            http://192.168.10.137/ecommerce/frmConfirmarSuscripcion.aspx?dest
            inatario=" + destinatario + Chr(13)
            texto += "Si usted no ha solicitado suscribirse a nuestra lista, su
            dirección de correo será eliminada de nuestra base de datos."
            Dim maildb As New mailDB
            Dim mail As Integer = maildb.EnviarMail(destinatario,
            "info@trebolroses.com", "Suscripción a TrebolRoses.com", texto)
            Return mail
        End Function
        Public Function InsertarSuscripcion(ByVal destinatario As String, ByVal
        activa As Boolean) As Integer
            Dim conn As SqlConnection = New
            SqlConnection(ConfigurationSettings.AppSettings("conexion"))
            Dim comando As SqlCommand = New
            SqlCommand("Sp_InsertarSuscripcion", conn)
            comando.CommandType = CommandType.StoredProcedure
            Dim prm_destinatario As New SqlParameter("@destinatario",
            SqlDbType.VarChar, 255)
            Dim prm_activa As New SqlParameter("@activa", SqlDbType.Bit)
            prm_destinatario.Value = destinatario
            prm_activa.Value = activa
            With comando.Parameters
                .Add(prm_destinatario)
                .Add(prm_activa)
            End With
            Try
                conn.Open()
                Dim r As Integer = comando.ExecuteNonQuery()
                conn.Close()
                Return r
            Catch ex As Exception
                Return 0
            End Try
```

```
End Function
 Public Function ConfirmarSuscripcion(ByVal destinatario As String) As
Integer
        Dim texto = "TrebolRoses" + Chr(13)
        texto += "Always the best Roses" + Chr(13) + Chr(13)
        texto += "Gracias por suscribirse a nuestra lista de correo electrónico."
        + Chr(13)
        texto += "Ahora usted recibirá periódicamente nuestras ofertas y
        novedades" + Chr(13)
        texto += "Si usted no ha solicitado suscribirse a nuestar lista, su
        dirección de correo será eliminada de nuestra base de datos."
        Dim maildb As New mailDB
        Dim mail As Integer = maildb.EnviarMail(destinatario,
        "info@trebolroses.com", "Suscripción a TrebolRoses.com", texto)
        Return mail
End Function
Public Function EliminarSuscripcion(ByVal destinatario As String) As
Integer
        Dim conn As SqlConnection = New
        SqlConnection(ConfigurationSettings.AppSettings("conexion"))
        Dim comando As SqlCommand = New
        SqlCommand("Sp_EliminarSuscripcion", conn)
        comando.CommandType = CommandType.StoredProcedure
        Dim prm_destinatario As New SqlParameter("@destinatario",
        SqlDbType.VarChar, 255)
        prm_destinatario.Value = destinatario
        With comando.Parameters
                .Add(prm_destinatario)
        End With
        Try
                conn.Open()
                Dim r As Integer = comando.ExecuteNonQuery()
                conn.Close()
                Return r
```

```
            Catch ex As Exception
                Return 0
            End Try
        End Function
    End Class
```

Mercado Cliente.

```
Public Class MercadoClienteDB
    Public Function InsertarMercadoCliente(ByVal nombre As String) As Integer
        Dim Conn As SqlConnection = New
        SqlConnection(ConfigurationSettings.AppSettings("Conexion"))
        Dim comando As SqlCommand = New
        SqlCommand("Sp_InsertarMercadoCliente", Conn)
        comando.CommandType = CommandType.StoredProcedure
        Dim prm_nombre As SqlParameter = New
        SqlParameter("@mcnombre", SqlDbType.VarChar, 40)
        prm_nombre.Value = nombre
        comando.Parameters.Add(prm_nombre)
        Try
            Conn.Open()
            Dim insertados = comando.ExecuteNonQuery()
            Conn.Close()
            Return insertados
        Catch ex As Exception
            Return 0
        End Try
    End Function
    Public Function TraerListadoMercadoCliente() As DataSet
        Dim Conn As SqlConnection = New
        SqlConnection(ConfigurationSettings.AppSettings("Conexion"))
        Dim comando As SqlCommand = New
        SqlCommand("Sp_TraerListadoMercadoCliente", Conn)
        comando.CommandType = CommandType.StoredProcedure
```

```
        Dim da As SqlDataAdapter = New
        SqlDataAdapter("Sp_TraerListadoMercadoCliente", Conn)
        da.SelectCommand.CommandType =
        CommandType.StoredProcedure
        Dim ds As DataSet = New DataSet
        da.Fill(ds, "ListadoMercadoCliente")
        Return ds
    End Function
    Public Function TraerMercadoClienteDetalle(ByVal codigo As String) As
    MercadoClienteDetalle
        Dim Conn As SqlConnection = New
        SqlConnection(ConfigurationSettings.AppSettings("Conexion"))
        Dim comando As SqlCommand = New
        SqlCommand("Sp_TraerMercadoClienteDetalle", Conn)
        comando.CommandType = CommandType.StoredProcedure
        Dim prm_codigo As SqlParameter = New
        SqlParameter("@mccodigo", SqlDbType.Int)
        Dim prm_nombre As SqlParameter = New
        SqlParameter("@mcnombre", SqlDbType.VarChar, 40)
        prm_codigo.Value = codigo
        prm_nombre.Direction = ParameterDirection.Output
        With comando.Parameters
            .Add(prm_codigo)
            .Add(prm_nombre)
        End With
        Dim MercadoClienteDetalle As MercadoClienteDetalle = New
        MercadoClienteDetalle
        Try
            Conn.Open()
            comando.ExecuteNonQuery()
            Conn.Close()
            With MercadoClienteDetalle
                .nombre = prm_nombre.Value
            End With
```

```
        Catch ex As Exception
            With MercadoClienteDetalle
                .nombre = ""
            End With
        End Try
        Return MercadoClienteDetalle
    End Function
    Public Function ModificarMercadoCliente(ByVal codigo As Integer, ByVal
    nombre As String) As Integer
        Dim Conn As SqlConnection = New
        SqlConnection(ConfigurationSettings.AppSettings("Conexion"))
        Dim comando As SqlCommand = New
        SqlCommand("Sp_ModificarMercadoCliente", Conn)
        comando.CommandType = CommandType.StoredProcedure
        Dim prm_codigo As SqlParameter = New
        SqlParameter("@mccodigo", SqlDbType.Int)
        Dim prm_nombre As SqlParameter = New
        SqlParameter("@mcnombre", SqlDbType.VarChar, 40)
        prm_codigo.Value = código
        prm_nombre.Value = nombre
        With comando.Parameters
            .Add(prm_codigo)
            .Add(prm_nombre)
        End With
        Try
            Conn.Open()
            Dim modificados = comando.ExecuteNonQuery()
            Conn.Close()
            Return modificados
        Catch ex As Exception
            Return 0
        End Try
    End Function
```

```
 Public Function EliminarMercadoCliente(ByVal codigo As Integer) As
Integer
        Dim Conn As SqlConnection = New
        SqlConnection(ConfigurationSettings.AppSettings("Conexion"))
        Dim comando As SqlCommand = New
        SqlCommand("Sp_EliminarMercadoCliente", Conn)
        comando.CommandType = CommandType.StoredProcedure
        Dim prm_codigo As SqlParameter = New
        SqlParameter("@mccodigo", SqlDbType.Int)
        prm_codigo.Value = código
        comando.Parameters.Add(prm_codigo)
        Try
            Conn.Open()
            Dim eliminados = comando.ExecuteNonQuery
            Conn.Close()
            Return eliminados
        Catch ex As Exception
            Return 0
        End Try
    End Function
End Class
```

Orden.

```
Public Class OrdenDB
    Public Function ColocarCompra(ByVal clcodigo As Integer) As DataSet
        Dim Conn As SqlConnection = New
        SqlConnection(ConfigurationSettings.AppSettings("Conexion"))
        Dim da As SqlDataAdapter = New
        SqlDataAdapter("Sp_FinalizarCompra", Conn)
        Dim prm_clcodigo As New SqlParameter("@clcodigo",
        SqlDbType.Int)
        prm_clcodigo.Value = clcodigo
```

```
            da.SelectCommand.CommandType =
            CommandType.StoredProcedure
            da.SelectCommand.Parameters.Add(prm_clcodigo)
            Dim ds As DataSet = New DataSet
            Try
                da.Fill(ds, "FinalizarCompra")
                Return ds
            Catch ex As Exception
            End Try
    End Function
    Public Function TraerOrdenDetalle(ByVal orcodigo As Integer, ByVal
    clcodigo As Integer) As OrdenDetalle
            Dim conn As New
            SqlConnection(ConfigurationSettings.AppSettings("conexion"))
            Dim comando As New SqlDataAdapter("Sp_TraerOrdenDetalle",
            conn)
            comando.SelectCommand.CommandType =
            CommandType.StoredProcedure
            Dim prm_orcodigo As New SqlParameter("@orcodigo",
            SqlDbType.Int)
            Dim prm_clcodigo As New SqlParameter("@clcodigo",
            SqlDbType.Int)
            Dim prm_orfecha As New SqlParameter("@orfecha",
            SqlDbType.DateTime)
            Dim prm_orfechasalidafinca As New
            SqlParameter("@orfechasalidafinca", SqlDbType.DateTime)
            Dim prm_orfechavuelo As New SqlParameter("@orfechavuelo",
            SqlDbType.DateTime)
            Dim prm_orobservaciones As New
            SqlParameter("@orobservaciones", SqlDbType.VarChar, 100)
            Dim prm_total As New SqlParameter("@total", SqlDbType.Money)
            Dim prm_oraprobado As New SqlParameter("@oraprobado",
            SqlDbType.Bit)
            prm_orcodigo.Value = orcodigo
```

```
            prm_clcodigo.Value = clcodigo
            prm_orfecha.Direction = ParameterDirection.Output
            prm_orfechasalidafinca.Direction = ParameterDirection.Output
            prm_orfechavuelo.Direction = ParameterDirection.Output
            prm_orobservaciones.Direction = ParameterDirection.Output
            prm_total.Direction = ParameterDirection.Output
            prm_oraprobado.Direction = ParameterDirection.Output
            With comando.SelectCommand.Parameters
                .Add(prm_orcodigo)
                .Add(prm_clcodigo)
                .Add(prm_orfecha)
                .Add(prm_orfechasalidafinca)
                .Add(prm_orfechavuelo)
                .Add(prm_orobservaciones)
                .Add(prm_total)
                .Add(prm_oraprobado)
            End With
            Dim ds As New DataSet
            comando.Fill(ds, "OrdenDetalle")
            Dim OrdenDetalle As New Ecommerce.OrdenDetalle
            With OrdenDetalle
                .orfecha = prm_orfecha.Value
                .orfechasalidafinca = prm_orfechasalidafinca.Value
                .orfechavuelo = prm_orfechavuelo.Value
                .orobservaciones = prm_orobservaciones.Value
                .total = prm_total.Value
                .oraprobado = prm_oraprobado.Value
                .items = ds
            End With
            Return OrdenDetalle
        End Function
        Public Function ChequearOrdenCliente(ByVal orcodigo As Integer, ByVal
        clcodigo As Integer) As Integer
```

```
        Dim conn As New
        SqlConnection(ConfigurationSettings.AppSettings("conexion"))
        Dim comando As New SqlCommand("Sp_ChequearOrdenCliente",
        conn)
        comando.CommandType = CommandType.StoredProcedure
        Dim prm_clcodigo As New SqlParameter("@clcodigo",
        SqlDbType.Int)
        Dim prm_orcodigo As New SqlParameter("@orcodigo",
        SqlDbType.Int)
        Dim prm_existe As New SqlParameter("@existe", SqlDbType.Int)
        prm_clcodigo.Value = clcodigo
        prm_orcodigo.Value = orcodigo
        prm_existe.Direction = ParameterDirection.Output
        With comando.Parameters
            .Add(prm_clcodigo)
            .Add(prm_orcodigo)
            .Add(prm_existe)
        End With
        Try
            conn.Open()
            comando.ExecuteNonQuery()
            conn.Close()
            Return prm_existe.Value
        Catch ex As Exception
            Return 0
        End Try
    End Function
    Public Function TraerOrdenesNuevas(ByVal uscodigo As String)
        Dim Conn As SqlConnection = New
        SqlConnection(ConfigurationSettings.AppSettings("Conexion"))
        Dim da As SqlDataAdapter = New
        SqlDataAdapter("Sp_TraerOrdenesNuevas", Conn)
        da.SelectCommand.CommandType =
        CommandType.StoredProcedure
```

```
        Dim prm_uscodigo As New SqlParameter("@uscodigo",
        SqlDbType.VarChar, 10)
        prm_uscodigo.Value = uscodigo
        da.SelectCommand.Parameters.Add(prm_uscodigo)
        Dim ds As DataSet = New DataSet
        Try
            da.Fill(ds, "OrdenesNuevas")
            Return ds
        Catch ex As Exception
        End Try
End Function
Public Function ActualizarOrden(ByVal orcodigo As Integer, ByVal
orfechasalidafinca As DateTime, ByVal orfechavuelo As DateTime, ByVal
orobservaciones As String, ByVal oraprobado As Boolean) As Integer
        Dim Conn As SqlConnection = New
        SqlConnection(ConfigurationSettings.AppSettings("Conexion"))
        Dim comando As New SqlCommand("Sp_ActualizarOrden", Conn)
        comando.CommandType = CommandType.StoredProcedure
        Dim prm_orcodigo As New SqlParameter("@orcodigo",
        SqlDbType.Int)
        Dim prm_orfechasalidafinca As New
        SqlParameter("@orfechasalidafinca", SqlDbType.DateTime)
        Dim prm_orfechavuelo As New SqlParameter("@orfechavuelo",
        SqlDbType.DateTime)
        Dim prm_orobservaciones As New
        SqlParameter("@orobservaciones", SqlDbType.VarChar, 100)
        Dim prm_oraprobado As New SqlParameter("@oraprobado",
        SqlDbType.Bit)
        prm_orcodigo.Value = orcodigo
        prm_orfechasalidafinca.Value = orfechasalidafinca
        prm_orfechavuelo.Value = orfechavuelo
        prm_orobservaciones.Value = orobservaciones
        prm_oraprobado.Value = oraprobado
        With comando.Parameters
```

```
                .Add(prm_orcodigo)
                .Add(prm_orfechasalidafinca)
                .Add(prm_orfechavuelo)
                .Add(prm_orobservaciones)
                .Add(prm_oraprobado)
        End With
        Try
                Conn.Open()
                Dim r = comando.ExecuteNonQuery()
                Conn.Close()
                Return r
        Catch ex As Exception
                Return 0
        End Try
End Function
Public Function ActualizarOrdenDetalle(ByVal orcodigo As Integer, ByVal
prcodigo As Integer, ByVal ordcantidad As Integer, ByVal ordmarcacion As
String, ByVal accodigo As Integer, ByVal ordcomentario As String, ByVal
prprecio As Single) As Integer
        Dim Conn As SqlConnection = New
        SqlConnection(ConfigurationSettings.AppSettings("Conexion"))
        Dim comando As New SqlCommand("Sp_ActualizarOrdenDetalle",
        Conn)
        comando.CommandType = CommandType.StoredProcedure
         Dim prm_orcodigo As New SqlParameter("@orcodigo",
        SqlDbType.Int)
        Dim prm_prcodigo As New SqlParameter("@prcodigo",
        SqlDbType.Int)
        Dim prm_ordcantidad As New SqlParameter("@ordcantidad",
        SqlDbType.SmallInt)
        Dim prm_ordmarcacion As New SqlParameter("@ordmarcacion",
        SqlDbType.VarChar, 50)
        Dim prm_accodigo As New SqlParameter("@accodigo",
        SqlDbType.Int)
```

```
        Dim prm_ordcomentario As New SqlParameter("@ordcomentario",
        SqlDbType.VarChar, 100)
        Dim prm_prprecio As New SqlParameter("@prprecio",
        SqlDbType.Money)
        prm_orcodigo.Value = orcodigo
        prm_prcodigo.Value = prcodigo
        prm_ordcantidad.Value = ordcantidad
        prm_ordmarcacion.Value = ordmarcacion
        prm_accodigo.Value = accodigo
        prm_ordcomentario.Value = ordcomentario
        prm_prprecio.Value = prprecio
        With comando.Parameters
            .Add(prm_orcodigo)
            .Add(prm_prcodigo)
            .Add(prm_ordcantidad)
            .Add(prm_ordmarcacion)
            .Add(prm_accodigo)
            .Add(prm_ordcomentario)
            .Add(prm_prprecio)
        End With
        Try
            Conn.Open()
            Dim r As Integer = comando.ExecuteNonQuery
            Conn.Close()
            Return r
        Catch ex As Exception
            Return 0
        End Try
    End Function
    Public Function EliminarProductoPedido(ByVal orcodigo As Integer, ByVal
    prcodigo As Integer) As Integer
        Dim Conn As SqlConnection = New
        SqlConnection(ConfigurationSettings.AppSettings("Conexion"))
```

```
        Dim comando As New SqlCommand("Sp_EliminarProductoPedido",
        Conn)
        comando.CommandType = CommandType.StoredProcedure
        Dim prm_orcodigo As New SqlParameter("@orcodigo",
        SqlDbType.Int)
        Dim prm_prcodigo As New SqlParameter("@prcodigo",
        SqlDbType.Int)
        prm_orcodigo.Value = orcodigo
        prm_prcodigo.Value = prcodigo
        With comando.Parameters
            .Add(prm_orcodigo)
            .Add(prm_prcodigo)
        End With
        Try
            Conn.Open()
            Dim r As Integer = comando.ExecuteNonQuery
            Conn.Close()
            Return r
        Catch ex As Exception
            Return 0
        End Try
    End Function
    Public Function CambiarMarcaOrden(ByVal orcodigo As Integer, ByVal
    valor As Boolean) As Integer
        Dim Conn As SqlConnection = New
        SqlConnection(ConfigurationSettings.AppSettings("Conexion"))
        Dim comando As New SqlCommand("Sp_CambiarMarcaOrden",
        Conn)
        comando.CommandType = CommandType.StoredProcedure
        Dim prm_orcodigo As New SqlParameter("@orcodigo",
        SqlDbType.Int)
        Dim prm_valor As New SqlParameter("@valor", SqlDbType.Bit)
        prm_orcodigo.Value = orcodigo
        prm_valor.Value = valor
```

```
            With comando.Parameters
                .Add(prm_orcodigo)
                .Add(prm_valor)
            End With
            Try
                Conn.Open()
                Dim r = comando.ExecuteNonQuery
                Conn.Close()
                Return r
            Catch ex As Exception
                Return 0
            End Try
    End Function
    Public Function TraerListaOrdenes(ByVal uscodigo As String)
            Dim Conn As SqlConnection = New
            SqlConnection(ConfigurationSettings.AppSettings("Conexion"))
            Dim da As SqlDataAdapter = New
            SqlDataAdapter("Sp_TraerListaOrdenes", Conn)
            da.SelectCommand.CommandType =
            CommandType.StoredProcedure
            Dim prm_uscodigo As New SqlParameter("@uscodigo",
            SqlDbType.VarChar, 10)
            prm_uscodigo.Value = uscodigo
            da.SelectCommand.Parameters.Add(prm_uscodigo)
            Dim ds As DataSet = New DataSet
            Try
                da.Fill(ds, "ListaOrdenes")
                Return ds
            Catch ex As Exception
            End Try
    End Function
End Class
```

Producto.

```
Public Class ProductoDB
    Public Function TraerProductoDisponibilidad(ByVal vpcodigo As Integer)
    As DataSet
        Dim Conn As SqlConnection = New
        SqlConnection(ConfigurationSettings.AppSettings("Conexion"))
        Dim da As SqlDataAdapter = New
        SqlDataAdapter("Sp_TraerProductoDisponibilidad", Conn)
        da.SelectCommand.CommandType =
        CommandType.StoredProcedure
        Dim prm_vpcodigo As New SqlParameter("@vpcodigo",
        SqlDbType.Int)
        prm_vpcodigo.Value = vpcodigo
        da.SelectCommand.Parameters.Add(prm_vpcodigo)
        Dim ds As DataSet = New DataSet
        da.Fill(ds, "ProductoDisponibilidad")
        Return ds
    End Function
    Public Function TraerCategoriaProductoDetalle(ByVal cpcodigo) As
    CategoriaProductoDetalle
        Dim Conn As SqlConnection = New
        SqlConnection(ConfigurationSettings.AppSettings("Conexion"))
        Dim comando As SqlCommand = New
        SqlCommand("Sp_TraerCategoriaProductoDetalle", Conn)
        comando.CommandType = CommandType.StoredProcedure
        Dim prm_cpcodigo As SqlParameter = New
        SqlParameter("@cpcodigo", SqlDbType.Int)
        Dim prm_cpnombre As SqlParameter = New
        SqlParameter("@cpnombre", SqlDbType.VarChar, 40)
        prm_cpcodigo.Value = cpcodigo
        prm_cpnombre.Direction = ParameterDirection.Output
        With comando.Parameters
            .Add(prm_cpcodigo)
            .Add(prm_cpnombre)
```

```
        End With
        Dim CategoriaProductoDetalle As CategoriaProductoDetalle = New
        CategoriaProductoDetalle
        Try
            Conn.Open()
            comando.ExecuteNonQuery()
            Conn.Close()
            With CategoriaProductoDetalle
                .nombre = prm_cpnombre.Value
            End With
        Catch ex As Exception
            With CategoriaProductoDetalle
                .nombre = ""
            End With
        End Try
        Return CategoriaProductoDetalle
    End Function
    Public Function TraerVariedadProductoDetalle(ByVal vpcodigo) As
    VariedadProductoDetalle
        Dim Conn As SqlConnection = New
        SqlConnection(ConfigurationSettings.AppSettings("Conexion"))
        Dim comando As SqlCommand = New
        SqlCommand("Sp_TraerVariedadProductoDetalle", Conn)
        comando.CommandType = CommandType.StoredProcedure
        Dim prm_vpcodigo As SqlParameter = New
        SqlParameter("@vpcodigo", SqlDbType.Int)
        Dim prm_vpsigla As SqlParameter = New SqlParameter("@vpsigla",
        SqlDbType.VarChar, 10)
        Dim prm_vpnombre As SqlParameter = New
        SqlParameter("@vpnombre", SqlDbType.VarChar, 50)
        Dim prm_vpdescripcion As SqlParameter = New
        SqlParameter("@vpdescripcion", SqlDbType.VarChar, 255)
        Dim prm_vptamcabeza As SqlParameter = New
        SqlParameter("@vptamcabeza", SqlDbType.Int)
```

```
Dim prm_cpcodigo As SqlParameter = New
SqlParameter("@cpcodigo", SqlDbType.Int)
Dim prm_vpdisponible As SqlParameter = New
SqlParameter("@vpdisponible", SqlDbType.Bit)
Dim prm_foto As SqlParameter = New SqlParameter("@vpimagen",
SqlDbType.VarChar, 255)
prm_vpcodigo.Value = vpcodigo
prm_vpsigla.Direction = ParameterDirection.Output
prm_vpnombre.Direction = ParameterDirection.Output
prm_vpdescripcion.Direction = ParameterDirection.Output
prm_vptamcabeza.Direction = ParameterDirection.Output
prm_cpcodigo.Direction = ParameterDirection.Output
prm_vpdisponible.Direction = ParameterDirection.Output
prm_foto.Direction = ParameterDirection.Output
With comando.Parameters
        .Add(prm_vpcodigo)
        .Add(prm_vpsigla)
        .Add(prm_vpnombre)
        .Add(prm_vpdescripcion)
        .Add(prm_vptamcabeza)
        .Add(prm_cpcodigo)
         .Add(prm_vpdisponible)
        .Add(prm_foto)
End With
Dim VariedadProductoDetalle As VariedadProductoDetalle = New
VariedadProductoDetalle
Try
        Conn.Open()
        comando.ExecuteNonQuery()
        Conn.Close()
        With VariedadProductoDetalle
                .vpsigla = prm_vpsigla.Value
                .vpnombre = prm_vpnombre.Value
                .vpdescripcion = prm_vpdescripcion.Value
```

```
                .cpcodigo = prm_cpcodigo.Value
                .vptamcabeza = prm_vptamcabeza.Value
                .vpimagen = ""
            End With
        Catch ex As Exception
        End Try
    End Function
    Public Function TraerVariedadades() As SqlDataReader
        Dim conn As SqlConnection = New
        SqlConnection(ConfigurationSettings.AppSettings("conexion"))
        Dim comando As SqlCommand = New
        SqlCommand("Sp_TraerVariedades", conn)
        comando.CommandType = CommandType.StoredProcedure
        conn.Open()
        Dim variedades As SqlDataReader =
        comando.ExecuteReader(CommandBehavior.CloseConnection)
        Return variedades
    End Function
    Public Function InsertarProducto(ByVal tallo As Integer, ByVal empaque As
    Integer, ByVal variedad As Integer, ByVal promocion As Boolean) As
    Integer
        Dim conn As SqlConnection = New
        SqlConnection(ConfigurationSettings.AppSettings("conexion"))
        Dim comando As SqlCommand = New
        SqlCommand("Sp_InsertarProducto", conn)
        comando.CommandType = CommandType.StoredProcedure
        Dim prm_tallo As SqlParameter = New SqlParameter("@prtallo",
        SqlDbType.Int)
        Dim prm_empaque As SqlParameter = New
        SqlParameter("@prempaquexbunch", SqlDbType.Int)
        Dim prm_variedad As SqlParameter = New
        SqlParameter("@vpcodigo", SqlDbType.Int)
        Dim prm_promocion As SqlParameter = New
        SqlParameter("@prpromocion", SqlDbType.Bit)
```

```
            prm_tallo.Value = tallo
            prm_empaque.Value = empaque
            prm_variedad.Value = variedad
            prm_promocion.Value = promocion
            With comando.Parameters
                .Add(prm_tallo)
                .Add(prm_empaque)
                .Add(prm_variedad)
                .Add(prm_promocion)
            End With
            Try
                conn.Open()
                Dim insertados = comando.ExecuteNonQuery
                conn.Close()
                Return insertados
            Catch ex As Exception
                Return 0
            End Try
        End Function
        Public Function TraerListadoProducto() As DataSet
            Dim Conn As SqlConnection = New
            SqlConnection(ConfigurationSettings.AppSettings("Conexion"))
            Dim da As SqlDataAdapter = New
            SqlDataAdapter("sp_TraerListadoProducto", Conn)
            da.SelectCommand.CommandType =
            CommandType.StoredProcedure
            Dim ds As DataSet = New DataSet
            da.Fill(ds, "ListadoProducto")
            Return ds
        End Function
        Public Function TraerProductoDetalle(ByVal codigo As Integer) As
        ProductoDetalle
            Dim conn As SqlConnection = New
            SqlConnection(ConfigurationSettings.AppSettings("conexion"))
```

```
Dim comando As SqlCommand = New
SqlCommand("Sp_TraerProductoDetalle", conn)
comando.CommandType = CommandType.StoredProcedure
Dim prm_codigo As SqlParameter = New SqlParameter("@prcodigo",
SqlDbType.Int)
Dim prm_tallo As SqlParameter = New SqlParameter("@prtallo",
SqlDbType.Int)
Dim prm_empaque As SqlParameter = New
SqlParameter("@prempaquexbunch", SqlDbType.Int)
Dim prm_variedad As SqlParameter = New
SqlParameter("@vpcodigo", SqlDbType.Int)
Dim prm_promocion As SqlParameter = New
SqlParameter("@prpromocion", SqlDbType.Bit)
prm_codigo.Value = codigo
prm_tallo.Direction = ParameterDirection.Output
prm_empaque.Direction = ParameterDirection.Output
prm_variedad.Direction = ParameterDirection.Output
prm_promocion.Direction = ParameterDirection.Output
With comando.Parameters
        .Add(prm_codigo)
        .Add(prm_tallo)
        .Add(prm_empaque)
        .Add(prm_variedad)
        .Add(prm_promocion)
End With
Dim ProductoDetalle As ProductoDetalle = New ProductoDetalle
Try
        conn.Open()
        comando.ExecuteNonQuery()
        conn.Close()
        With ProductoDetalle
                .prtallo = prm_tallo.Value
                .prempaquexbunch = prm_empaque.Value
                .vpcodigo = prm_variedad.Value
```

```
                    .prpromocion = prm_promocion.Value
                End With
            Catch ex As Exception
                With ProductoDetalle
                    .prtallo = 0
                    .prempaquexbunch = 0
                    .vpcodigo = 0
                    .prpromocion = False
                End With
            End Try
            Return ProductoDetalle
        End Function
        Public Function ModificarProducto(ByVal codigo As Integer, ByVal tallo As
        Integer, ByVal empaque As Integer, ByVal variedad As Integer, ByVal
        promocion As Boolean) As Integer
            Dim conn As SqlConnection = New
            SqlConnection(ConfigurationSettings.AppSettings("conexion"))
            Dim comando As SqlCommand = New
            SqlCommand("Sp_ModificarProducto", conn)
            comando.CommandType = CommandType.StoredProcedure
            Dim prm_codigo As SqlParameter = New SqlParameter("@prcodigo",
            SqlDbType.Int)
            Dim prm_tallo As SqlParameter = New SqlParameter("@prtallo",
            SqlDbType.Int)
            Dim prm_empaque As SqlParameter = New
            SqlParameter("@prempaquexbunch", SqlDbType.Int)
            Dim prm_variedad As SqlParameter = New
            SqlParameter("@vpcodigo", SqlDbType.Int)
            Dim prm_promocion As SqlParameter = New
            SqlParameter("@prpromocion", SqlDbType.Bit)
            prm_codigo.Value = código
            prm_tallo.Value = tallo
            prm_empaque.Value = empaque
            prm_variedad.Value = variedad
```

```
            prm_promocion.Value = promocion
            With comando.Parameters
                .Add(prm_codigo)
                .Add(prm_tallo)
                .Add(prm_empaque)
                .Add(prm_variedad)
                .Add(prm_promocion)
            End With
            Try
                conn.Open()
                Dim modificados = comando.ExecuteNonQuery
                conn.Close()
                Return modificados
            Catch ex As Exception
                Return 0
            End Try
    End Function
    Public Function EliminarProducto(ByVal codigo As Integer) As Integer
            Dim conn As SqlConnection = New
            SqlConnection(ConfigurationSettings.AppSettings("conexion"))
            Dim comando As SqlCommand = New
            SqlCommand("Sp_EliminarProducto", conn)
            comando.CommandType = CommandType.StoredProcedure
            Dim prm_codigo As SqlParameter = New SqlParameter("@prcodigo",
            SqlDbType.Int)
            prm_codigo.Value = código
            comando.Parameters.Add(prm_codigo)
            Try
                conn.Open()
                Dim eliminados = comando.ExecuteNonQuery
                conn.Close()
                Return eliminados
            Catch ex As Exception
                Return 0
```

```
        End Try
    End Function
    Public Function TraerVariedadxCategoria(ByVal cpcodigo As Integer) As
    DataSet
        Dim Conn As SqlConnection = New
        SqlConnection(ConfigurationSettings.AppSettings("Conexion"))
        Dim da As SqlDataAdapter = New
        SqlDataAdapter("sp_TraerVariedadxCategoria", Conn)
        da.SelectCommand.CommandType =
        CommandType.StoredProcedure
        Dim prm_vpcodigo As SqlParameter = New
        SqlParameter("@cpcodigo", SqlDbType.Int)
        prm_vpcodigo.Value = cpcodigo
        da.SelectCommand.Parameters.Add(prm_vpcodigo)
        Dim ds As DataSet = New DataSet
        da.Fill(ds, "ListadoVariedad")
        Return ds
    End Function
    Public Function VerDetalleVariedad(ByVal codigo As String) As DataSet
        Dim Conn As SqlConnection = New
        SqlConnection(ConfigurationSettings.AppSettings("Conexion"))
        Dim da As SqlDataAdapter = New
        SqlDataAdapter("sp_VerDetalleVariedad", Conn)
        da.SelectCommand.CommandType =
        CommandType.StoredProcedure
        Dim prm_vpcodigo As SqlParameter = New
        SqlParameter("@vpcodigo", SqlDbType.Int)
        prm_vpcodigo.Value = codigo
        da.SelectCommand.Parameters.Add(prm_vpcodigo)
        Dim ds As DataSet = New DataSet
        da.Fill(ds, "ListadoVariedad")
        Return ds
    End Function
```

```
Public Function TraerTamanosProducto(ByVal vpcodigo As Integer) As
DataSet
        Dim Conn As SqlConnection = New
        SqlConnection(ConfigurationSettings.AppSettings("Conexion"))
        Dim da As SqlDataAdapter = New
        SqlDataAdapter("sp_TraerProductoxTamano", Conn)
        da.SelectCommand.CommandType =
        CommandType.StoredProcedure
        Dim prm_vpcodigo As SqlParameter = New
        SqlParameter("@vpcodigo", SqlDbType.Int)
        prm_vpcodigo.Value = vpcodigo
        da.SelectCommand.Parameters.Add(prm_vpcodigo)
        Dim ds As DataSet = New DataSet
        da.Fill(ds, "ListadoTamanos")
        Return ds
End Function
Public Function TraerListaCategoriaProducto() As DataSet
        Dim Conn As SqlConnection = New
        SqlConnection(ConfigurationSettings.AppSettings("Conexion"))
        Dim da As SqlDataAdapter = New
        SqlDataAdapter("sp_TraerCategoriaProducto", Conn)
        da.SelectCommand.CommandType =
        CommandType.StoredProcedure
        Dim ds As DataSet = New DataSet
        da.Fill(ds, "ListadoCategorias")
        Return ds
End Function
Public Function TraerVariedadesPromocion() As DataSet
        Dim Conn As SqlConnection = New
        SqlConnection(ConfigurationSettings.AppSettings("Conexion"))
        Dim da As SqlDataAdapter = New
        SqlDataAdapter("sp_TraerVariedadesPromocion", Conn)
        da.SelectCommand.CommandType =
        CommandType.StoredProcedure
```

```
            Dim ds As DataSet = New DataSet
            da.Fill(ds, "ListadoVariedadPromocion")
            Return ds
        End Function
        Public Function TraerProductosPromocion(ByVal vpcodigo As Integer)
            Dim Conn As SqlConnection = New
            SqlConnection(ConfigurationSettings.AppSettings("Conexion"))
            Dim da As SqlDataAdapter = New
            SqlDataAdapter("sp_TraerProductosPromocion", Conn)
            da.SelectCommand.CommandType =
            CommandType.StoredProcedure
            Dim prm_vpcodigo As New SqlParameter("@vpcodigo",
            SqlDbType.Int)
            prm_vpcodigo.Value = vpcodigo
            da.SelectCommand.Parameters.Add(prm_vpcodigo)
            Dim ds As DataSet = New DataSet
            da.Fill(ds, "ListadoProductosPromocion")
            Return ds
        End Function
End Class
```

Categoría Producto.

```
Public Class CategoriaProductoDB
        Public Function InsertarCategoriaProducto(ByVal nombre As String) As
        Integer
            Dim conn As SqlConnection = New
            SqlConnection(ConfigurationSettings.AppSettings("conexion"))
            Dim comando As SqlCommand = New
            SqlCommand("Sp_InsertarCategoriaProducto", conn)
            comando.CommandType = CommandType.StoredProcedure
            Dim prm_nombre As SqlParameter = New
            SqlParameter("@cpnombre", SqlDbType.VarChar, 40)
            prm_nombre.Value = nombre
```

```
        comando.Parameters.Add(prm_nombre)
        Try
            conn.Open()
            Dim insertados = comando.ExecuteNonQuery
             conn.Close()
            Return insertados
        Catch ex As Exception
            Return 0
        End Try
    End Function
    Public Function TraerCategoriaProductoDetalle(ByVal codigo As String) As
    CategoriaProductoDetalle
        Dim Conn As SqlConnection = New
        SqlConnection(ConfigurationSettings.AppSettings("Conexion"))
        Dim comando As SqlCommand = New
        SqlCommand("Sp_TraerCategoriaProductoDetalle", Conn)
        comando.CommandType = CommandType.StoredProcedure
        Dim prm_codigo As SqlParameter = New SqlParameter("@cpcodigo",
        SqlDbType.Int)
        Dim prm_nombre As SqlParameter = New
        SqlParameter("@cpnombre", SqlDbType.VarChar, 40)
        prm_codigo.Value = codigo
        prm_nombre.Direction = ParameterDirection.Output
        With comando.Parameters
            .Add(prm_codigo)
            .Add(prm_nombre)
        End With
        Dim CategoriaProductoDetalle As CategoriaProductoDetalle = New
        CategoriaProductoDetalle
        Try
            Conn.Open()
            comando.ExecuteNonQuery()
            Conn.Close()
            With CategoriaProductoDetalle
```

```
                .nombre = prm_nombre.Value
            End With
        Catch ex As Exception
            With CategoriaProductoDetalle
                .nombre = ""
            End With
        End Try
        Return CategoriaProductoDetalle
    End Function
    Public Function TraerListadoCategoriaProducto() As DataSet
        Dim Conn As SqlConnection = New
        SqlConnection(ConfigurationSettings.AppSettings("Conexion"))
        Dim da As SqlDataAdapter = New
        SqlDataAdapter("Sp_TraerListadoCategoriaProducto", Conn)
        da.SelectCommand.CommandType =
        CommandType.StoredProcedure
        Dim ds As DataSet = New DataSet
        da.Fill(ds, "ListadoCategoriaProducto")
        Return ds
    End Function
    Public Function ModificarCategoriaProducto(ByVal codigo As Integer,
    ByVal nombre As String) As Integer
        Dim Conn As SqlConnection = New
        SqlConnection(ConfigurationSettings.AppSettings("Conexion"))
        Dim comando As SqlCommand = New
        SqlCommand("Sp_ModificarCategoriaProducto", Conn)
        comando.CommandType = CommandType.StoredProcedure
        Dim prm_codigo As SqlParameter = New SqlParameter("@cpcodigo",
        SqlDbType.Int)
        Dim prm_nombre As SqlParameter = New
        SqlParameter("@cpnombre", SqlDbType.VarChar, 40)
        prm_codigo.Value = código
        prm_nombre.Value = nombre
        With comando.Parameters
```

```
                .Add(prm_codigo)
                .Add(prm_nombre)
            End With
            Try
                Conn.Open()
                Dim modificados = comando.ExecuteNonQuery()
                Conn.Close()
                Return modificados
            Catch ex As Exception
                Return 0
            End Try
        End Function
        Public Function EliminarCategoriaProducto(ByVal codigo As Integer) As
        Integer
            Dim conn As SqlConnection = New
            SqlConnection(ConfigurationSettings.AppSettings("conexion"))
            Dim comando As SqlCommand = New
            SqlCommand("Sp_EliminarCategoriaProducto", conn)
            comando.CommandType = CommandType.StoredProcedure
            Dim prm_codigo As SqlParameter = New SqlParameter("@cpcodigo",
            SqlDbType.Int)
            prm_codigo.Value = código
            comando.Parameters.Add(prm_codigo)
            Try
                conn.Open()
                Dim eliminados = comando.ExecuteNonQuery
                conn.Close()
                Return eliminados
            Catch ex As Exception
                Return 0
            End Try
        End Function
    End Class
```

Roles.

```
Public Class RolesDB
    Public Function TraerListadoRol() As DataSet
        Dim Conn As SqlConnection = New
        SqlConnection(ConfigurationSettings.AppSettings("Conexion"))
        Dim comando As SqlCommand = New
        SqlCommand("Sp_TraerListadoRoles", Conn)
        comando.CommandType = CommandType.StoredProcedure
        Dim da As SqlDataAdapter = New
        SqlDataAdapter("sp_TraerListadoRoles", Conn)
        da.SelectCommand.CommandType =
        CommandType.StoredProcedure
        Dim ds As DataSet = New DataSet
        da.Fill(ds, "ListadoRoles")
        Return ds
    End Function

    Public Function TraerRolDetalle(ByVal codigo As Integer) As RolDetalle
        Dim conn As SqlConnection = New
        SqlConnection(ConfigurationSettings.AppSettings("conexion"))
        Dim comando As SqlCommand = New
        SqlCommand("Sp_TraerRolDetalle", conn)
        comando.CommandType = CommandType.StoredProcedure
        Dim prm_codigo As SqlParameter = New SqlParameter("@rocodigo",
        SqlDbType.Int)
        Dim prm_nombre As SqlParameter = New
        SqlParameter("@ronombre", SqlDbType.VarChar, 100)
        prm_codigo.Value = codigo
        prm_nombre.Direction = ParameterDirection.Output
        With comando.Parameters
            .Add(prm_codigo)
            .Add(prm_nombre)
        End With
        Dim RolDetalle As RolDetalle = New RolDetalle
```

```
        Try
            conn.Open()
            comando.ExecuteNonQuery()
            conn.Close()
            With RolDetalle
                .prnombre = prm_nombre.Value
            End With
        Catch ex As Exception
            With RolDetalle
                .prnombre = ""
            End With
        End Try
        Return RolDetalle
    End Function
    Public Function TraerUsuarios() As SqlDataReader
        Dim conn As SqlConnection = New
        SqlConnection(ConfigurationSettings.AppSettings("conexion"))
        Dim comando As SqlCommand = New
        SqlCommand("Sp_TraerListadoUsuarios", conn)
        comando.CommandType = CommandType.StoredProcedure
        conn.Open()
        Dim usuarios As SqlDataReader =
        comando.ExecuteReader(CommandBehavior.CloseConnection)
        Return usuarios
    End Function
    Public Function TraerRoles() As SqlDataReader
        Dim conn As SqlConnection = New
        SqlConnection(ConfigurationSettings.AppSettings("conexion"))
        Dim comando As SqlCommand = New
        SqlCommand("Sp_TraerListadoRolesUsuario", conn)
        comando.CommandType = CommandType.StoredProcedure
        conn.Open()
        Dim roles As SqlDataReader =
        comando.ExecuteReader(CommandBehavior.CloseConnection)
```

```
        Return roles
    End Function
    Public Function InsertarRolesUsuario(ByVal usuario As String, ByVal rol As
    Integer) As Integer
        Dim conn As SqlConnection = New
        SqlConnection(ConfigurationSettings.AppSettings("conexion"))
        Dim comando As SqlCommand = New
        SqlCommand("Sp_AsignarRolUsuario", conn)
        comando.CommandType = CommandType.StoredProcedure
        Dim prm_usuario As SqlParameter = New SqlParameter("@uscedula",
        SqlDbType.VarChar, 10)
        Dim prm_rol As SqlParameter = New SqlParameter("@rocodigo",
        SqlDbType.Int)
        prm_usuario.Value = usuario
        prm_rol.Value = rol
        With comando.Parameters
            .Add(prm_usuario)
            .Add(prm_rol)
        End With
        Try
            conn.Open()
            Dim insertados = comando.ExecuteNonQuery
            conn.Close()
            Return insertados
        Catch ex As Exception
            Return 0
        End Try
    End Function
    Public Function TraerListadoUsuario() As DataSet
        Dim Conn As SqlConnection = New
        SqlConnection(ConfigurationSettings.AppSettings("Conexion"))
        Dim comando As SqlCommand = New
        SqlCommand("Sp_TraerListadoUsuarioRol", Conn)
        comando.CommandType = CommandType.StoredProcedure
```

```
            Dim da As SqlDataAdapter = New
            SqlDataAdapter("Sp_TraerListadoUsuarioRol", Conn)
            da.SelectCommand.CommandType =
            CommandType.StoredProcedure
            Dim ds As DataSet = New DataSet
            da.Fill(ds, "ListadoUsuarioRol")
            Return ds
        End Function
        Public Function TraerRolAsignado(ByVal cedula As String) As DataSet
            Dim Conn As SqlConnection = New
            SqlConnection(ConfigurationSettings.AppSettings("Conexion"))
            Dim prm_cedula As SqlParameter = New SqlParameter("@uscedula",
            SqlDbType.VarChar, 10)
            prm_cedula.Value = cedula
            Dim da As SqlDataAdapter = New
            SqlDataAdapter("Sp_TraerRolAsignado", Conn)
            da.SelectCommand.CommandType =
            CommandType.StoredProcedure
            da.SelectCommand.Parameters.Add(prm_cedula)
            Dim ds As DataSet = New DataSet
            da.Fill(ds, "ListadoRolAsignado")
            Return ds
        End Function
End Class
```

Seguridad.

```
Public Class SeguridadDB
        Public Function TraerRol()
            Return HttpContext.Current.User.IsInRole("Admins")
        End Function
        Public Function ChequearRol(ByVal uscodigo As String, ByVal rocodigo As
        Integer) As Boolean
```

```
            Dim conn As SqlConnection = New
            SqlConnection(ConfigurationSettings.AppSettings("conexion"))
            Dim comando As SqlCommand = New
            SqlCommand("Sp_ChequearRole", conn)
            comando.CommandType = CommandType.StoredProcedure
            Dim prm_uscodigo As New SqlParameter("@uscedula",
            SqlDbType.VarChar, 10)
            Dim prm_rocodigo As New SqlParameter("@rocodigo",
            SqlDbType.Int)
            Dim prm_estaenrol As New SqlParameter("@EstaEnRol",
            SqlDbType.Bit)
            With comando.Parameters
                .Add(prm_uscodigo)
                .Add(prm_rocodigo)
                .Add(prm_estaenrol)
            End With
            Try
                conn.Open()
                comando.ExecuteNonQuery()
                conn.Close()
                Return prm_estaenrol.Value
            Catch ex As Exception
                Return False
            End Try
        End Function
End Class
```

Suscripcion.

```
Public Class SuscripcionDB
        Public Function TraerListadoSuscripcion() As DataSet
            Dim Conn As SqlConnection = New
            SqlConnection(ConfigurationSettings.AppSettings("Conexion"))
```

```
        Dim comando As SqlCommand = New
        SqlCommand("Sp_TraerListadoSuscripcion", Conn)
        comando.CommandType = CommandType.StoredProcedure
        Dim da As SqlDataAdapter = New
        SqlDataAdapter("sp_TraerListadoSuscripcion", Conn)
        da.SelectCommand.CommandType =
        CommandType.StoredProcedure
        Dim ds As DataSet = New DataSet
        da.Fill(ds, "ListadoSuscripcion")
        Return ds
    End Function
    Public Function TraerSuscripcionDetalle(ByVal codigo As Integer) As
    SuscripcionDetalle
        Dim conn As SqlConnection = New
        SqlConnection(ConfigurationSettings.AppSettings("conexion"))
        Dim comando As SqlCommand = New
        SqlCommand("Sp_TraerSuscripcionDetalle", conn)
        comando.CommandType = CommandType.StoredProcedure
        Dim prm_codigo As SqlParameter = New SqlParameter("@sucodigo",
        SqlDbType.Int)
        Dim prm_mail As SqlParameter = New SqlParameter("@sumail",
        SqlDbType.VarChar, 255)
        Dim prm_activa As SqlParameter = New SqlParameter("@suactiva",
        SqlDbType.Bit)
        prm_codigo.Value = codigo
        prm_mail.Direction = ParameterDirection.Output
        prm_activa.Direction = ParameterDirection.Output
        With comando.Parameters
            .Add(prm_codigo)
            .Add(prm_mail)
            .Add(prm_activa)
        End With
        Dim SuscripcionDetalle As SuscripcionDetalle = New
        SuscripcionDetalle
```

```
        Try
            conn.Open()
            comando.ExecuteNonQuery()
            conn.Close()
            With SuscripcionDetalle
                .sumail = prm_mail.Value
                .suactiva = prm_activa.Value
            End With
        Catch ex As Exception
            With SuscripcionDetalle
                .sumail = ""
                .suactiva = False
            End With
        End Try
        Return SuscripcionDetalle
    End Function
    Public Function DesactivarSuscripcion(ByVal codigo As Integer, ByVal
    activa As Boolean) As Integer
        Dim Conn As SqlConnection = New
        SqlConnection(ConfigurationSettings.AppSettings("conexion"))
        Dim comando As SqlCommand = New
        SqlCommand("Sp_DesactivarSuscripcion", Conn)
        comando.CommandType = CommandType.StoredProcedure
        Dim prm_codigo As SqlParameter = New SqlParameter("@sucodigo",
        SqlDbType.Int)
        Dim prm_activa As SqlParameter = New SqlParameter("@suactiva",
        SqlDbType.Bit)
        prm_codigo.Value = código
        prm_activa.Value = activa
        With comando.Parameters
            .Add(prm_codigo)
            .Add(prm_activa)
        End With
        Try
```

```
                Conn.Open()
                Dim modificados = comando.ExecuteNonQuery
                Conn.Close()
                Return modificados
            Catch ex As Exception
                Return 0
            End Try
        End Function
End Class
```

Usuario.

```
Public Class UsuarioDB
    Public Function IngresoUsuario(ByVal cedula As String, ByVal nombre As
    String, ByVal apellido As String, ByVal nombreusuario As String, ByVal
    contrasena As String, ByVal email As String, ByVal teltrabajo As String,
    ByVal extension As String, ByVal telmovil As String, ByVal foto As String,
    ByVal activo As Boolean) As Integer
        Dim Conn As SqlConnection = New
        SqlConnection(ConfigurationSettings.AppSettings("Conexion"))
        Dim comando As SqlCommand = New
        SqlCommand("Sp_InsertarUsuario", Conn)
        comando.CommandType = CommandType.StoredProcedure
        Dim prm_cedula As SqlParameter = New SqlParameter("@uscedula",
        SqlDbType.VarChar, 10)
        Dim prm_nombre As SqlParameter = New
        SqlParameter("@usnombre", SqlDbType.VarChar, 40)
        Dim prm_apellido As SqlParameter = New
        SqlParameter("@usapellido", SqlDbType.VarChar, 40)
        Dim prm_nombreusuario As SqlParameter = New
        SqlParameter("@usnombreusuario", SqlDbType.VarChar, 40)
        Dim prm_contrasena As SqlParameter = New
        SqlParameter("@uscontrasena", SqlDbType.VarChar, 255)
```

```
Dim prm_email As SqlParameter = New SqlParameter("@usemail",
SqlDbType.VarChar, 100)
Dim prm_telefono As SqlParameter = New
SqlParameter("@ustelefono", SqlDbType.VarChar, 15)
Dim prm_extension As SqlParameter = New
SqlParameter("@usextension", SqlDbType.VarChar, 5)
Dim prm_movil As SqlParameter = New SqlParameter("@usmovil",
SqlDbType.VarChar, 15)
Dim prm_foto As SqlParameter = New SqlParameter("@usfoto",
SqlDbType.VarChar, 255)
Dim prm_activo As SqlParameter = New SqlParameter("@usactivo",
SqlDbType.Bit)
prm_cedula.Value = cedula
prm_nombre.Value = nombre
prm_apellido.Value = apellido
prm_nombreusuario.Value = nombreusuario
prm_contrasena.Value = contrasena
prm_email.Value = email
prm_telefono.Value = teltrabajo
prm_extension.Value = extension
prm_movil.Value = telmovil
prm_foto.Value = foto
prm_activo.Value = activo
With comando.Parameters
        .Add(prm_cedula)
        .Add(prm_nombre)
        .Add(prm_apellido)
        .Add(prm_nombreusuario)
        .Add(prm_contrasena)
        .Add(prm_email)
        .Add(prm_telefono)
        .Add(prm_extension)
        .Add(prm_movil)
        .Add(prm_foto)
```

```
                .Add(prm_activo)
        End With
        Try
                Conn.Open()
                Dim insertados = comando.ExecuteNonQuery()
                Conn.Close()
                Return insertados
        Catch ex As Exception
                Return 0
        End Try
End Function
Public Function TraerUsuarioDetalle(ByVal cedula As String) As
UsuarioDetalle
        Dim Conn As SqlConnection = New
        SqlConnection(ConfigurationSettings.AppSettings("Conexion"))
        Dim comando As SqlCommand = New
        SqlCommand("Sp_TraerUsuarioDetalle", Conn)
        comando.CommandType = CommandType.StoredProcedure
        Dim prm_cedula As SqlParameter = New SqlParameter("@uscedula",
        SqlDbType.VarChar, 10)
        Dim prm_nombre As SqlParameter = New
        SqlParameter("@usnombre", SqlDbType.VarChar, 40)
        Dim prm_apellido As SqlParameter = New
        SqlParameter("@usapellido", SqlDbType.VarChar, 40)
        Dim prm_nombreusuario As SqlParameter = New
        SqlParameter("@usnombreusuario", SqlDbType.VarChar, 40)
        Dim prm_contrasena As SqlParameter = New
        SqlParameter("@uscontrasena", SqlDbType.VarChar, 255)
        Dim prm_email As SqlParameter = New SqlParameter("@usemail",
        SqlDbType.VarChar, 100)
        Dim prm_telefono As SqlParameter = New
        SqlParameter("@ustelefono", SqlDbType.VarChar, 15)
        Dim prm_extension As SqlParameter = New
        SqlParameter("@usextension", SqlDbType.VarChar, 5)
```

```
Dim prm_movil As SqlParameter = New
SqlParameter("@ustelfmovil", SqlDbType.VarChar, 15)
Dim prm_foto As SqlParameter = New SqlParameter("@usfoto",
SqlDbType.VarChar, 255)
Dim prm_activo As SqlParameter = New SqlParameter("@usactivo",
SqlDbType.Bit)
prm_cedula.Value = cedula
prm_nombre.Direction = ParameterDirection.Output
prm_apellido.Direction = ParameterDirection.Output
prm_nombreusuario.Direction = ParameterDirection.Output
prm_contrasena.Direction = ParameterDirection.Output
prm_email.Direction = ParameterDirection.Output
prm_telefono.Direction = ParameterDirection.Output
prm_extension.Direction = ParameterDirection.Output
prm_movil.Direction = ParameterDirection.Output
prm_foto.Direction = ParameterDirection.Output
prm_activo.Direction = ParameterDirection.Output
With comando.Parameters
        .Add(prm_cedula)
        .Add(prm_nombre)
        .Add(prm_apellido)
        .Add(prm_nombreusuario)
        .Add(prm_contrasena)
        .Add(prm_email)
        .Add(prm_telefono)
        .Add(prm_extension)
        .Add(prm_movil)
        .Add(prm_foto)
        .Add(prm_activo)
End With
Dim UsuarioDetalle As UsuarioDetalle = New UsuarioDetalle
Try
        Conn.Open()
        comando.ExecuteNonQuery()
```

```
                Conn.Close()
                With UsuarioDetalle
                    .nombre = prm_nombre.Value
                    .apellido = prm_apellido.Value
                    .nombreusuario = prm_nombreusuario.Value
                    .contrasena = prm_contrasena.Value
                    .email = prm_email.Value
                    .telefono = prm_telefono.Value
                    .extension = prm_extension.Value
                    .movil = prm_movil.Value
                    .foto = prm_foto.Value
                    .activo = prm_activo.Value
                End With
            Catch ex As Exception
                With UsuarioDetalle
                    .nombre = ""
                    .apellido = ""
                    .nombreusuario = ""
                    .contrasena = ""
                    .email = ""
                    .telefono = ""
                    .extension = ""
                    .movil = ""
                    .foto = ""
                    .activo = False
                End With
            End Try
            Return UsuarioDetalle
        End Function
        Public Function ModificarUsuario(ByVal cedula As String, ByVal nombre
        As String, ByVal apellido As String, ByVal nombreusuario As String, ByVal
        contrasena As String, ByVal email As String, ByVal teltrabajo As String,
        ByVal extension As String, ByVal telmovil As String, ByVal foto As String,
        ByVal activo As Boolean) As Integer
```

```
Dim Conn As SqlConnection = New
SqlConnection(ConfigurationSettings.AppSettings("Conexion"))
Dim comando As SqlCommand = New
SqlCommand("Sp_ModificarUsuario", Conn)
comando.CommandType = CommandType.StoredProcedure
Dim prm_cedula As SqlParameter = New SqlParameter("@uscedula",
SqlDbType.VarChar, 10)
Dim prm_nombre As SqlParameter = New
SqlParameter("@usnombre", SqlDbType.VarChar, 40)
Dim prm_apellido As SqlParameter = New
SqlParameter("@usapellido", SqlDbType.VarChar, 40)
Dim prm_nombreusuario As SqlParameter = New
SqlParameter("@usnombreusuario", SqlDbType.VarChar, 40)
Dim prm_contrasena As SqlParameter = New
SqlParameter("@uscontrasena", SqlDbType.VarChar, 255)
Dim prm_email As SqlParameter = New SqlParameter("@usemail",
SqlDbType.VarChar, 100)
Dim prm_telefono As SqlParameter = New
SqlParameter("@ustelefono", SqlDbType.VarChar, 15)
Dim prm_extension As SqlParameter = New
SqlParameter("@usextension", SqlDbType.VarChar, 5)
Dim prm_movil As SqlParameter = New
SqlParameter("@ustelfmovil", SqlDbType.VarChar, 15)
Dim prm_foto As SqlParameter = New SqlParameter("@usfoto",
SqlDbType.VarChar, 255)
Dim prm_activo As SqlParameter = New SqlParameter("@usactivo",
SqlDbType.Bit)
prm_cedula.Value = cedula
prm_nombre.Value = nombre
prm_apellido.Value = apellido
prm_nombreusuario.Value = nombreusuario
prm_contrasena.Value = contrasena
prm_email.Value = email
prm_telefono.Value = teltrabajo
```

```
            prm_extension.Value = extension
            prm_movil.Value = telmovil
            prm_foto.Value = foto
            prm_activo.Value = activo
            With comando.Parameters
                .Add(prm_cedula)
                .Add(prm_nombre)
                .Add(prm_apellido)
                .Add(prm_nombreusuario)
                .Add(prm_contrasena)
                .Add(prm_email)
                .Add(prm_telefono)
                .Add(prm_extension)
                .Add(prm_movil)
                .Add(prm_foto)
                .Add(prm_activo)
            End With
            Try
                Conn.Open()
                Dim insertados = comando.ExecuteNonQuery()
                Conn.Close()
                Return insertados
            Catch ex As Exception
                Return 0
            End Try
        End Function
        Public Function EliminarUsuario(ByVal cedula As String) As Integer
            Dim Conn As SqlConnection = New
            SqlConnection(ConfigurationSettings.AppSettings("Conexion"))
            Dim comando As SqlCommand = New
            SqlCommand("Sp_EliminarUsuario", Conn)
            comando.CommandType = CommandType.StoredProcedure
            Dim prm_cedula As SqlParameter = New SqlParameter("@uscedula",
            SqlDbType.VarChar, 10)
```

```
            prm_cedula.Value = cedula
            comando.Parameters.Add(prm_cedula)
            Try
                Conn.Open()
                Dim eliminados = comando.ExecuteNonQuery
                Conn.Close()
                Return eliminados
            Catch ex As Exception
                Return 0
            End Try
        End Function
        Public Function TraerListadoUsuario() As DataSet
            Dim Conn As SqlConnection = New
            SqlConnection(ConfigurationSettings.AppSettings("Conexion"))
            Dim da As SqlDataAdapter = New
            SqlDataAdapter("sp_TraerListadoUsuario", Conn)
            da.SelectCommand.CommandType =
            CommandType.StoredProcedure
            Dim ds As DataSet = New DataSet
            da.Fill(ds, "ListadoUsuario")
            Return ds
        End Function
        Public Function TraerRolesUsuario(ByVal uscodigo As String) As String()
            Dim Conn As SqlConnection = New
            SqlConnection(ConfigurationSettings.AppSettings("Conexion"))
            Dim comando As SqlCommand = New
            SqlCommand("Sp_TraerRolesUsuario", Conn)
            comando.CommandType = CommandType.StoredProcedure
            Dim prm_uscodigo As New SqlParameter("@uscodigo",
            SqlDbType.VarChar, 10)
            prm_uscodigo.Value = uscodigo
            comando.Parameters.Add(prm_uscodigo)
            Dim dr As SqlDataReader
            Conn.Open()
```

```
            dr = comando.ExecuteReader(CommandBehavior.CloseConnection)
            Dim RolesUsuario As New ArrayList
            While dr.Read()
                RolesUsuario.Add(dr("ronombre"))
            End While
            dr.Close()
            Return CType(RolesUsuario.ToArray(GetType(String)), String())
        End Function
        Public Function LoginUsuario(ByVal nombreusuario As String, ByVal
        contrasena As String) As String
            Dim Conn As SqlConnection = New
            SqlConnection(ConfigurationSettings.AppSettings("Conexion"))
            Dim comando As SqlCommand = New
            SqlCommand("Sp_LoginUsuario", Conn)
            comando.CommandType = CommandType.StoredProcedure
            Dim prm_nombreusuario As New SqlParameter("@usnombreusuario",
            SqlDbType.VarChar, 40)
            Dim prm_contrasena As New SqlParameter("@uscontrasena",
            SqlDbType.VarChar, 255)
            Dim prm_uscodigo As New SqlParameter("@uscodigo",
            SqlDbType.VarChar, 10)
            prm_nombreusuario.Value = nombreusuario
            prm_contrasena.Value = contrasena
            prm_uscodigo.Direction = ParameterDirection.Output
            With comando.Parameters
                .Add(prm_nombreusuario)
                .Add(prm_contrasena)
                .Add(prm_uscodigo)
            End With
            Try
                Conn.Open()
                comando.ExecuteNonQuery()
                Conn.Close()
                Return prm_uscodigo.Value
```

```
            Catch ex As Exception
                Return Nothing
            End Try
        End Function
End Class
```

Variedad Producto.

```
Public Class VariedadProductoDB
    Public Function InsertarVariedadProducto(ByVal sigla As String, ByVal
    nombre As String, ByVal descripcion As String, ByVal tamcabeza As Integer,
    ByVal categoria As Integer, ByVal disponible As Boolean, ByVal foto As
    String) As Integer
        Dim conn As SqlConnection = New
        SqlConnection(ConfigurationSettings.AppSettings("conexion"))
        Dim comando As SqlCommand = New
        SqlCommand("Sp_InsertarVariedadProducto", conn)
        comando.CommandType = CommandType.StoredProcedure
        Dim prm_sigla As SqlParameter = New SqlParameter("@vpsigla",
        SqlDbType.VarChar, 10)
        Dim prm_nombre As SqlParameter = New
        SqlParameter("@vpnombre", SqlDbType.VarChar, 50)
        Dim prm_descripcion As SqlParameter = New
        SqlParameter("@vpdescripcion", SqlDbType.VarChar, 255)
        Dim prm_tamcabeza As SqlParameter = New
        SqlParameter("@vptamcabeza", SqlDbType.SmallInt)
        Dim prm_categoria As SqlParameter = New
        SqlParameter("@cpcodigo", SqlDbType.Int)
        Dim prm_disponible As SqlParameter = New
        SqlParameter("@vpdisponible", SqlDbType.Bit)
        Dim prm_foto As SqlParameter = New SqlParameter("@vpimagen",
        SqlDbType.VarChar, 255)
        prm_sigla.Value = sigla
        prm_nombre.Value = nombre
        prm_descripcion.Value = descripcion
```

```
            prm_tamcabeza.Value = tamcabeza
            prm_categoria.Value = categoria
            prm_disponible.Value = disponible
            prm_foto.Value = foto
            With comando.Parameters
                .Add(prm_sigla)
                .Add(prm_nombre)
                .Add(prm_descripcion)
                .Add(prm_tamcabeza)
                .Add(prm_categoria)
                .Add(prm_disponible)
                .Add(prm_foto)
            End With
            Try
                conn.Open()
                Dim insertados = comando.ExecuteNonQuery
                conn.Close()
                Return insertados
            Catch ex As Exception
                Return 0
            End Try
    End Function
    Public Function TraerListadoVariedadProducto() As DataSet
            Dim Conn As SqlConnection = New
            SqlConnection(ConfigurationSettings.AppSettings("Conexion"))
            Dim comando As SqlCommand = New
            SqlCommand("Sp_TraerListadoVariedadProducto", Conn)
            comando.CommandType = CommandType.StoredProcedure
            Dim da As SqlDataAdapter = New
            SqlDataAdapter("Sp_TraerListadoVariedadProducto", Conn)
            da.SelectCommand.CommandType =
            CommandType.StoredProcedure
            Dim ds As DataSet = New DataSet
            da.Fill(ds, "ListadoVariedadProducto")
```

```
        Return ds
    End Function
    Public Function Traercategoria() As SqlDataReader
        Dim Conn As SqlConnection = New
        SqlConnection(ConfigurationSettings.AppSettings("Conexion"))
        Dim comando As SqlCommand = New
        SqlCommand("Sp_TraerCategoriaProducto", Conn)
        comando.CommandType = CommandType.StoredProcedure
        Conn.Open()
        Dim CategoriaProducto As SqlDataReader =
        comando.ExecuteReader(CommandBehavior.CloseConnection)
        Return CategoriaProducto
    End Function
    Public Function TraerVariedadProductoDetalle(ByVal codigo As Integer) As
    VariedadProductoDetalle
        Dim conn As SqlConnection = New
        SqlConnection(ConfigurationSettings.AppSettings("conexion"))
        Dim comando As SqlCommand = New
        SqlCommand("Sp_TraerVariedadProductoDetalle", conn)
        comando.CommandType = CommandType.StoredProcedure
        Dim prm_codigo As SqlParameter = New SqlParameter("@vpcodigo",
        SqlDbType.Int)
        Dim prm_sigla As SqlParameter = New SqlParameter("@vpsigla",
        SqlDbType.VarChar, 10)
        Dim prm_nombre As SqlParameter = New
        SqlParameter("@vpnombre", SqlDbType.VarChar, 50)
        Dim prm_descripcion As SqlParameter = New
        SqlParameter("@vpdescripcion", SqlDbType.VarChar, 255)
        Dim prm_tamcabeza As SqlParameter = New
        SqlParameter("@vptamcabeza", SqlDbType.SmallInt)
        Dim prm_categoria As SqlParameter = New
        SqlParameter("@cpcodigo", SqlDbType.Int)
        Dim prm_disponible As SqlParameter = New
        SqlParameter("@vpdisponible", SqlDbType.Bit)
```

```
Dim prm_foto As SqlParameter = New SqlParameter("@vpimagen",
SqlDbType.VarChar, 255)
prm_codigo.Value = codigo
prm_sigla.Direction = ParameterDirection.Output
prm_nombre.Direction = ParameterDirection.Output
prm_descripcion.Direction = ParameterDirection.Output
prm_tamcabeza.Direction = ParameterDirection.Output
prm_categoria.Direction = ParameterDirection.Output
prm_disponible.Direction = ParameterDirection.Output
prm_foto.Direction = ParameterDirection.Output
With comando.Parameters
        .Add(prm_codigo)
        .Add(prm_sigla)
        .Add(prm_nombre)
        .Add(prm_descripcion)
        .Add(prm_tamcabeza)
        .Add(prm_categoria)
        .Add(prm_disponible)
        .Add(prm_foto)
End With
Dim VariedadProductoDetalle As VariedadProductoDetalle = New
VariedadProductoDetalle
Try
        conn.Open()
        comando.ExecuteNonQuery()
        conn.Close()
        With VariedadProductoDetalle
                .vpsigla = prm_sigla.Value
                .vpnombre = prm_nombre.Value
                .vpdescripcion = prm_descripcion.Value
                .vptamcabeza = prm_tamcabeza.Value
                .cpcodigo = prm_categoria.Value
                .vpdisponible = prm_disponible.Value
                .vpimagen = prm_foto.Value
```

```
            End With
        Catch ex As Exception
            With VariedadProductoDetalle
                .vpsigla = ""
                .vpnombre = ""
                .vpdescripcion = ""
                .vptamcabeza = 0
                .cpcodigo = 0
                .vpdisponible = False
                .vpimagen = ""
            End With
        End Try
        Return VariedadProductoDetalle
    End Function
    Public Function ModificarVariedadProducto(ByVal codigo As Integer,
    ByVal sigla As String, ByVal nombre As String, ByVal descripcion As
    String, ByVal tamcabeza As Integer, ByVal categoria As Integer, ByVal
    disponible As Boolean, ByVal foto As String) As Integer
        Dim conn As SqlConnection = New
        SqlConnection(ConfigurationSettings.AppSettings("conexion"))
        Dim comando As SqlCommand = New
        SqlCommand("Sp_ModificarVariedadProducto", conn)
        comando.CommandType = CommandType.StoredProcedure
        Dim prm_codigo As SqlParameter = New SqlParameter("@vpcodigo",
        SqlDbType.Int)
        Dim prm_sigla As SqlParameter = New SqlParameter("@vpsigla",
        SqlDbType.VarChar, 10)
        Dim prm_nombre As SqlParameter = New
        SqlParameter("@vpnombre", SqlDbType.VarChar, 50)
        Dim prm_descripcion As SqlParameter = New
        SqlParameter("@vpdescripcion", SqlDbType.VarChar, 255)
        Dim prm_tamcabeza As SqlParameter = New
        SqlParameter("@vptamcabeza", SqlDbType.SmallInt)
```

```
        Dim prm_categoria As SqlParameter = New
        SqlParameter("@cpcodigo", SqlDbType.Int)
        Dim prm_disponible As SqlParameter = New
        SqlParameter("@vpdisponible", SqlDbType.Bit)
        Dim prm_foto As SqlParameter = New SqlParameter("@vpimagen",
        SqlDbType.VarChar, 255)
        prm_codigo.Value = código
        prm_sigla.Value = sigla
        prm_nombre.Value = nombre
        prm_descripcion.Value = descripcion
        prm_tamcabeza.Value = tamcabeza
        prm_categoria.Value = categoria
        prm_disponible.Value = disponible
        prm_foto.Value = foto
        With comando.Parameters
            .Add(prm_codigo)
            .Add(prm_sigla)
            .Add(prm_nombre)
            .Add(prm_descripcion)
            .Add(prm_tamcabeza)
            .Add(prm_categoria)
            .Add(prm_disponible)
            .Add(prm_foto)
        End With
        Try
            conn.Open()
            Dim modificados = comando.ExecuteNonQuery
            conn.Close()
            Return modificados
        Catch ex As Exception
            Return 0
        End Try
    End Function
```

```
    Public Function EliminarVariedadProducto(ByVal codigo As Integer) As
    Integer
        Dim conn As SqlConnection = New
        SqlConnection(ConfigurationSettings.AppSettings("conexion"))
        Dim comando As SqlCommand = New
        SqlCommand("Sp_EliminarVariedadProducto", conn)
        comando.CommandType = CommandType.StoredProcedure
        Dim prm_codigo As SqlParameter = New SqlParameter("@vpcodigo",
        SqlDbType.Int)
        prm_codigo.Value = código
        comando.Parameters.Add(prm_codigo)
        Try
            conn.Open()
            Dim eliminados = comando.ExecuteNonQuery
            conn.Close()
            Return eliminados
        Catch ex As Exception
            Return 0
        End Try
    End Function
End Class
```

Mensaje.

```
Public Class MensajeDB
    Public asunto As String
    Public mensaje As String
    Public numero As Integer
    Public Function InsertarMensaje(ByVal cliente As String, ByVal asunto As
    String, ByVal mensaje As String, ByVal para As Boolean) As Integer
        Dim Conn As SqlConnection = New
        SqlConnection(ConfigurationSettings.AppSettings("Conexion"))
        Dim comando As SqlCommand = New
        SqlCommand("Sp_InsertarMensaje", Conn)
        comando.CommandType = CommandType.StoredProcedure
```

```
        Dim prm_cliente As SqlParameter = New SqlParameter("@clcodigo",
        SqlDbType.Int)
        Dim prm_asunto As SqlParameter = New
        SqlParameter("@measunto", SqlDbType.VarChar, 255)
        Dim prm_mensaje As SqlParameter = New
        SqlParameter("@memensaje", SqlDbType.VarChar, 5000)
        Dim prm_para As SqlParameter = New SqlParameter("@mepara",
        SqlDbType.Bit)
        prm_cliente.Value = cliente
        prm_asunto.Value = asunto
        prm_mensaje.Value = mensaje
        prm_para.Value = para
        With comando.Parameters
            .Add(prm_cliente)
            .Add(prm_asunto)
            .Add(prm_mensaje)
            .Add(prm_para)
        End With
        Try
            Conn.Open()
            Dim insertados = comando.ExecuteNonQuery
            Conn.Close()
            Return insertados
        Catch ex As Exception
            Return 0
        End Try
    End Function
    Public Function InsertarMensajeCliente(ByVal agente As String, ByVal
    cliente As String, ByVal asunto As String, ByVal mensaje As String, ByVal
    para As Boolean) As Integer
        Dim Conn As SqlConnection = New
        SqlConnection(ConfigurationSettings.AppSettings("Conexion"))
        Dim comando As SqlCommand = New
        SqlCommand("Sp_InsertarMensajeCliente", Conn)
```

```
            comando.CommandType = CommandType.StoredProcedure
            Dim prm_agente As SqlParameter = New SqlParameter("@uscedula",
            SqlDbType.VarChar, 10)
            Dim prm_cliente As SqlParameter = New SqlParameter("@clcodigo",
            SqlDbType.Int)
            Dim prm_asunto As SqlParameter = New
            SqlParameter("@measunto", SqlDbType.VarChar, 255)
            Dim prm_mensaje As SqlParameter = New
            SqlParameter("@memensaje", SqlDbType.VarChar, 5000)
            Dim prm_para As SqlParameter = New SqlParameter("@mepara",
            SqlDbType.Bit)
            prm_agente.Value = agente
            prm_cliente.Value = cliente
            prm_asunto.Value = asunto
            prm_mensaje.Value = mensaje
            prm_para.Value = para
            With comando.Parameters
                .Add(prm_agente)
                .Add(prm_cliente)
                .Add(prm_asunto)
                .Add(prm_mensaje)
                .Add(prm_para)
            End With
            Try
                Conn.Open()
                Dim insertados = comando.ExecuteNonQuery
                Conn.Close()
                Return insertados
            Catch ex As Exception
                Return 0
            End Try
        End Function
        Public Function TraerMensajeCliente(ByVal codigo As Integer) As DataSet
```

```
        Dim Conn As SqlConnection = New
        SqlConnection(ConfigurationSettings.AppSettings("Conexion"))
        Dim da As SqlDataAdapter = New
        SqlDataAdapter("Sp_TraerMensajesCliente", Conn)
        Dim prm_codigo As SqlParameter = New SqlParameter("@clcodigo",
        SqlDbType.Int)
        prm_codigo.Value = codigo
        da.SelectCommand.CommandType =
        CommandType.StoredProcedure
        da.SelectCommand.Parameters.Add(prm_codigo)
        Dim ds As DataSet = New DataSet
        da.Fill(ds, "MensajesCliente")
        Return ds
    End Function
    Public Function TraerMensajeAgente(ByVal codigo As String) As DataSet
        Dim Conn As SqlConnection = New
        SqlConnection(ConfigurationSettings.AppSettings("Conexion"))
        Dim da As SqlDataAdapter = New
        SqlDataAdapter("Sp_TraerMensajesAgente", Conn)
        Dim prm_codigo As SqlParameter = New SqlParameter("@uscedula",
        SqlDbType.VarChar, 10)
        prm_codigo.Value = codigo
        da.SelectCommand.CommandType =
        CommandType.StoredProcedure
        da.SelectCommand.Parameters.Add(prm_codigo)
        Dim ds As DataSet = New DataSet
        da.Fill(ds, "MensajesAgente")
        Return ds
    End Function
    Public Function LeerMensaje(ByVal codigo As Integer, ByVal para As
    Boolean) As MensajeDetalle
        Dim Conn As SqlConnection = New
        SqlConnection(ConfigurationSettings.AppSettings("Conexion"))
```

```
Dim comando As SqlCommand = New
SqlCommand("Sp_LeerMensaje", Conn)
comando.CommandType = CommandType.StoredProcedure
Dim prm_para As SqlParameter = New SqlParameter("@mepara",
SqlDbType.Bit)
Dim prm_codigo As SqlParameter = New
SqlParameter("@mecodigo", SqlDbType.Int)
Dim prm_asunto As SqlParameter = New
SqlParameter("@measunto", SqlDbType.VarChar, 255)
Dim prm_mensaje As SqlParameter = New
SqlParameter("@memensaje", SqlDbType.VarChar, 5000)
prm_para.Value = para
prm_codigo.Value = código
prm_asunto.Direction = ParameterDirection.Output
prm_mensaje.Direction = ParameterDirection.Output
With comando.Parameters
        .Add(prm_para)
        .Add(prm_codigo)
        .Add(prm_asunto)
        .Add(prm_mensaje)
End With
Dim MensajeDetalle As MensajeDetalle = New MensajeDetalle
Try
        Conn.Open()
        comando.ExecuteNonQuery()
        Conn.Close()
        With MensajeDetalle
                .asunto = prm_asunto.Value
                .mensaje = prm_mensaje.Value
        End With
Catch ex As Exception
        With MensajeDetalle
                .asunto = ""
                .mensaje = ""
```

```
                End With
            End Try
            Return MensajeDetalle
        End Function
        Public Function MarcarMensajeLeido(ByVal codigo As Integer) As Integer
            Dim Conn As SqlConnection = New
            SqlConnection(ConfigurationSettings.AppSettings("Conexion"))
            Dim comando As SqlCommand = New
            SqlCommand("Sp_MarcarMensajeLeido", Conn)
            comando.CommandType = CommandType.StoredProcedure
            Dim prm_codigo As SqlParameter = New
            SqlParameter("@mecodigo", SqlDbType.Int)
            prm_codigo.Value = código
            comando.Parameters.Add(prm_codigo)
            Try
                Conn.Open()
                Dim modificados = comando.ExecuteNonQuery
                Conn.Close()
                Return modificados
            Catch ex As Exception
                Return 0
            End Try
        End Function
        Public Function ContarMensajesCliente(ByVal codigo As Integer) As
        ContarMensaje
            Dim Conn As SqlConnection = New
            SqlConnection(ConfigurationSettings.AppSettings("Conexion"))
            Dim comando As SqlCommand = New
            SqlCommand("Sp_ContarMensajesCliente", Conn)
            comando.CommandType = CommandType.StoredProcedure
            Dim prm_codigo As SqlParameter = New SqlParameter("@clcodigo",
            SqlDbType.Int)
            Dim prm_numero As SqlParameter = New
            SqlParameter("@mensajes", SqlDbType.Int)
```

```
        prm_codigo.Value = codigo
        prm_numero.Direction = ParameterDirection.Output
        With comando.Parameters
            .Add(prm_codigo)
            .Add(prm_numero)
        End With
        Dim NumeroMensaje As ContarMensaje = New ContarMensaje
        Try
            Conn.Open()
            comando.ExecuteNonQuery()
            Conn.Close()
            With NumeroMensaje
                .numero = prm_numero.Value
            End With
        Catch ex As Exception
            With NumeroMensaje
                .numero = 0
            End With
        End Try
        Return NumeroMensaje
    End Function
    Public Function ContarMensajesAgente(ByVal cedula As Integer) As
    ContarMensaje
        Dim Conn As SqlConnection = New
        SqlConnection(ConfigurationSettings.AppSettings("Conexion"))
        Dim comando As SqlCommand = New
        SqlCommand("Sp_ContarMensajesAgente", Conn)
        comando.CommandType = CommandType.StoredProcedure
        Dim prm_cedula As SqlParameter = New SqlParameter("@uscedula",
        SqlDbType.VarChar, 10)
        Dim prm_numero As SqlParameter = New
        SqlParameter("@mensajes", SqlDbType.Int)
        prm_cedula.Value = cedula
        prm_numero.Direction = ParameterDirection.Output
```

```
        With comando.Parameters
            .Add(prm_cedula)
            .Add(prm_numero)
        End With
        Dim NumeroMensaje As ContarMensaje = New ContarMensaje
        Try
            Conn.Open()
            comando.ExecuteNonQuery()
            Conn.Close()
            With NumeroMensaje
                .numero = prm_numero.Value
            End With
        Catch ex As Exception
            With NumeroMensaje
                .numero = 0
            End With
        End Try
        Return NumeroMensaje
    End Function
    Public Function TraerClientesAgente(ByVal cedula As String) As
    SqlDataReader
        Dim conn As SqlConnection = New
        SqlConnection(ConfigurationSettings.AppSettings("conexion"))
        Dim comando As SqlCommand = New
        SqlCommand("Sp_TraerClientesAgente", conn)
        comando.CommandType = CommandType.StoredProcedure
        Dim prm_cedula As SqlParameter = New SqlParameter("@uscedula",
        SqlDbType.VarChar, 10)
        prm_cedula.Value = cedula
        comando.Parameters.Add(prm_cedula)
        conn.Open()
        Dim clientes As SqlDataReader =
        comando.ExecuteReader(CommandBehavior.CloseConnection)
        Return clientes
```

```
        End Function
End Class
```

4.4. Solución en programa de Visual Studio .Net.

La solución en programa, se la encontrará en disco compacto adjunto a la presente tesis.

Capítulo 5

Implementación y Pruebas.

En este capítulo, se ilustra el procedimiento para poner en marcha el servidor en donde se ejecutará la aplicación E-Commerce. Se explica paso a paso la configuración del servidor Web, Base de Datos, instalación de la aplicación y su puesta en funcionamiento.

5.1. Configuración del Servidor.

5.1.1. Internet Information Server (IIS).

El Internet Information Server (IIS) es el servidor de aplcaciones web que permite publicar en Internet aplicaciones ASP.NET. Para instalar Internet Information Server (IIS) se debe realizar lo siguiente:

- Abrir la Ficha Agregar o Quitar Programas que se encuentra en el Panel de Control.

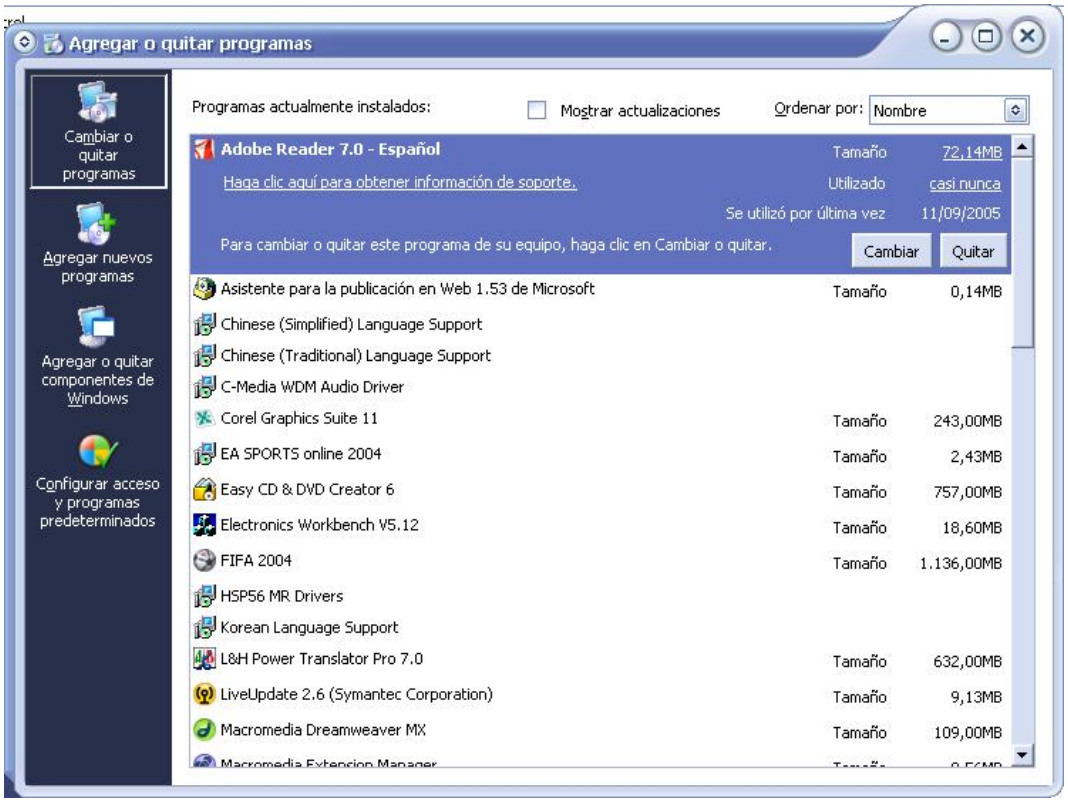


Figura 5.1. Agregar o Quitar Programas

- Haga clic en la opción Agregar o quitar componentes de Windows, aparecerá la siguiente pantalla:

Figura 5.2. Asistente para Componentes de Windows.

- Marque la casilla de verificación: Servicios de Internet Information Server (IIS) y haga un clic en Detalles, aparecerá la siguiente pantalla:

Figura 5.3. Servicios de Internet Information Server (IIS).

- Asegúrese de activar las casillas de verificación: Extensiones de Servidor de FrontPage 2000, Servicio World Wide Web Complemento de Servicios de Internet Information Server, a continuación debe hacer clic en Aceptar, aparecerá la siguiente pantalla:

5.4. Asistente para componentes de Windows (Componentes de Windows).

- Haga clic en Siguiente, aparecerá la pantalla que se muestra a continuación:

Figura 5.5. Asistente para componentes de Windows (Configurando Componentes).

- Espere mientras el asistente realiza las acciones necesarias, una vez que el proceso concluya, aparecerá la siguiente pantalla:

Figura 5.6. Asistente para componentes de Windows (Finalización del Asistente para Componentes de Windows).

- Haga clic en Finalizar para concluir la configuración de Internet Information Server (IIS).

5.1.2. Instalación de SQL Server 2000.

Para el desarrollo de la aplicación Web E-Commerce se ha utilizado el Gestor de Bases de Datos SQL Server 2000, a continuación se detalla los pasos que se deben seguir para la instalación de dicho gestor.

- Inserte el CD con el Gestor de Bases de Datos SQL Server 2000 en la unidad de CD-Rom, con lo que aparecerá la siguiente pantalla:

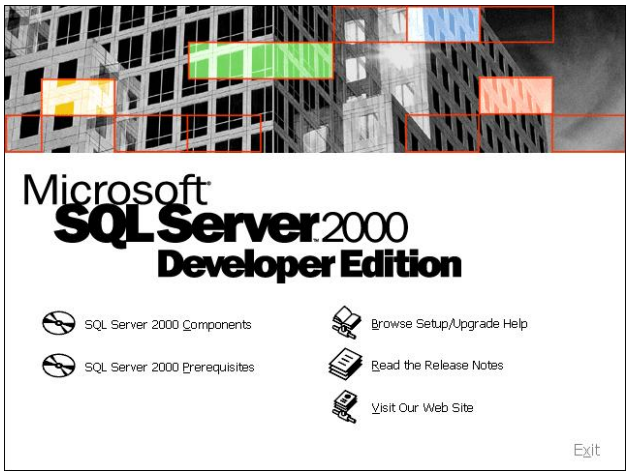


Figura 5.7. Instalación de SQL Server 2000 (Opciones de Instalación).

- Haga clic en la opción SQL Server 2000 Components (Componentes de SQL Server 2000), aparecerá la siguiente pantalla:

Figura 5.7. Instalación de SQL Server 2000 (Instalar Componentes).

- Haga clic en la opción Intall Database Server (Instalar Servidor de Bases de Datos), aparecerá la siguiente pantalla:

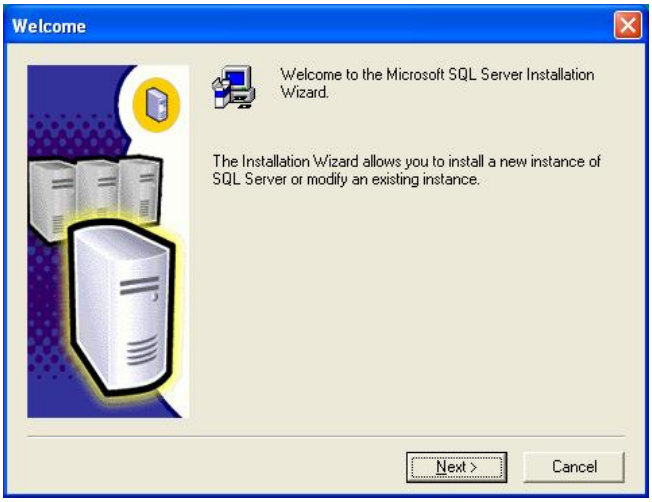


Figura 5.8. Instalación de SQL Server 2000 (Bienvenido al Asistente de Instalación).

- Haga clic en el botón Next (Siguiente), aparecerá la siguiente pantalla:

Figura 5.9. Instalación de SQL Server 2000 (Nombre del Computador).

- Elija el tipo de computador en el que se desea instalar el Gestor de Bases de Datos SQL Server 2000, para este caso elija la opción Local Computer (Computador Local), luego haga clic en el botón Next (Siguiente), aparecerá la siguiente pantalla:

Figura 5.10. Instalación de SQL Server 2000 (Selección de la Instalación).

- Seleccione el tipo de Instalación que desea realizar, para el caso de Ecommerce elija Create a new instante of SQL Server, or install Client Tools (Crear una nueva instancia de SQL Server o instalar Herramientas de Cliente), luego haga clic en el botón Next (Siguiente), aparecerá la siguiente pantalla:

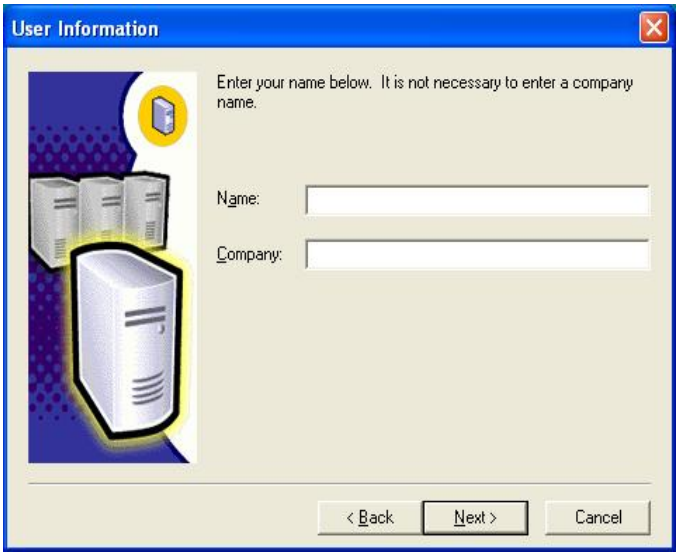


Figura 5.11. Instalación de SQL Server 2000 (Información de Usuario).

- Ingrese la información del nombre del propietario de la licencia y la compañía a la cual pertenece, luego haga clic en el botón Next (Siguiente), aparecerá la pantalla que se muestra a continuación:

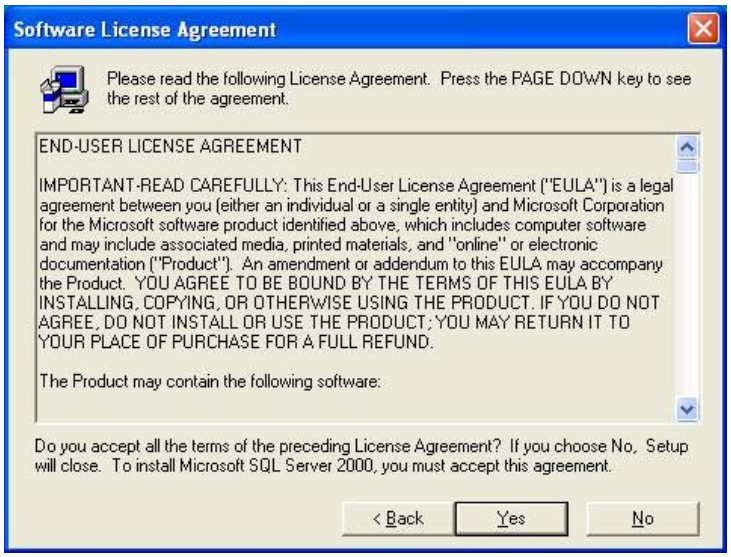


Figura 5.12. Instalación de SQL Server 2000 (Aceptación de la Licencia del Software).

- Haga clic en el botón Yes (Si), para aceptar los términos de la licencia del Software, aparecerá la siguiente pantalla:

Figura 5.13. Instalación de SQL Server 2000 (Definición del tipo de Instalación).

- Seleccione el tipo de Instalación que desea realizar, para el caso de Ecommerce elija la opción Server and Client Tools (Herramientas de Cliente y Servidor), al escoger esta opción está instalando tanto la Base de Datos como los clientes para utilizar la Base de Datos, luego haga clic en el botón Next (Siguiente), aparecerá la pantalla que se muestra a continuación:

Figura 5.14. Instalación de SQL Server 2000 (Nombre de la Instancia de la Base de Datos).

- Ingrese el nombre de la Instancia de la Base de Datos SQL Server, o puede escoger que se coloque el nombre de la instancia de la Base de Datos con el

nombre establecido por defecto, luego haga clic en el botón Next (Siguiente), aparecerá la pantalla que se muestra a continuación:

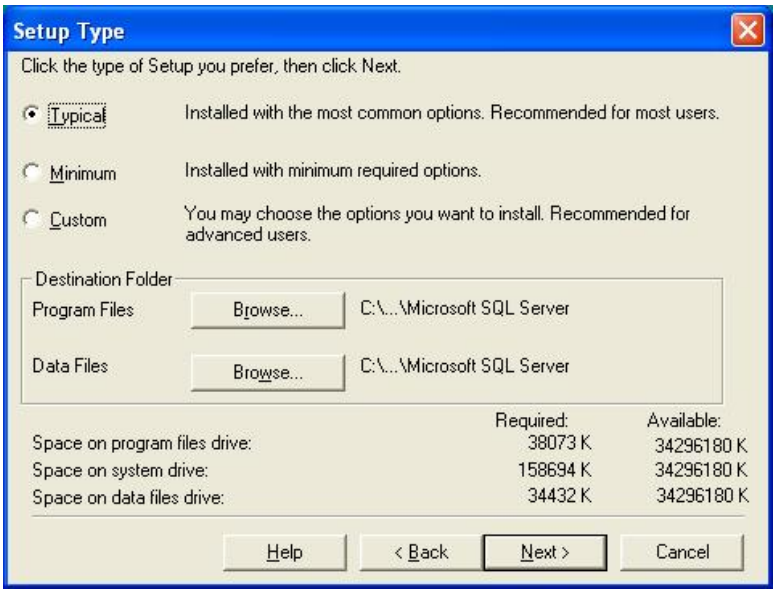


Figura 5.15. Instalación de SQL Server 2000 (Tipo de Instalación).

- Escoja el tipo de instalación de su preferencia, el directorio en el que se almacenaran los archivos y los datos de SQL Server 2000, luego haga clic en el botón Next (Siguiente), aparecerá la pantalla que se muestra a continuación:

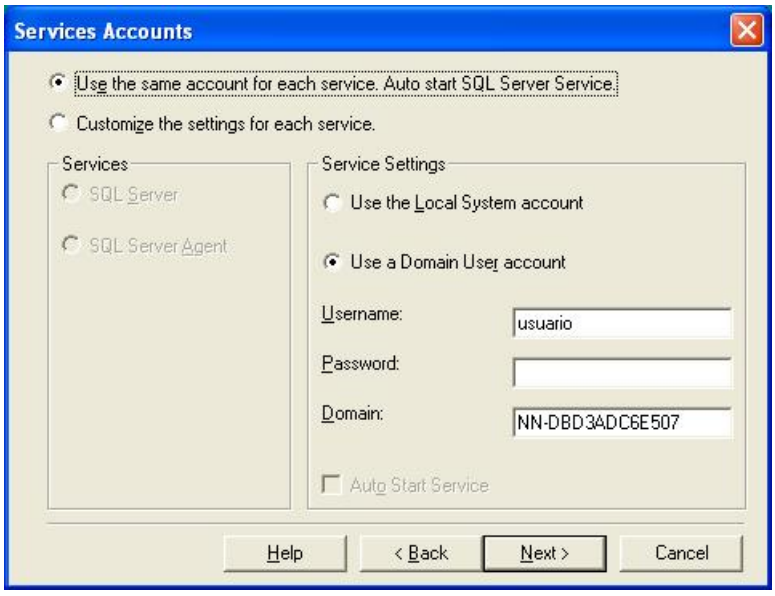


Figura 5.16. Instalación de SQL Server 2000 (Configuración de Servicios).

- Escoja las configuraciones de servicio, y luego haga clic en el botón Next (Siguiente), aparecerá la pantalla que se muestra a continuación:

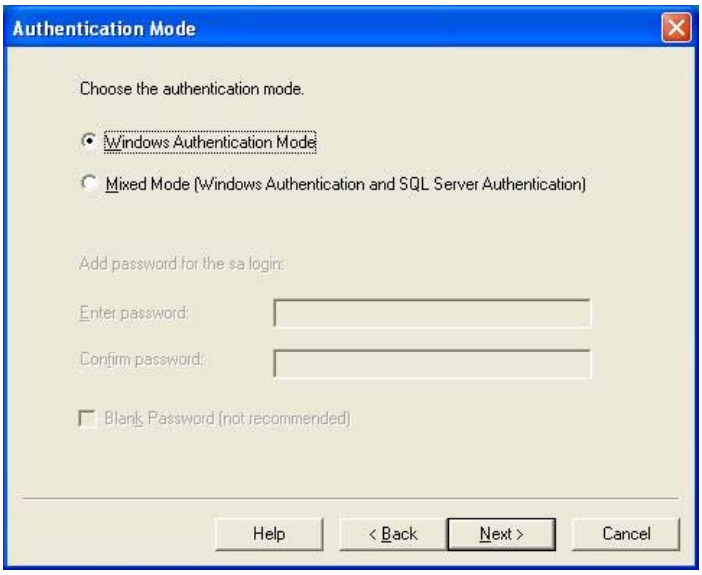


Figura 5.17. Instalación de SQL Server 2000 (Modo de Autenticación).

- Elija el modo de autenticación que se utilizará para conectarse con la Base de Datos, puede escoger la opción marcada por defecto Windows Authentication Mode (Modo de Autenticación de Windows), luego haga clic en el botón Next (Siguiente), aparecerá la pantalla que se muestra a continuación:

Figura 5.18. Instalación de SQL Server 2000 (Iniciar la copia de los Archivos).

- Haga clic en el botón Next (Siguiente), para iniciar el proceso de copia de los archivos necesarios para la instalación de SQL Server 2000, aparecerá la siguiente pantalla:

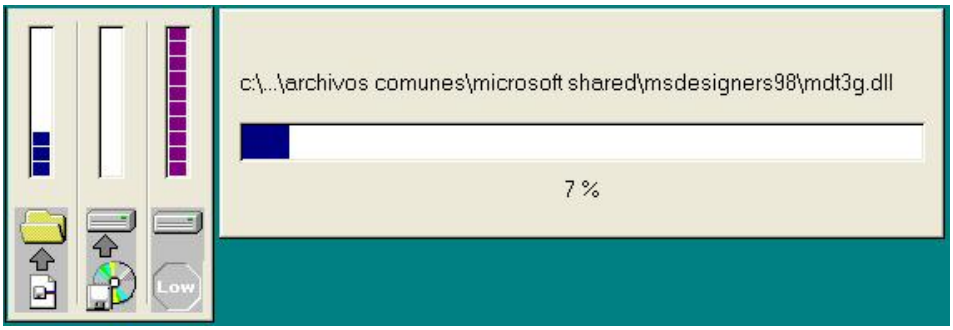


Figura 5.19. Instalación de SQL Server 2000 (Copiando los archivos de Instalación).

- Espere mientras el asistente copia los archivos de la instalación de SQL Server 2000, finalizar aparecerá la siguiente pantalla:

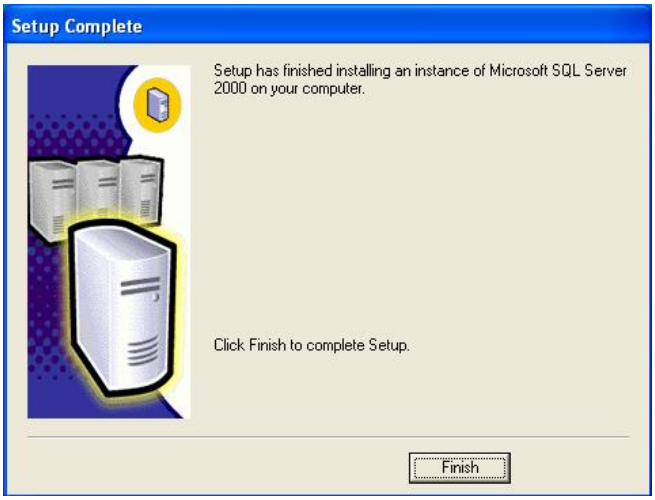


Figura 5.20. Instalación de SQL Server 2000 (Instalación Completa).

- Presione el botón Finísh (Finalizar), para completar la instalación de SQL Server 2000.

5.1.3. Configuración de SQL Server 2000.

Para realizar el enlace entre la aplicación Ecommerce y el Gestor de Bases de Datos SQL Server 2000 realice la siguiente configuración:

- Abra Enterprise Manager de SQL Server 2000.
- Seleccione el servidor local (Windows NT)
- Posteriormente en la opción de seguridad haga un clic con el botón derecho en Inicio de Sesión y seleccione Nuevo Inicio de Sesión.

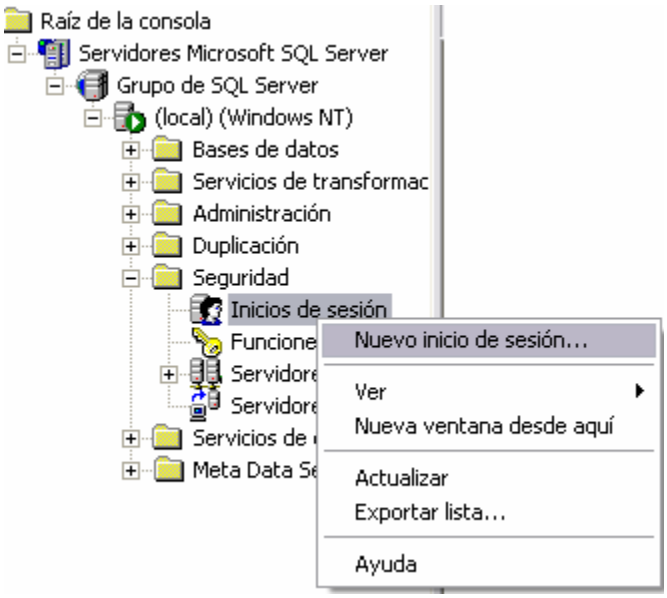


Figura 5.21. Inicios de Sesión (SQL Server Enterprise Manager).

- Se presentará la siguiente pantalla, seleccione la pestaña General, en el campo Name ingrese el nombre del usuario en este caso escriba TrebolUsr.

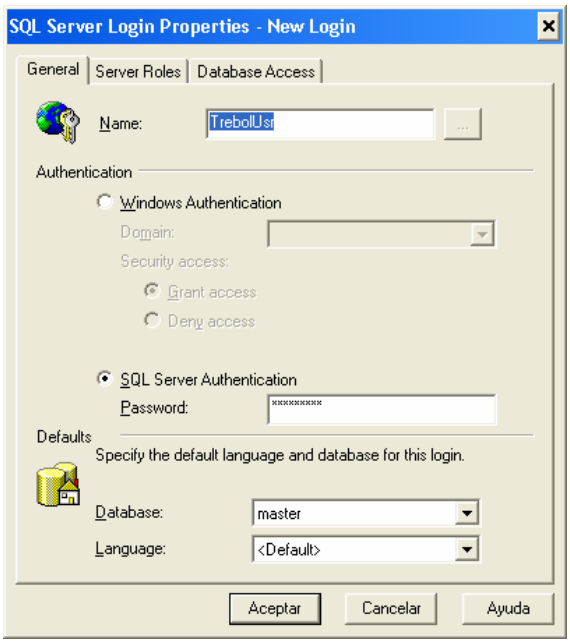


Figura 5.22. Propiedades de Inicio de Sesión (SQL Server Enterprise Manager).

- Marque la opción Autenticación de SQL Server.
- En el campo password ingrese la contraseña para conectarse a la Base de Datos en este caso digite “Ecommerce”.
- Seleccione la pestaña Acceso a base de datos.

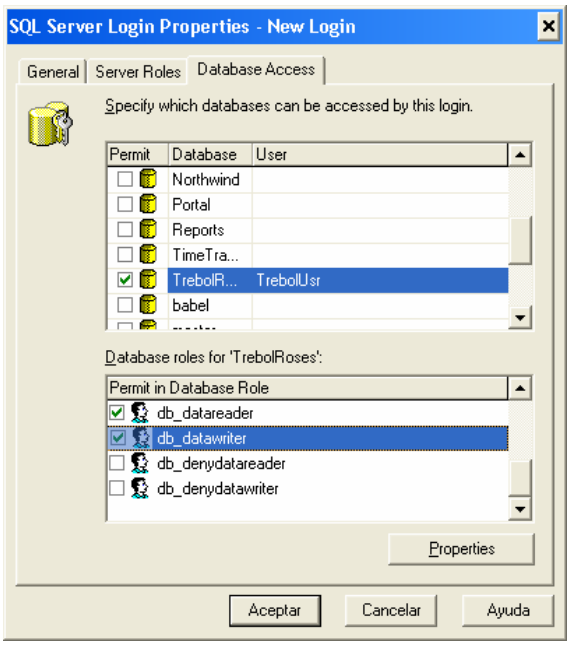


Figura 5.23. Propiedades de Inicio de Sesión con el Usuario TrebolUsr agregado (SQL Server Enterprise Manager).

- Seleccione la Bases de Datos TrebolRoses, y active los siguientes roles de acceso: Public, db_datareader, db_datawriter.

- Haga un clic en botón Aceptar.

5.2. Instalación de la Solución.

E-Commerce viene acompañado de un medio de instalación denominado EcommerceSetup", el mismo que facilita la distribución de la aplicación.

En caso de que necesite instalar la aplicación en un servidor de hosting, tendrá que utilizar un cliente ftp y transferir el contenido de la carpeta "WWW", provista con sus medios de instalación. Para configurar la base de datos, siga las instrucciones de su proveedor de Hosting.

5.2.1. Requisitos del sistema.

Los requerimientos mínimos de instalación para E-Commerce son:

- Windows XP o Windows Server 2003

- 256 MB de Ram
- 10 MB de Espacio en Disco inicial.
- .NET Framework 1.1
- Internet Information Services 5.0 o Superior.
- SQL Server 2000 o MSDE
- Servicio de DNS

5.2.2. Proceso de Instalación.

- Para iniciar el proceso de instalación simplemente necesita hacer doble clic sobre el archivo Setup.exe, contenido dentro de la carpeta "Setup" en los medios de instalación provistos con la aplicación. Inmediatamente aparecerá la siguiente pantalla:

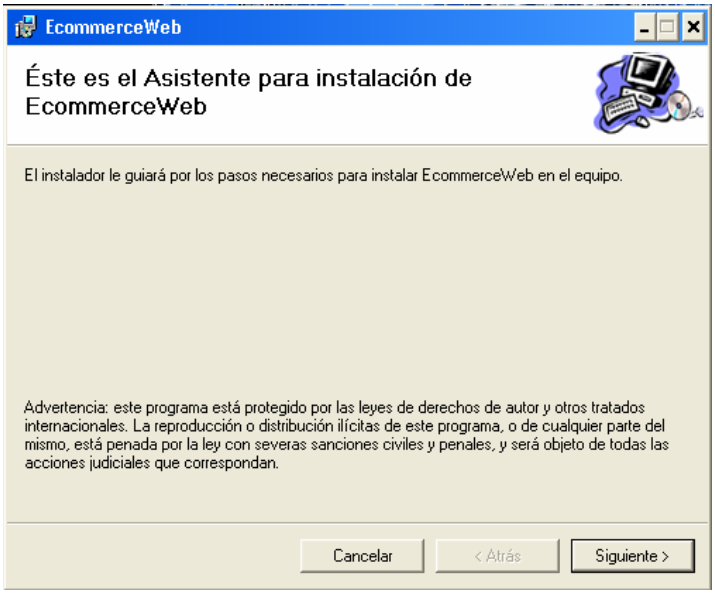


Figura 5.24. Asistente de Instalación de EcommerceWeb.

- Haga clic en el botón "Siguiente" para continuar el proceso.
- En la siguiente pantalla, escoja el directorio virtual en el cual desea copiar los archivos de instalación, así como el puerto que desea asignar para escuchar las solicitudes Web para la aplicación.

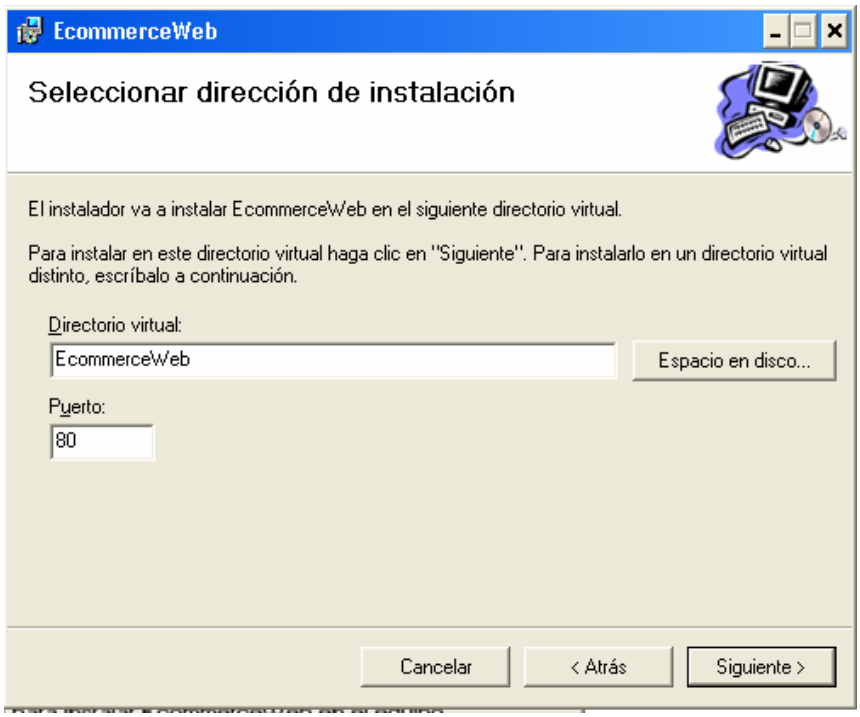


Figura 5.25. Asistente de Instalación de EcommerceWeb (Seleccionar dirección de Instalación).

- Si hace clic en el botón "Espacio en disco", podrá ver la siguiente pantalla, la cual le informa a cerca del espacio que necesitará la instalación.

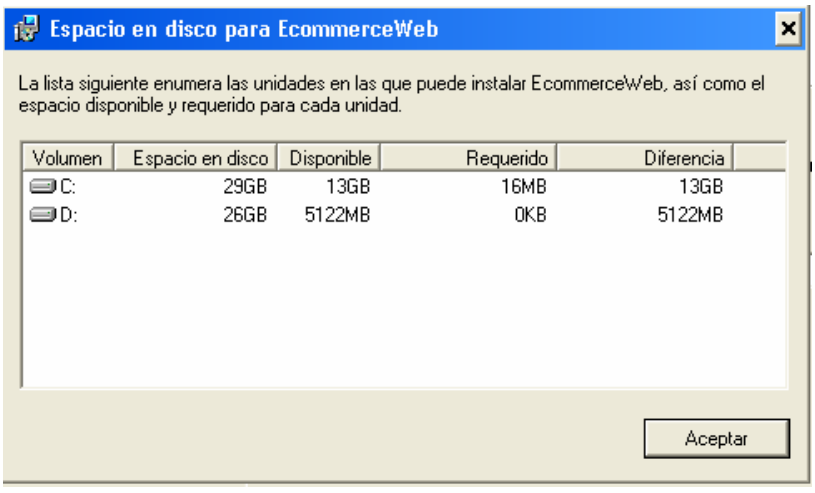


Figura 5.26. Asistente de Instalación de EcommerceWeb (Espacio en disco para Ecommerce Web).

- A continuación confirme su decisión de continuar con la instalación, adicionalmente puede regresar a las pantallas anteriores para corregir la información suministrada o de cancelar definitivamente el proceso.

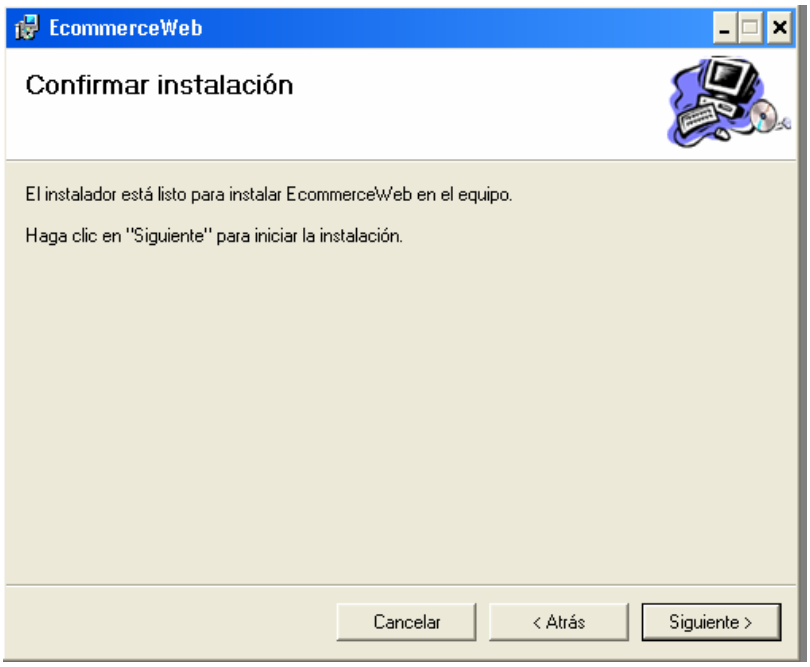


Figura 5.27. Asistente de Instalación de EcommerceWeb (Confirmar Instalación).

- Una vez confirmada la instalación, el asistente iniciará el proceso de copia de los archivos necesarios en su sistema.

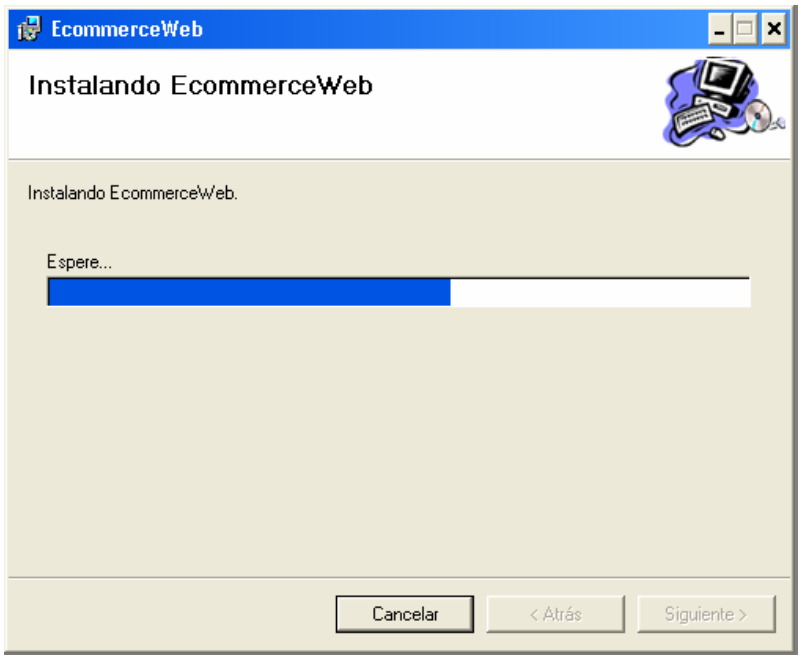


Figura 5.28. Asistente de Instalación de EcommerceWeb (Instalando Ecommerce Web).

- Al finalizar la copia de los archivos, aparecerá la siguiente pantalla:

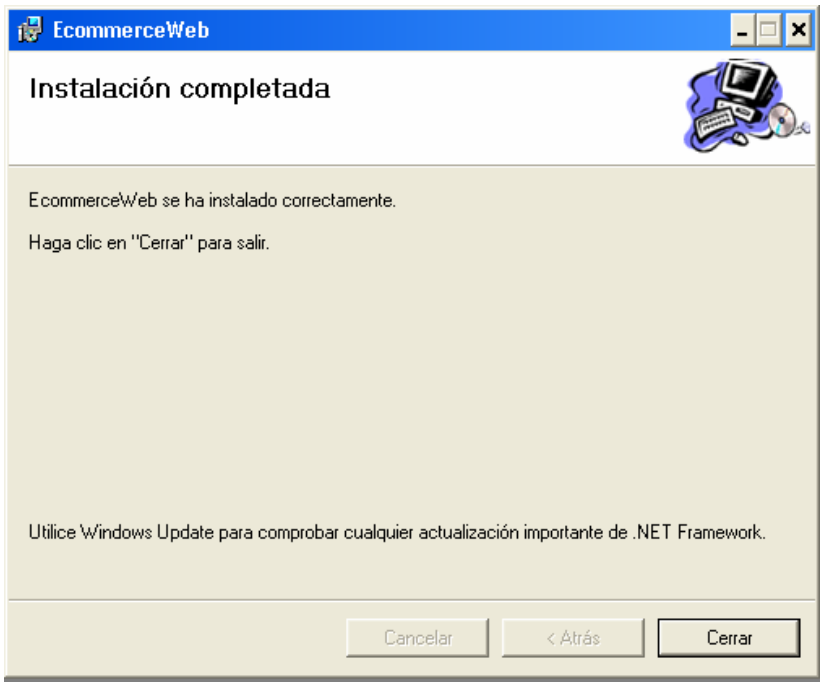


Figura 5.29. Asistente de Instalación de EcommerceWeb (Instalación Completada).

- Haga clic en el botón Cerrar para completar la operación.

5.2.3. Creación de la Base de Datos.

Al finalizar la instalación del sistema, se creará una carpeta denominada "Ecommerce" dentro del menú Inicio/Programas, dicha carpeta apunta hacia el archivo de creación de la base de datos, ubicada en "%ProgramFiles%\Ecommerce\", el archivo se denomina Ecommerce.sql, y se debe abrir en SQL Server Query Analyzer.

Ecommerce.sql es un script SQL, que contiene la definición completa de la base de datos del sistema, una vez abierto el archivo, simplemente presione la tecla F5 para ejecutar el proceso de creación del almacén de datos.

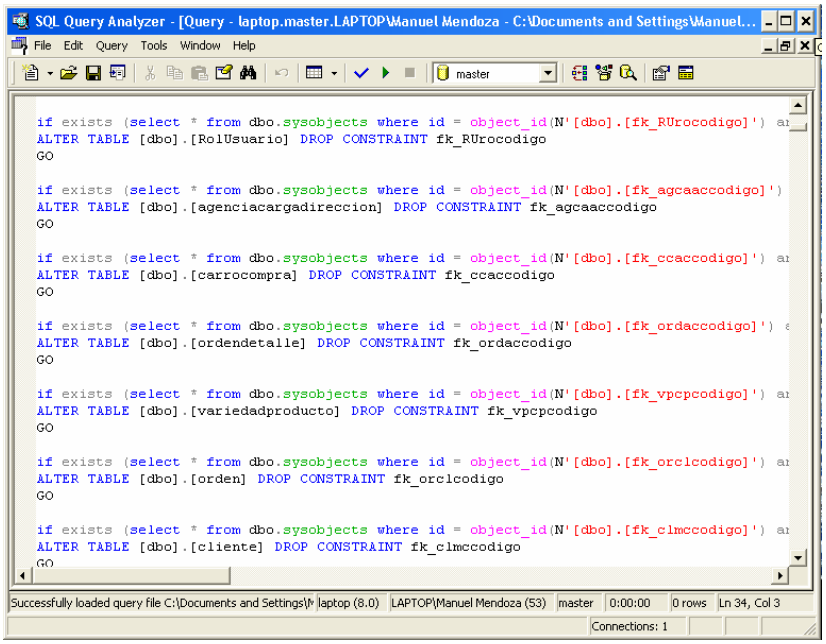


Figura 5.30. SQL Query Analizer.

Una vez creada la base de datos, es necesario asignar los permisos necesarios al usuario ASP.NET, para lo cual debe seguir los pasos descritos en el apartado "Configuración del Servidor".

En este punto, la instalación de la aplicación ha sido completada y está lista para su utilización.

5.2.4. Configuración del Servicio de DNS.

Para el correcto funcionamiento de Ecommerce, es necesario que cuente con un servicio DNS correctamente configurado y funcionando, también necesita crear los siguientes registros en su servidor DNS:

- Registro A, con nombre de host www.
- Registro A, con nombre de host admin.
- Registro A, con nombre de host ws.
- Agregue los registros correspondientes para la zona de búsqueda inversa.

5.2.5. Configuración de los Sitios Web

Una vez terminada la instalación de Ecommerce mediante el asistente y creados los registros en el servidor DNS, se tiene que configurar tres sitios Web en Internet

Information Services, correspondientes a los dos módulos Web que conforma Ecommerce:

5.2.5.1. Módulo para el cliente.

Para configurar este módulo, abra el administrador de Internet Information Services:

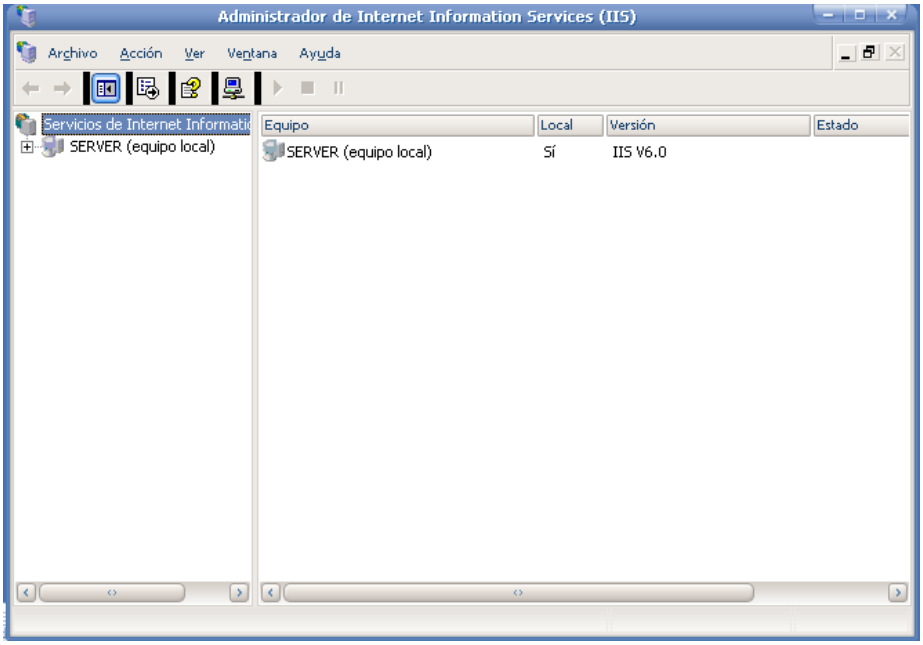


Figura 5.31. Administrador de Internet Information Services (IIS).

Expanda el Servidor en el que instaló la aplicación Ecommerce, y dentro de este último, expanda la carpeta Sitios Web:

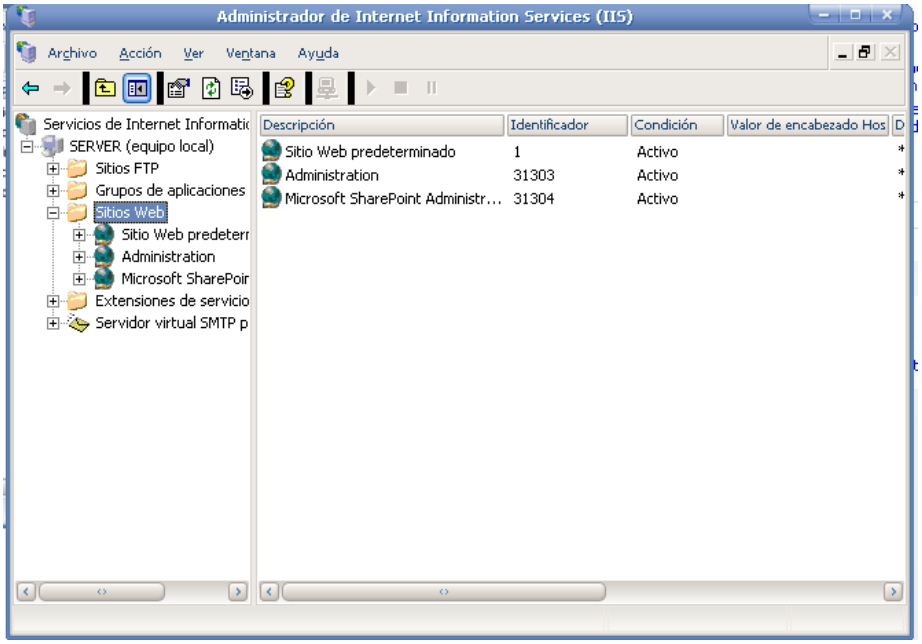


Figura 5.32. Administrador de Internet Information Services (IIS) con la carpetas Sitios Web expandida.

Haga clic derecho sobre la carpeta Sitios Web, en el menú que se despliega seleccione Nuevo y luego Sitio Web:

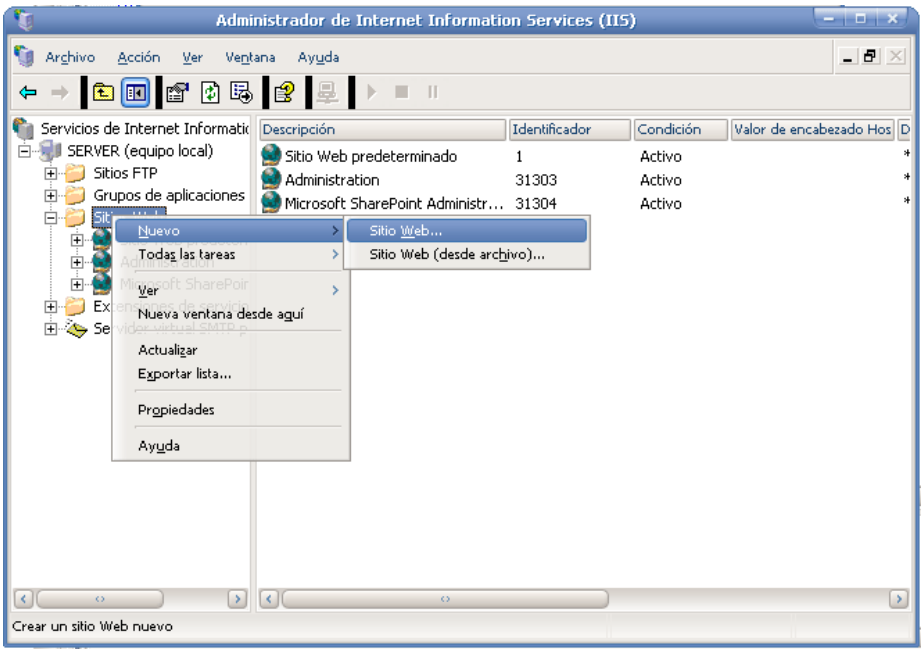


Figura 5.33. Administrador de Internet Information Services (IIS) Creación de un Nuevo Sitio Web.

Aparecerá el asistente para la creación de sitios Web:

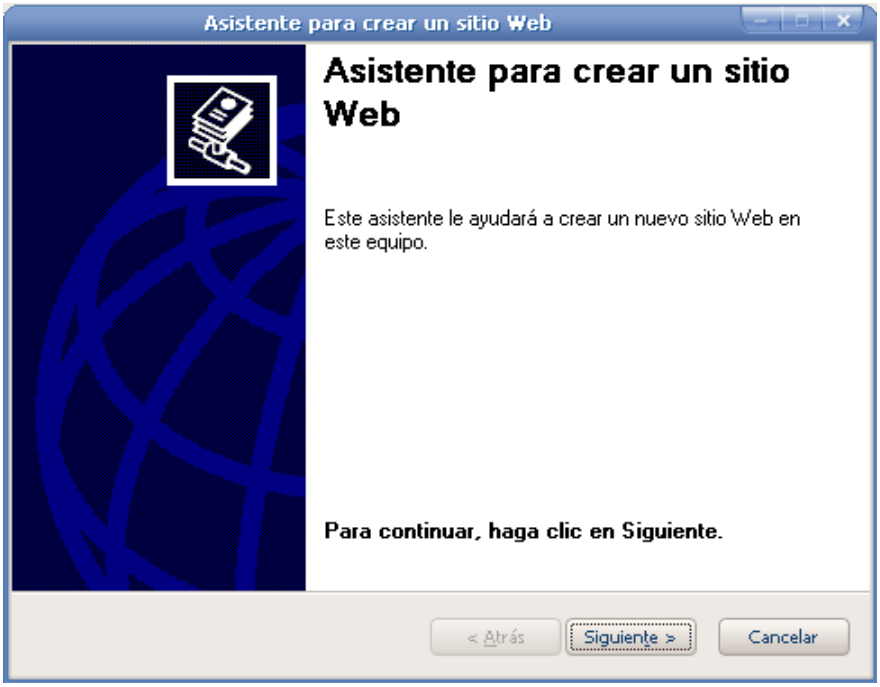


Figura 5.34. Asistente para crear un Sitio Web.

Haga clic en el botón Siguiente, aparecerá la pantalla que se muestra a continuación:

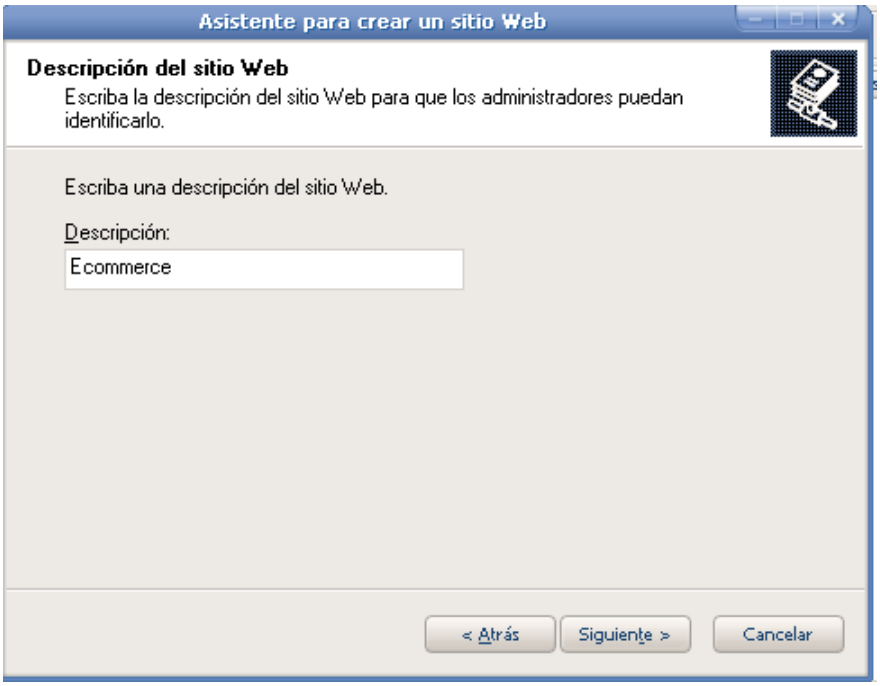


Figura 5.35. Asistente para crear un Sitio Web (Descripción del Sitio Web).

Se le solicitará una descripción para el nuevo sitio Web, escriba Ecommerce y haga clic en Siguiente, aparecerá la pantalla que se muestra a continuación:

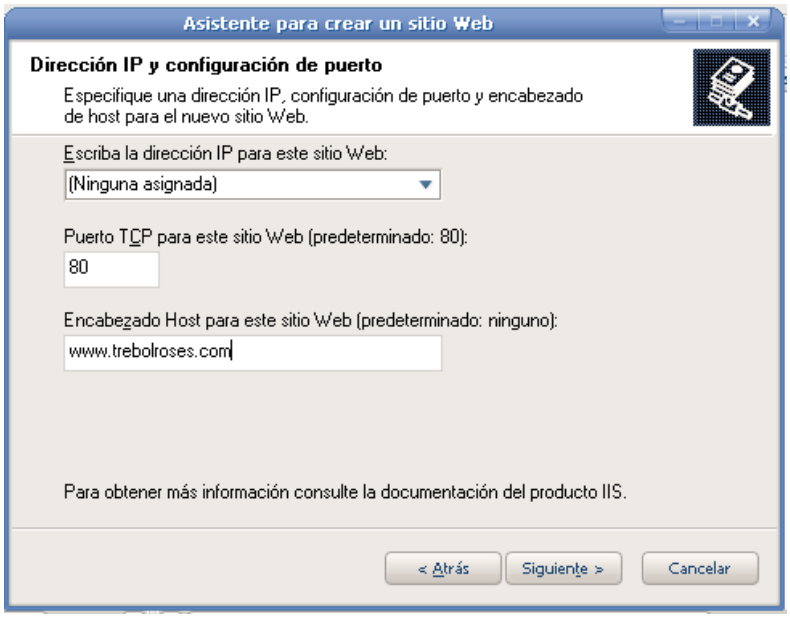


Figura 5.36. Asistente para crear un Sitio Web (Dirección IP y Configuración de Puerto).

En la dirección IP, digite la dirección que desee asignar al sitio Web, o deje el valor por defecto, en Puerto TCP, digite el puerto que desee asignar al sitio Web o deje el valor por defecto, en Encabezado Host, escriba WWW. Seguido del nombre de dominio, tal como se muestra en la figura anterior.

Una vez completada la información, haga clic en Siguiente, aparecerá la pantalla que se muestra a continuación:

Figura 5.37. Asistente para crear un Sitio Web (Directorio Particular del Sitio).

Haga clic en el botón examinar y navegue hasta la carpeta %systemdrive%\Inetpub\wwwroot\Ecommerce o hasta la carpeta en donde instaló Ecommerce.

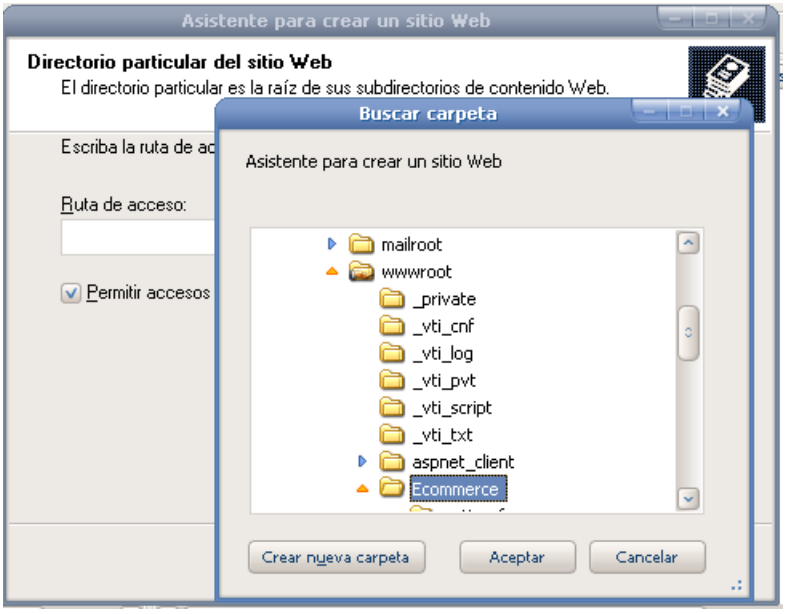


Figura 5.38. Asistente para crear un Sitio Web (Buscar Carpetas).

Haga clic en el botón Aceptar y luego en Siguiente, aparecerá la pantalla que se muestra a continuación:

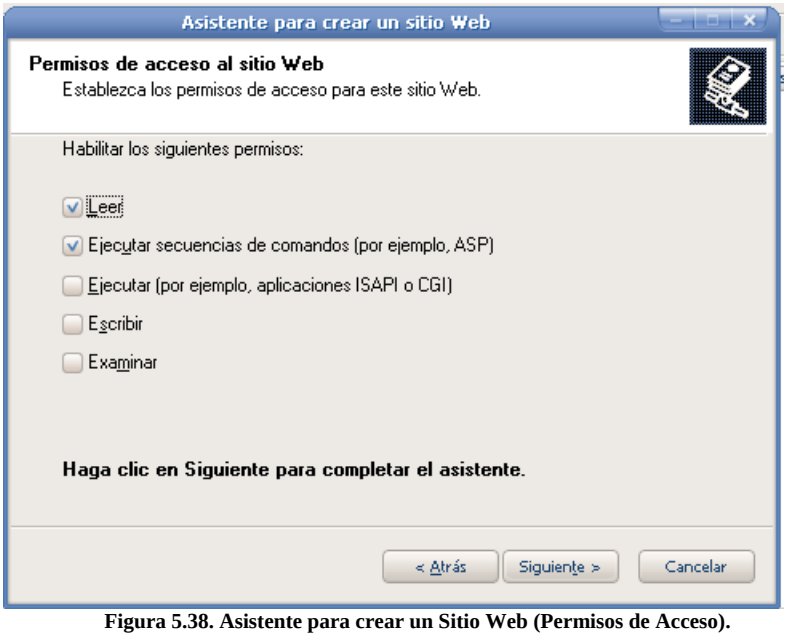


Figura 5.38. Asistente para crear un Sitio Web (Permisos de Acceso).

Deje los valores por defecto y haga clic en el botón Siguiente, aparecerá la pantalla que se muestra a continuación:

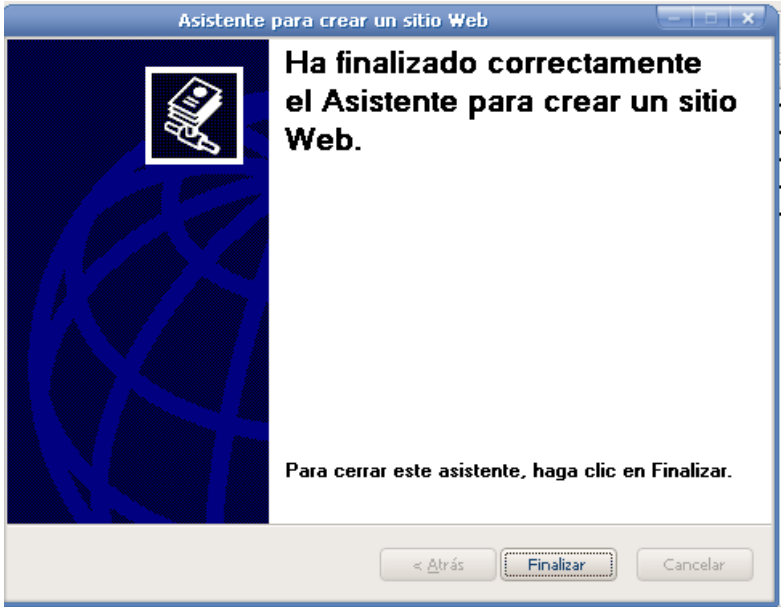


Figura 5.39. Asistente para crear un Sitio Web (Finalización de la Creación del Sitio Web).

Haga clic en Finalizar para concluir la configuración del sitio.

5.2.5.2. Módulo para el administrador.

Para configurar este módulo, abra el administrador de Internet Information Services:

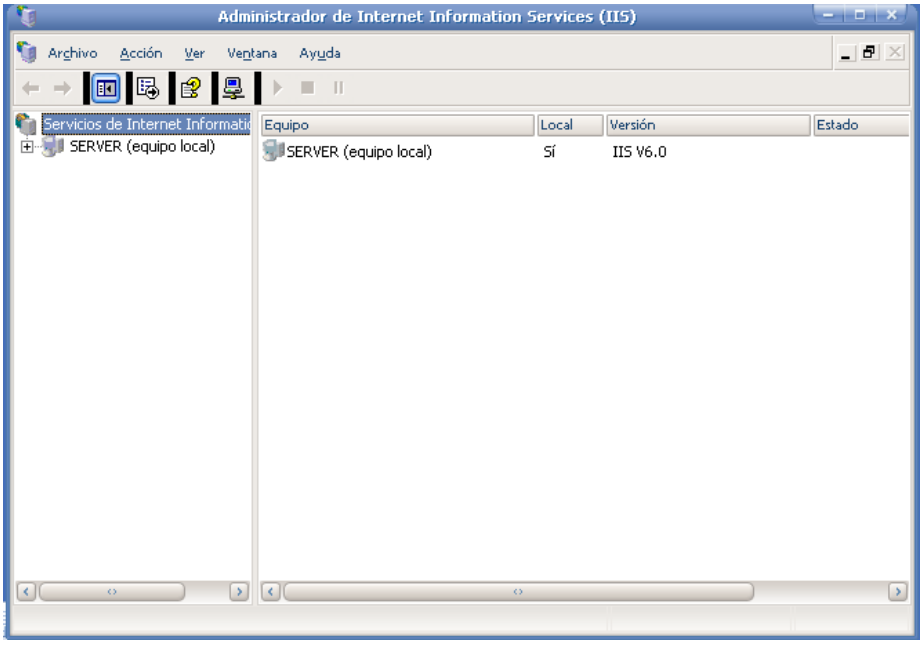


Figura 5.40. Administrador de Internet Information Services (IIS).

Expanda el Servidor en el que instaló la aplicación Ecommerce, y dentro de este último, expanda la carpeta Sitios Web:

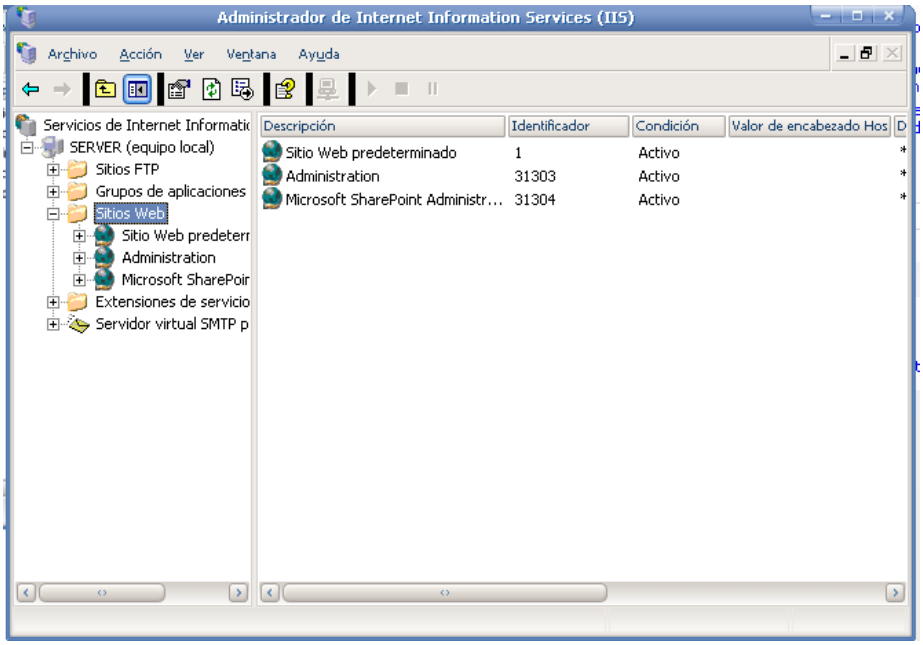


Figura 5.41. Administrador de Internet Information Services (IIS) con la carpetas Sitios Web expandida.

Haga clic derecho sobre la carpeta Sitios Web, en el menú que se despliega seleccione Nuevo y luego Sitio Web:

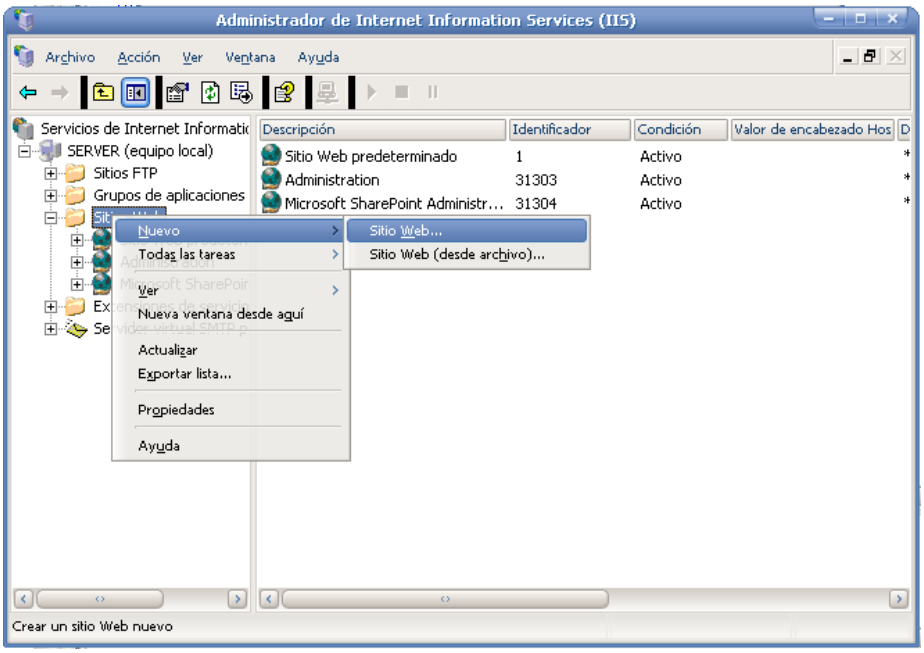


Figura 5.42. Administrador de Internet Information Services (IIS) Creación de un Nuevo Sitio Web.

Aparecerá el asistente para la creación de sitios Web:

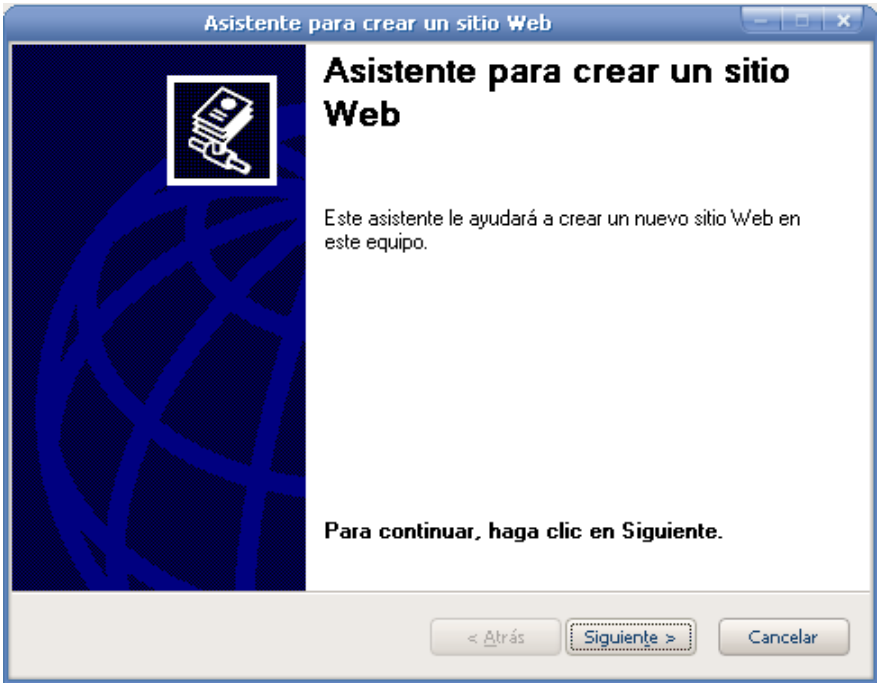


Figura 5.43. Asistente para crear un Sitio Web.

Haga clic en el botón Siguiente, aparecerá la pantalla que se muestra a continuación:

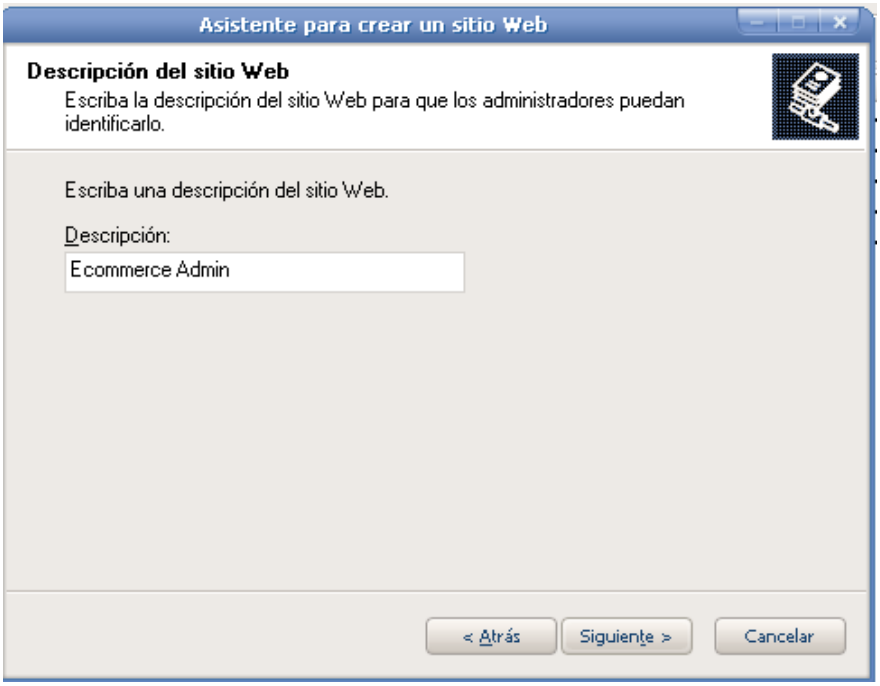


Figura 5.44. Asistente para crear un Sitio Web (Descripción del Sitio Web).

Se le solicitará una descripción para el nuevo sitio Web, escriba EcommerceAdmin y haga clic en Siguiente, aparecerá la pantalla que se muestra a continuación:

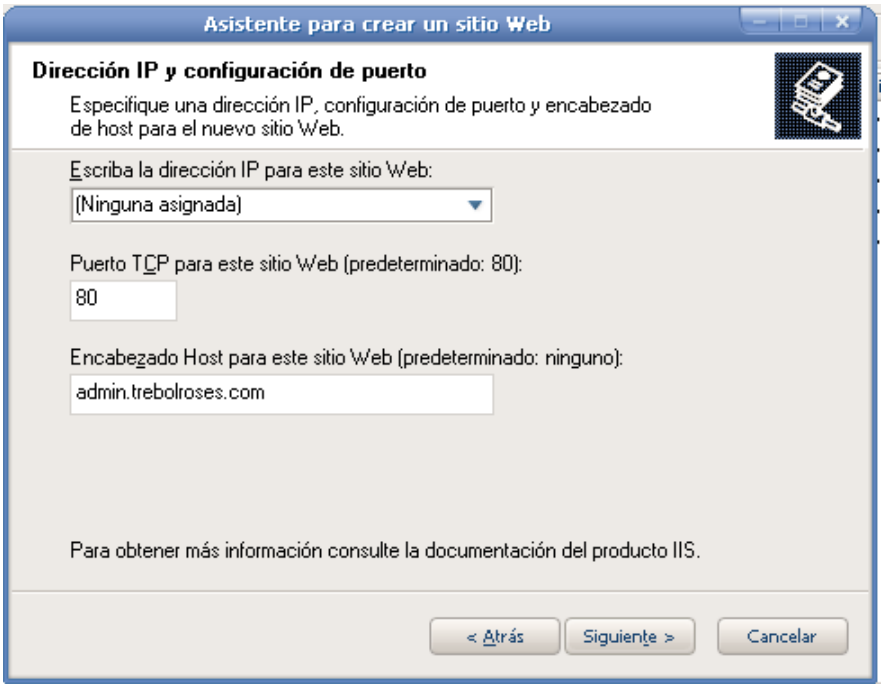


Figura 5.45. Asistente para crear un Sitio Web (Dirección IP y Configuración del Puerto).

En la dirección IP, digite la dirección que desee asignar al sitio Web, o deje el valor por defecto, en Puerto TCP, digite el puerto que desee asignar al sitio Web o deje el

valor por defecto, en Encabezado Host, escriba admin. Seguido del nombre de dominio, tal como se muestra en la figura anterior.

Una vez completada la información, haga clic en Siguiente, aparecerá la pantalla que se muestra a continuación:

Figura 5.46. Asistente para crear un Sitio Web (Directorio del Sitio Web).

Haga clic en el botón examinar y navegue hasta la carpeta %systemdrive%\Inetpub\wwwroot\EcommerceAdmin o hasta la carpeta en donde instaló Ecommerce.

Figura 5.47. Asistente para crear un Sitio Web (Buscar Carpetas).

Haga clic en el botón Aceptar y luego en Siguiente, aparecerá la pantalla que se muestra a continuación:

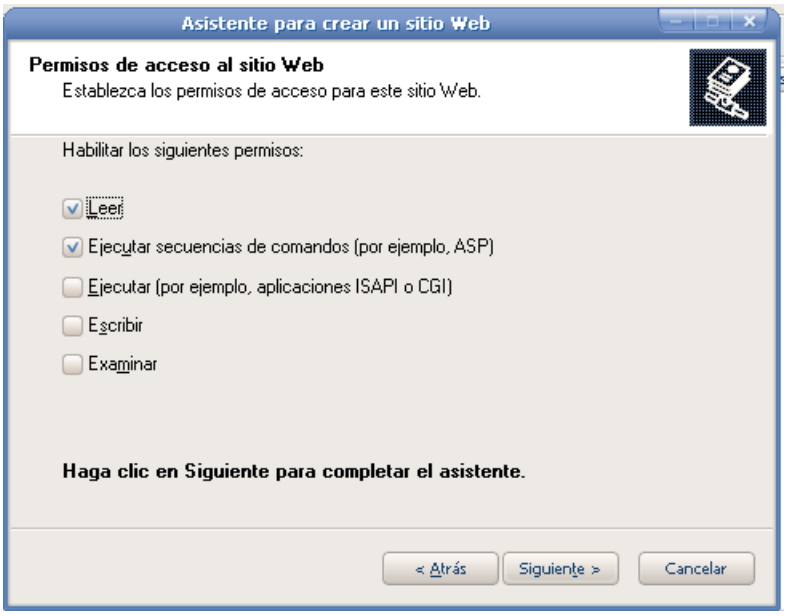


Figura 5.48. Asistente para crear un Sitio Web (Permisos de Acceso).

Deje los valores por defecto y haga clic en el botón Siguiente, aparecerá la pantalla que se muestra a continuación:

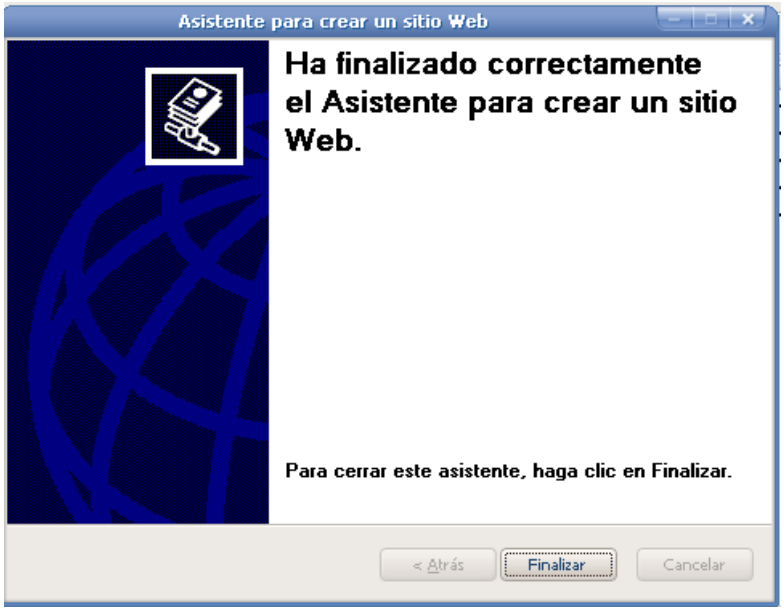


Figura 5.49. Asistente para crear un Sitio Web (Finalización de la Creación del Sitio Web).

Haga clic en Finalizar para concluir la configuración del sitio.

5.2.5.3. Módulo para los Servicios Web.

Para configurar este módulo, abra el administrador de Internet Information Services:

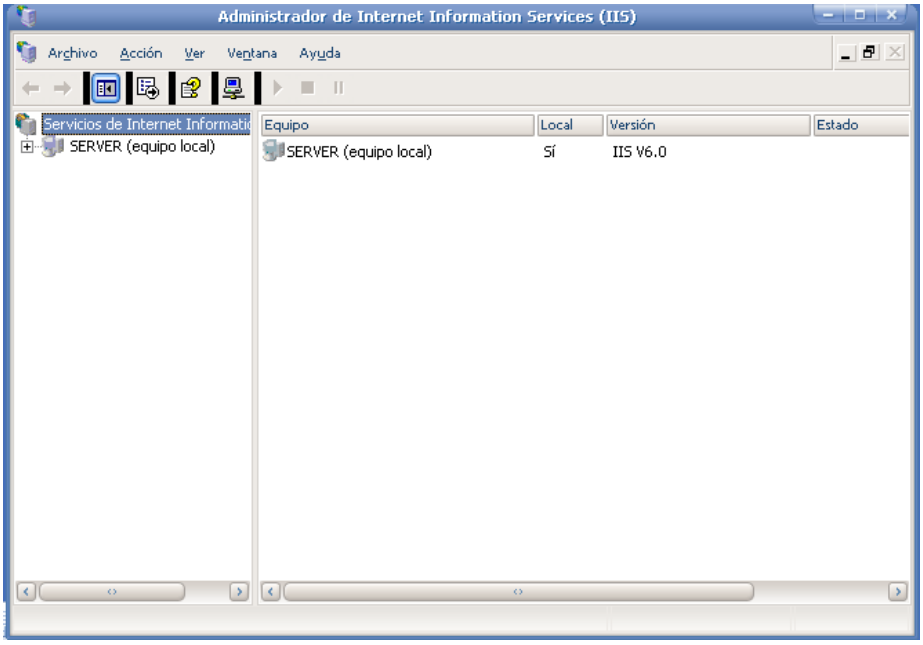


Figura 5.50. Administrador de Internet Information Services (IIS).

Expanda el Servidor en el que instaló la aplicación Ecommerce, y dentro de este último, expanda la carpeta Sitios Web:

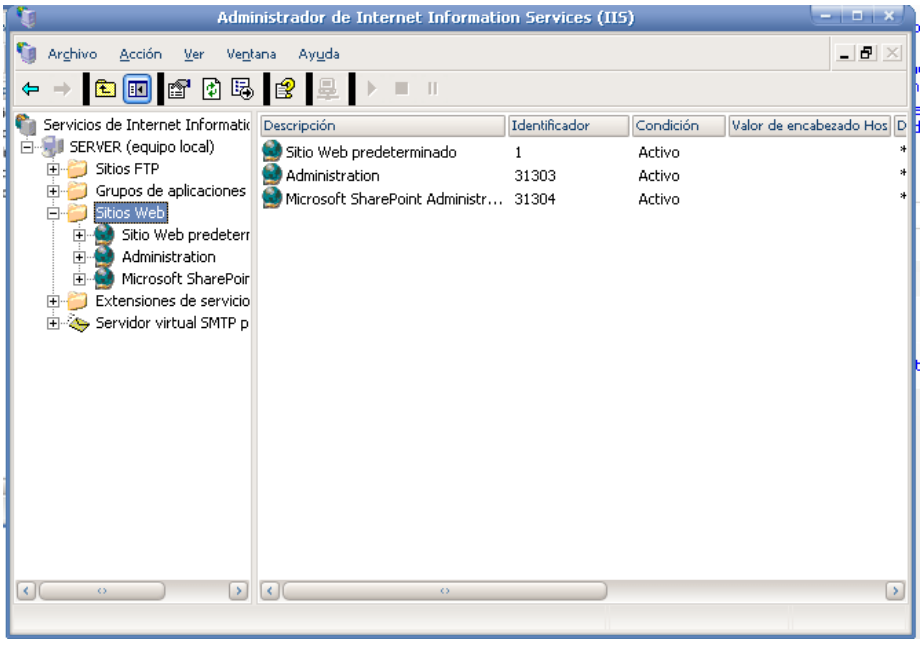


Figura 5.51. Administrador de Internet Information Services (IIS) con la carpeta Sitios Web expandida.

Haga clic derecho sobre la carpeta Sitios Web, en el menú que se despliega seleccione Nuevo y luego Sitio Web:

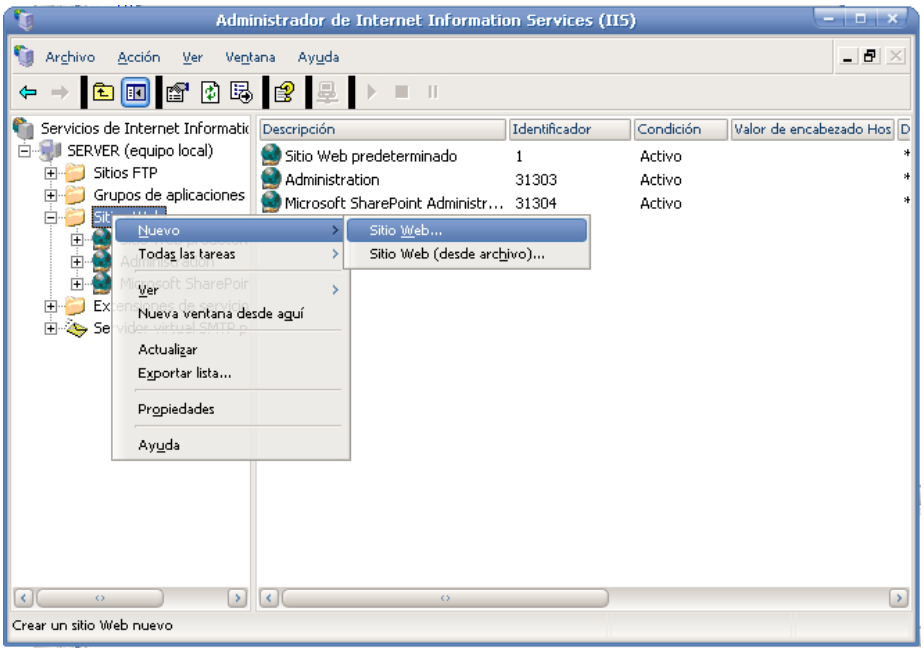


Figura 5.52. Administrador de Internet Information Services (IIS) Creación de un Nuevo Sitio Web.

Aparecerá el asistente para la creación de sitios Web:

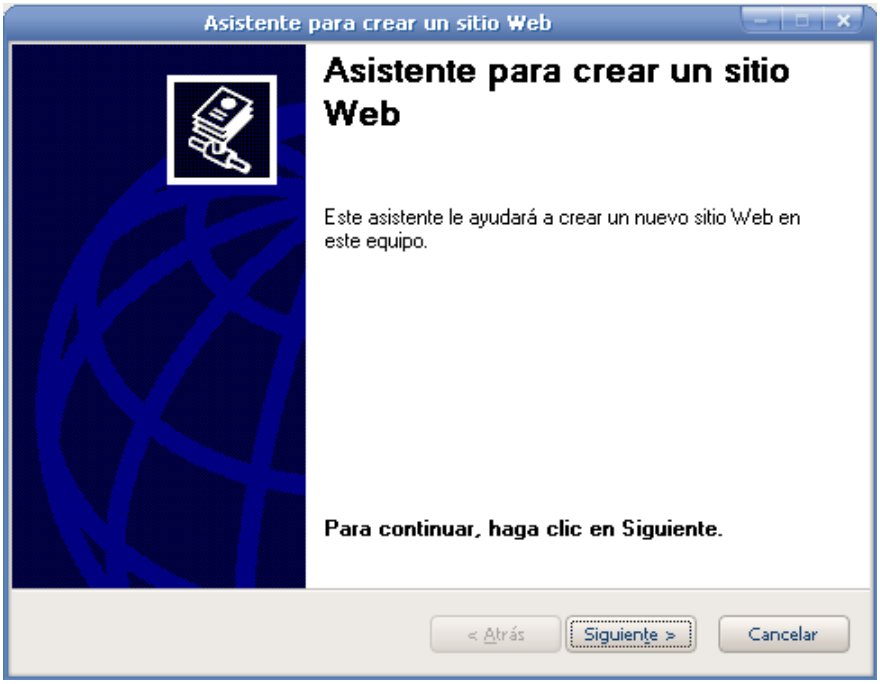


Figura 5.53. Asistente para crear un Sitio Web.

Haga clic en el botón Siguiente, aparecerá la pantalla que se muestra a continuación:

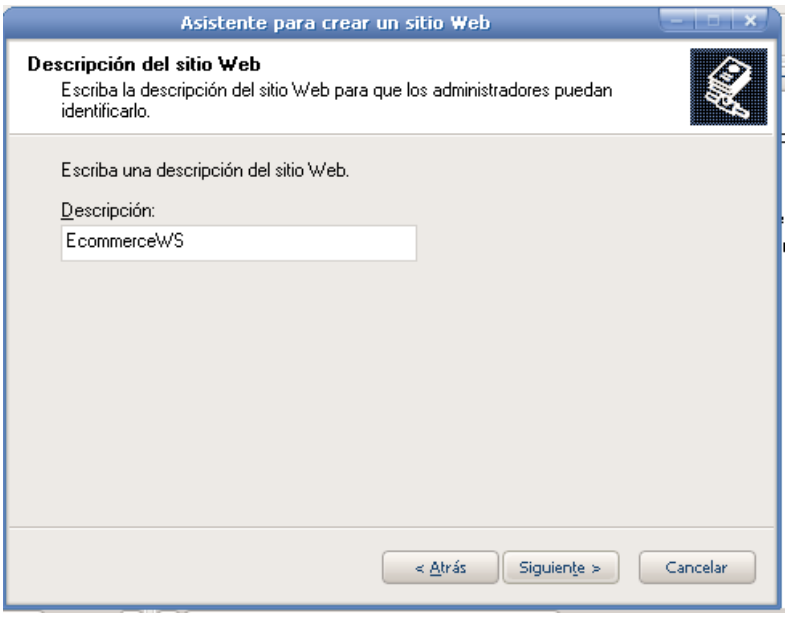


Figura 5.54. Asistente para crear un Sitio Web (Descripción del Sitio Web).

Se le solicitará una descripción para el nuevo sitio Web, escriba EcommerceWS y haga clic en Siguiente, aparecerá la pantalla que se muestra a continuación:

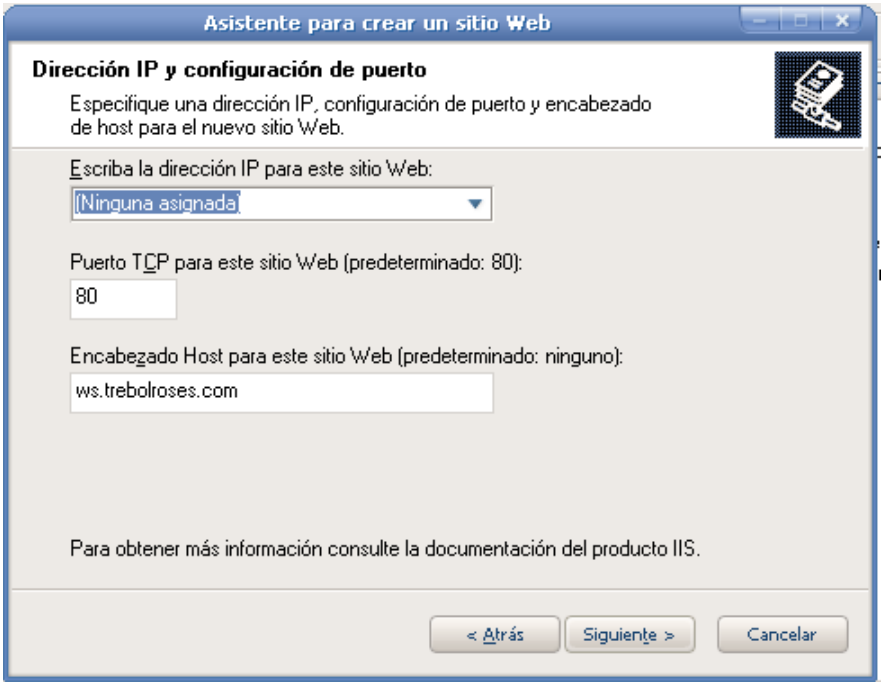


Figura 5.55. Asistente para crear un Sitio Web (Dirección IP y Configuración del Puerto).

En la dirección IP, digite la dirección que desee asignar al sitio Web, o deje el valor por defecto, en Puerto TCP, digite el puerto que desee asignar al sitio Web o deje el

valor por defecto, en Encabezado Host, escriba www. Seguido del nombre de dominio, tal como se muestra en la figura anterior.

Una vez completada la información, haga clic en Siguiente, aparecerá la pantalla que se muestra a continuación:

Figura 5.56. Asistente para crear un Sitio Web (Directorio del Sitio Web).

Haga clic en el botón examinar y navegue hasta la carpeta %systemdrive%\Inetpub\wwwroot\EcommerceWS o hasta la carpeta en donde instaló Ecommerce.

Figura 5.57. Asistente para crear un Sitio Web (Buscar Carpetas).

Haga clic en el botón Aceptar y luego en Siguiente, aparecerá la pantalla que se muestra a continuación:

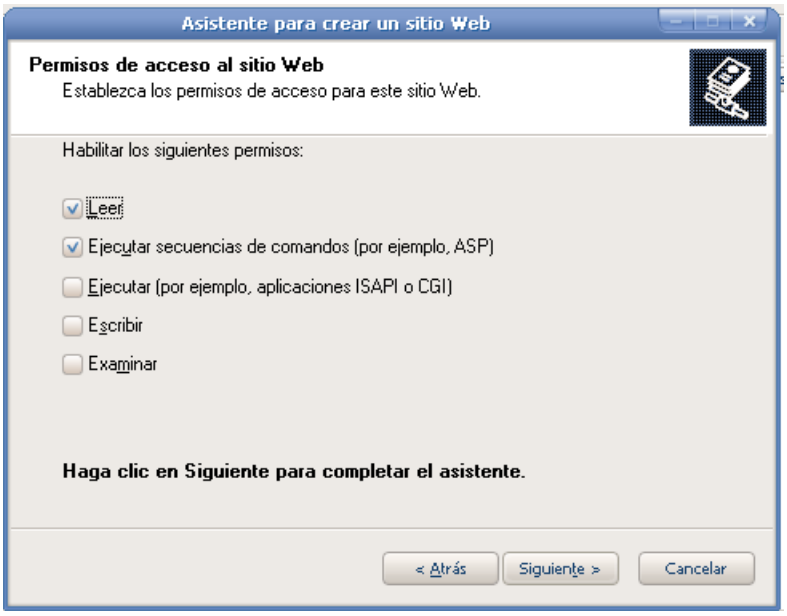


Figura 5.58. Asistente para crear un Sitio Web (Permisos de Acceso).

Deje los valores por defecto y haga clic en el botón Siguiente, aparecerá la pantalla que se muestra a continuación:

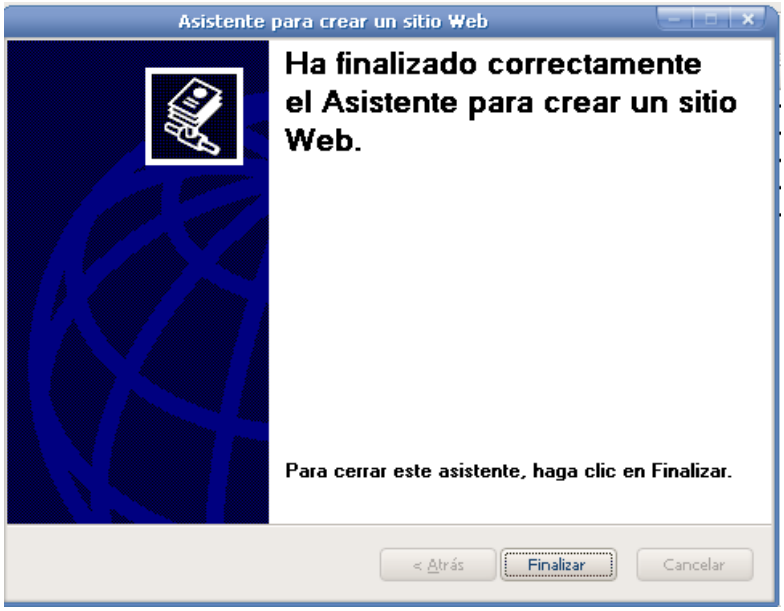


Figura 5.59. Asistente para crear un Sitio Web (Finalización de la Creación del Sitio Web).

Haga clic en Finalizar para concluir la configuración del sitio.

5.3. Pruebas de la Aplicación.

El siguiente diagrama muestra el ciclo de pruebas utilizado durante el desarrollo de E-Commerce:

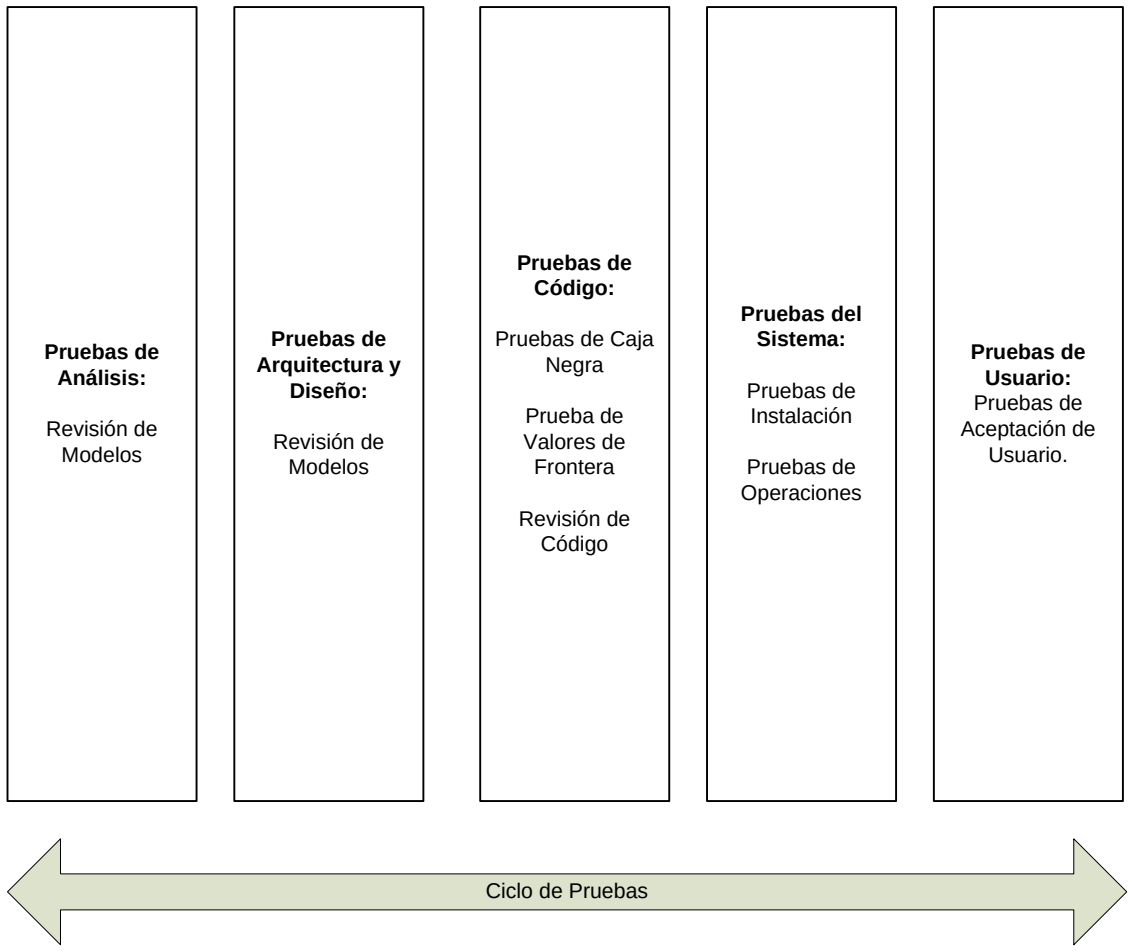


Figura 5.60. Ciclo de Pruebas de la Aplicación.

5.4. Corrección de los errores detectados.

El objetivo es encontrar los defectos de la aplicación lo antes posible durante el ciclo de desarrollo, de forma que el impacto sea el menor posible a la hora de corregirlos.

Durante las dos primeras fases (Pruebas de Análisis, Pruebas de Arquitectura y Diseño) , las pruebas se basaron específicamente en la revisión de los modelos planteados, de esta forma se pudieron corregir las inconsistencias que se tuvieron

durante la determinación de las necesidades del cliente, en este punto fue muy importante la participación de los usuarios proporcionados por el cliente.

Durante las Pruebas de Código, se utilizaron las siguientes técnicas:

- Pruebas de Caja Negra.- Se verificó pantalla por pantalla, que cada ítem, producía los resultados esperados al asignar los valores adecuados para las entradas necesarias.

- Pruebas de Valores de Frontera.- Se expusieron los elementos a situaciones extremas o inusuales para determinar que el ítem es capaz de manejarlas.

- Revisión del Código.- En este punto se corrigieron las inconsistencias detectadas en los puntos anteriores.

Las Pruebas de Instalación, estuvieron formadas por dos etapas:

- Pruebas de Instalación.- Se determinó que los métodos descritos en el apartado "Instalación de la Aplicación" eran correctos y funcionales.
- Pruebas de Operaciones.- Se determinó que la aplicación funcionaba correctamente después de haber realizado la instalación indicada en el punto anterior.

Durante las pruebas de Usuario, se logró confirmar el correcto funcionamiento de E-Commerce, lo cual se determinó mediante la aceptación de los mismos.

5.5. Sistema en producción.

Una vez superadas todas las etapas de este ciclo de desarrollo, y que se determinó que los métodos mencionados en este documento son correctos, se colocó definitivamente el sistema en producción. Logrando una aplicación con una correcta arquitectura y diseño en capas, y una aplicación adecuada de los conceptos de Ingeniería de Software, para obtener un máximo aprovechamiento de las ventajas del desarrollo orientado a objetos.

Capítulo 6.
Documentación

6.1 Manual Técnico.

6.1.1. Arquitectura de E-Commerce.

E-Commerce implementa la gestión de ventas de rosas a través del Internet. Cuando un nuevo cliente se suscribe, puede utilizar directamente todos los servicios necesarios para su funcionamiento. Estos servicios son ofrecidos de forma centralizada a través de Internet y se basan en servicios Web. De esta forma dicho cliente puede implementar en cualquier tecnología sus propios clientes consumidores de estos servicios o simplemente reutilizar los ya proporcionados.

Los servicios proporcionados por el servidor central cubren las principales necesidades:

- **Disponibilidad**: Ofrece un catálogo actualizado de productos, con la información detallada de existencia, tamaños y descripciones de cada uno de ellos.
- **Usuarios**: Gestiona los usuarios asociados con el sistema y que incluyen los clientes finales, los vendedores y los administradores, dicha funcionalidad está implementada mediante un conjunto de roles.

La Aplicación Web: Permite al usuario final acceder a la funcionalidad del sistema por medio de un navegador de Internet. Entre las operaciones más habituales que realizará el usuario están la consulta del catálogo de productos, la compra, revisión de su historial de compras entre otras.

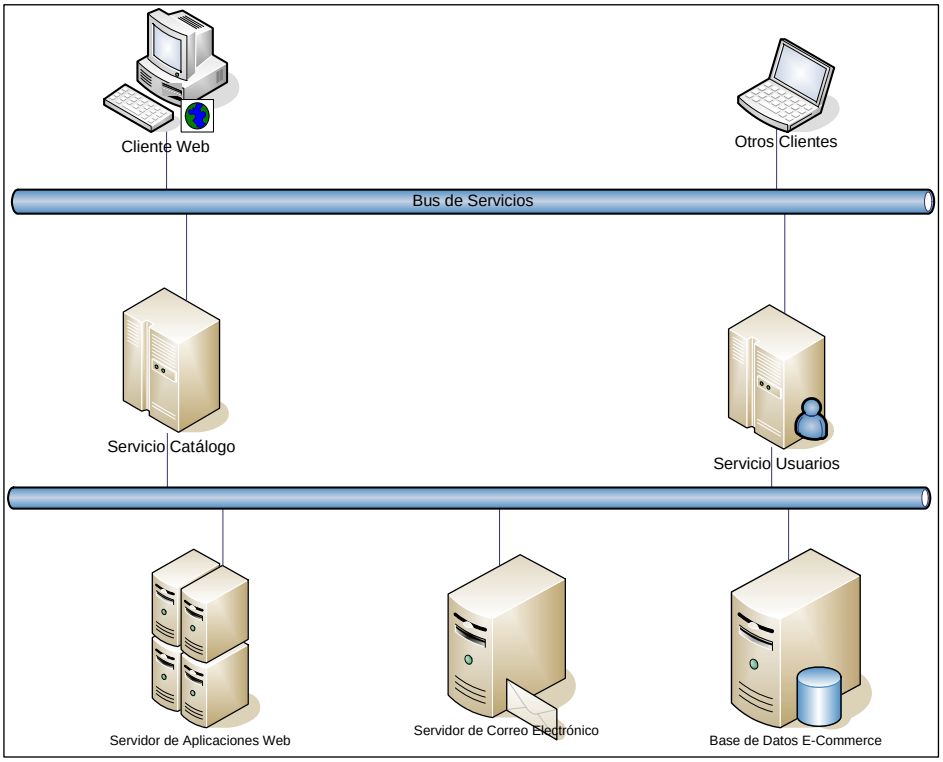


Figura 6.1. Arquitectura E-Commerce

6.1.2. Diseño en Capas.

Cada servicio del servidor central está formado por las tres capas típicas en una aplicación distribuida:

- **Capa de acceso a datos**: Es la responsable del acceso a las fuentes de datos del sistema. Independiza a la aplicación de las características, almacenamiento y estructura de los datos. Su labor principal es la de implementar las operaciones CRUD (creación, lectura, actualización y eliminación).
- **Capa de negocio**: Implementa la lógica de la aplicación, definida por las reglas que gobiernan el comportamiento del sistema. La capa de negocio expone todas las operaciones del sistema y es la única entrada posible para el uso del mismo.
- **Capa de presentación**: Expone las operaciones definidas en la capa de negocio en forma del cliente Web E-Commerce y a través de servicios Web, para permitir su consumo a los clientes del sistema y a otras aplicaciones que hagan uso de él.

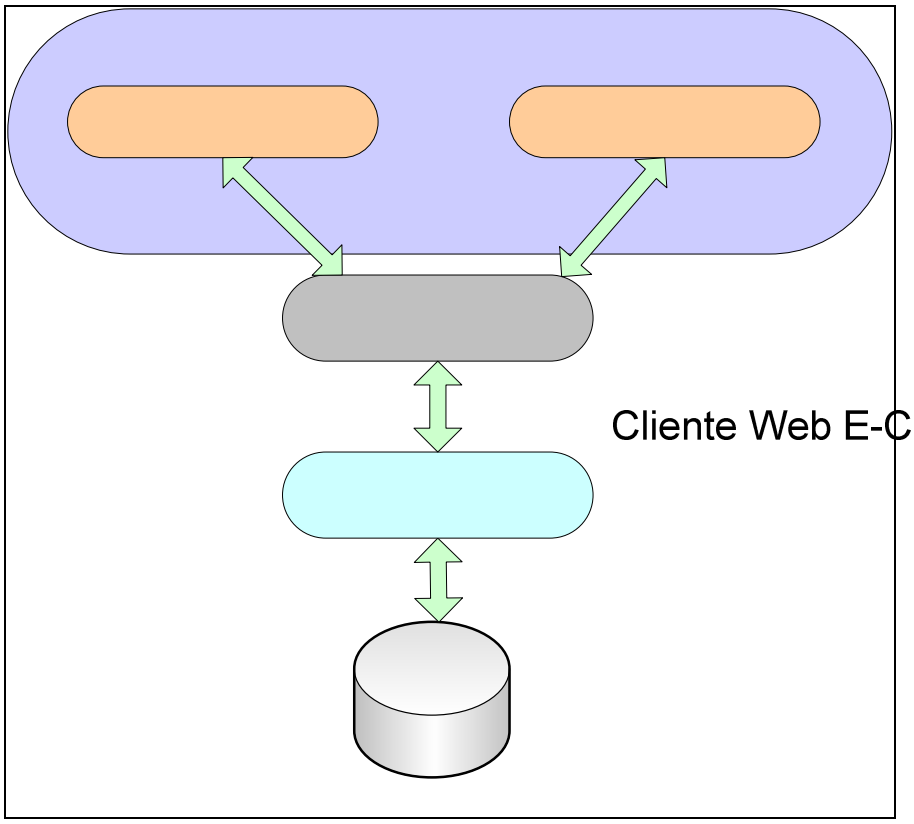


Figura 6.2. Diseño en capas de E-Commerce.

Para representar las entidades de negocio, se utilizaron los siguientes recursos de .NET:

- **Connection.-** Provee conectividad a un Origen de Datos.
- **Command.-** Provee acceso a comandos de Base de Datos como Select, Delete, Insert, Update y procedimientos almacenados.
- **DataSets ADO.NET**.- Un DataSet es una estructura de tablas en memoria, obtenidas de una base de datos. El acceso a un DataSet es sencillo y no necesita ningún procesamiento. Además el DataSet soporta las relaciones y reglas encontradas en las tablas de una base de datos y es capaz de recordar los cambios que las distintas capas hacen sobre él. Está formado por uno o más objetos de tipo **DataTables**, fue pensado para acceder a datos independientemente del origen.
- **DataAdapters ADO.NET.-** Utiliza el objeto Connection para enlazar un objeto DataSet con un Proveedor de Datos. También permite actualizar los Datos en el origen a partir de las modificaciones hechas en el DataSet.

- **DataReaders ADO.NET.-** Provee acceso a datos de solo lectura. Se utilizan para recuperar una gran cantidad de registros de un origen de datos el objeto DataTable puede usar demasiada memoria y recursos. El objeto DataReader permite usar menos recursos y acceder más rápidamente a los datos. El costo de esto es que puede ser recorrido únicamente hacia adelante y sus datos no pueden ser modificados Además la conexión al origen de datos debe hacerse en forma explícita.

6.1.3. Autenticación con la base de datos.

La aplicación funciona bajo el usuario de ASP.NET (*ASPNET* en Windows 2000/XP y *Network Service* en Windows 2003) que por defecto no tiene permisos para acceder a SQL Server, por lo que será necesario aumentar los permisos de dicho usuario en SQL Server para que pueda acceder a nuestra base de datos.

6.1.4. Paginación.

E-Commerce permite presentar grandes cantidades de registros (como en el historial de compras del cliente), utilizando un mecanismo de paginación. Este mecanismo es muy importante y permite ver los resultados página a página, con un número especificado (15 por página) de registros cada vez.

6.1.5. Métodos de negocio y mantenimiento.

La capa de negocio de E-Commerce implementa dos tipos distintos de métodos:

6.1.5.1. Métodos de mantenimiento: Son los métodos que exponen el mantenimiento básico CRUD (creación, lectura, actualización y borrado) de las entidades de negocio.

6.1.5.2. Métodos de negocio: Compuesto por el resto de métodos de la capa de negocio. Implementan funcionalidad más avanzada que el simple mantenimiento de las entidades e involucran varias entidades que cooperan entre sí. Por ejemplo el método FinalizarCompra de la clase de negocio OrdenCompra tiene que efectuar todas estas operaciones mediante los componentes de acceso a datos:

- Recuperar la información del cliente a partir de la entidad Cliente
- Recuperar la información de la agencia de carga a partir de la entidad AgenciaCarga

- Insertar la cabecera y el detalle de la orden a partir de la entidad CarroCompra

6.1.6. Validación de entidades.

Cuando una entidad es añadida o modificada en la capa de negocio, el sistema debe validar si su estructura y contenido son correctos. Se han utilizado las siguientes técnicas para lograr este cometido:

- **Validaciones en cliente**.- Muchas de las reglas que definen la estructura válida de una entidad pueden ser verificadas desde el propio cliente. Esto proporciona una enorme ventaja en la usabilidad del sistema ya que el usuario puede ser notificado de los problemas encontrados sin realizar una petición al servidor.
- **Validaciones en la base de datos.-** Definen la estructura de los datos y las relaciones entre ellos para que sea garantizada desde el propio almacenamiento.
- **Validaciones en la capa de negocio**.- Antes de empezar el procesamiento de cualquier método de negocio o mantenimiento deberemos comprobar que las instancias involucradas son válidas y cumplen la estructura exigida. Este tipo de validación se realiza al comienzo de cada método de la capa de negocio y no se permite la ejecución del resto si los datos no son válidos.

6.1.7. Control de concurrencia.

Por la propia naturaleza de los servicios web y de los componentes de la capa de negocio sin estado, todos los clientes accederán a E-Commerce de forma desconectada.

El proceso típico es el siguiente:

- Un cliente accede E-Commerce y recupera una o varias entidades de negocio.
- Las entidades pueden ser después modificadas en la máquina del cliente, modificando, añadiendo o borrando registros.
- Los cambios son enviados al servidor para que los actualice en la base de datos.
- El servidor utiliza el adaptador de datos correspondiente de la capa de acceso a datos para conciliar los cambios.

Durante todo este proceso no existe ninguna conexión entre el cliente y el servidor y por tanto tampoco se mantiene ningún estado, lo que incluye cualquier tipo de bloqueo en la base de datos. Esto significa que mientras se está modificando una entidad en el lado del cliente, los registros asociados en la base de datos del servidor están libres de ser leídos o modificados. Un segundo cliente podría llegar en ese momento, leer la misma entidad, modificarla y actualizar los cambios. Cuando el primer cliente actualizara sus cambios mancharía los del segundo cliente pudiendo provocar pérdidas de datos no controladas.

Aunque la mayoría de la gestión de control de concurrencia, la realiza SQL Server, en caso de producirse un error de este tipo, la configuración de los adaptadores de datos de E-Commerce, permiten solventar fácilmente estos inconvenientes, mediante la utilización de conjuntos de datos en modo optimista. Las apariciones de este tipo de errores de concurrencia dentro de un método de negocio son poco frecuentes, porque tienen que coincidir dos acciones simultáneas sobre la misma entidad de negocio. Además para resolverlo sólo es necesario reintentar la operación que ha dado el error y que esta vez no coincida con otra, lo que casi con toda probabilidad no ocurrirá. La solución pasa entonces por el concepto de reintentos.
Además E-Commerce implementa una metodología segura de tratamiento de datos en el servidor, mediante la utilización de procedimientos almacenados y transacciones en el almacén de datos, garantizando siempre la coherencia de la información.

6.1.8. Autenticación de usuarios.

Los métodos LoginCliente y LoginUsuario validan el nombre de usuario y la clave proporcionados por el llamante con la base de datos de usuarios y clientes. Si son correctos recupera el conjunto de role a los que pertenece el usuario y con toda esa información establece la identidad del hilo actual. A partir de ese momento cualquier llamada a la capa de negocio irá autenticada con el usuario proporcionado por el cliente y tendrá acceso a todas las funciones permitidas por su rol.

6.1.9. Autorización de usuarios

E-Commerce utiliza la autorización de usuarios basada en roles proporcionada por .NET. Esta autorización consiste en restringir el acceso a las funciones expuestas por la capa de negocio dependiendo de la identidad del llamante.

El entorno de ejecución de .NET mantiene en todo momento un objeto principal asociado al hilo de ejecución. Este objeto representa la identidad de un usuario que ha sido autenticado por el sistema y es del tipo "Personalizado", es decir implementado por clases propias de E-Commerce.

E-Commerce implementa las siguientes identidades de llamante:

- Administrador.- Usuarios autenticados dentro del rol Administrador, pueden realizar cualquier acción en el sistema.
- Vendedor.- Usuarios autenticados dentro del rol vendedor, pueden realizar mantenimiento de las ordenes de compra de los clientes asignados, y realizar consulta de la disponibilidad
- Público.- Usuarios autenticados dentro de cualquier rol.
- Anónimo.- Usuarios no autenticados, sólo pueden acceder a ciertos lugares sin restricción del sitio y no pueden realizar ninguna acción de negocio.

6.1.10. Cifrado de las claves de usuario.

Las contraseñas proporcionadas por los usuarios son una información muy sensible en cualquier sistema. Es muy importante proteger su envío a través de la red y su almacenamiento en la base de datos. E-Commerce utiliza el estándar MD5 para el cifrado de contraseñas, de forma que dicha información estará segura de cualquier intento de acceso sin autorización.
La lógica utilizada para aplicar la encriptación, se almacena en la clase CriptoDB, la misma que utiliza la implementación del algoritmo MD5 de .NET.

6.1.11. Monitorización del sistema.

Los sistemas operativos Windows ofrecen un mecanismo centralizado para controlar el funcionamiento de cualquier aplicación: el monitor de rendimiento (Performance Monitor). Puede localizar esta herramienta en el menú de Herramientas Administrativas del sistema o ejecutando el comando perform.

El monitor de rendimiento permite ver gráficamente la evolución de variables en el sistema o programar acciones y alarmas basadas en valores de las mismas. Estas variables se denominan contadores de rendimiento y el sistema proporciona un gran

número de ellos por defecto que permiten monitorizar el funcionamiento del sistema operativo.

Una aplicación puede exponer sus propios contadores de rendimiento. Por ejemplo pulse el símbolo "+" localizado en la barra de herramientas de la aplicación. En el desplegable Objeto de Rendimiento puede ver todos los contadores disponibles en el sistema, verá que existen contadores de aplicaciones que tenga instaladas en su máquina, como SQL Server, ASP.NET, etc.

Algunos contadores que se habilitaron para monitorear E-Commerce son:

6.1.11.1. ASP.NET Applications.

Sesiones Activas.- Permite conocer el número usuarios realizando actividad en el sistema.

Total de Sesiones.- Permite conocer el número total de sesiones en el sistema (activas + no activas)

6.1.11.2. SQL SERVER.

Transacciones Activas.- Permite analizar el grado de transaccionabilidad de la base de datos.

6.1.11.3. Procesador: % de tiempo: Permite evaluar el requerimiento de procesador de las tareas anteriores, de esta forma se puede conocer si el equipo satisface la demanda de la aplicación.

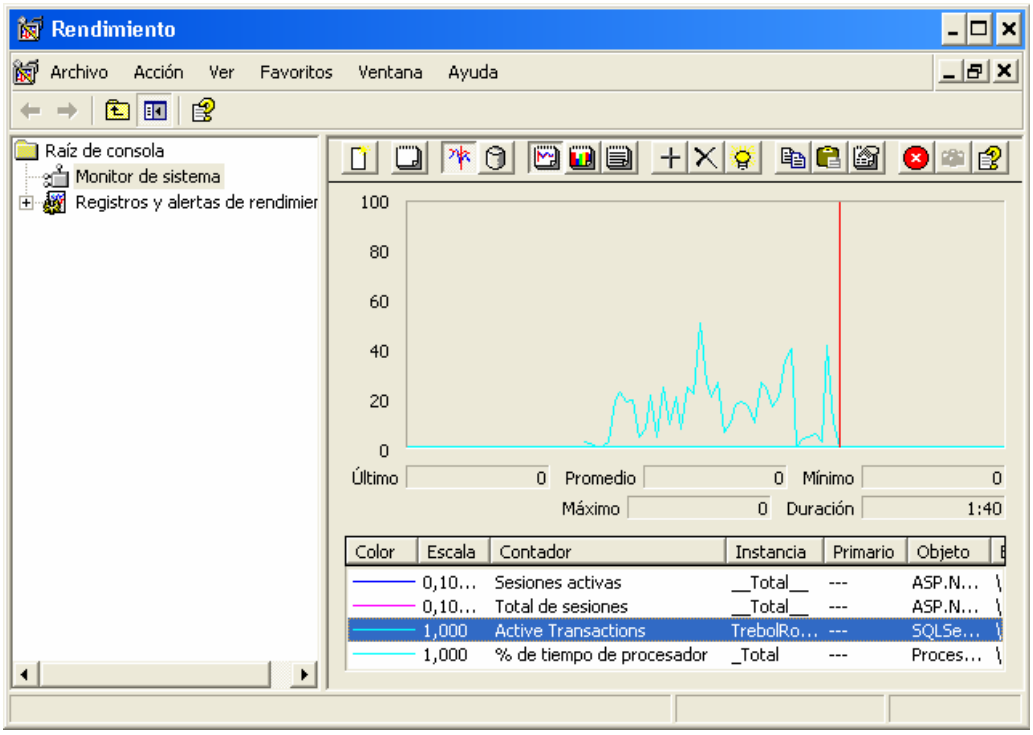


Figura 6.3. Monitorización de E-Commerce

6.1.12. Diseño de las páginas.

El cliente web de E-Commerce está formado por un conjunto de páginas ASP.NET que comparten la misma apariencia. Esta uniformidad es conseguida mediante hojas de estilo (CSS, Cascade Style Sheets) que definen de forma centralizada los estilos que se utilizan en todos los elementos gráficos de la aplicación.

Todos los estilos se encuentran definidos en el archivo styles.css, que forma parte de la solución.

6.1.12.1 Controles de usuario.

E-Commerce define varios controles de usuario ASP.NET que nos permitirán reutilizar funcionalidad entre las distintas páginas de la aplicación. Los controles desarrollados dentro de E-Commerce son los siguientes:

6.1.12.1.1. CtrlMenuSuperior2.- Encapsula el menú que ofrece la funcionalidad de consultar la cuenta del cliente, el acceso al formulario de contacto, y el acceso hacia el carro de compras. La apariencia del control es la siguiente:

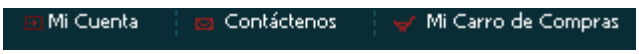


Figura 6.4. Control Menú superior 2.

6.1.12.1.2. CtrlMenuSuperior.- Encapsula el menú que proporciona el acceso hacia la página de inicio del sitio, el acceso a las promociones, consulta de órdenes, información sobre la empresa, acceso a la página de inicio de sesión y el cambio de idioma, ya que el sitio está diseñado en inglés y español . La apariencia es la siguiente:

Figura 6.5. Control Menú Superior.

6.1.12.1.3. CtrlEmail.- Permite la funcionalidad de registro del cliente para recibir notificaciones vía correo electrónico, su apariencia es:

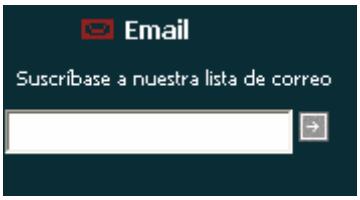


Figura 6.6. Control Email.

6.1.12.1.4. CtrlCategoriaProducto.- Permite visualizar las categorías de los productos, su apariencia es:

Figura 6.7. Control Categoría Producto.

6.1.12.1.5. CtrlLogin.- Permite la funcionalidad de inicio de sesión en el sistema, su apariencia es la siguiente:

Figura 6.8. Control Login.

6.1.12.1.6. CtrlBuscar.- Sirve para realizar la búsquela de una orden de compra por su código.

Figura 6.9. Control Buscar.

En todos estos controles se encapsula tanto la apariencia HTML como la lógica de presentación. La apariencia está definida en los ficheros .ascx, que tienen un formato muy parecido a las páginas .aspx.

6.1.13. Tratamiento de errores.

En aplicaciones web es muy importante no mostrar información detallada de los errores porque puede proporcionar una información muy valiosa a un usuario malicioso sobre nuestro código o nuestra arquitectura. Sin embargo en el proceso de desarrollo resulta muy útil la información proporcionada por defecto por ASP.NET, porque nos sirve para detectar la causa del error y localizar la fuente del mismo.

ASP.NET permite configurar las páginas de error según estemos en un entorno u otro y personalizar cada una de ellas con la salida que queramos. En E-Commerce se ha utilizado una aproximación mixta que nos servirá tanto para el entorno de desarrollo como para el de producción. Para ello se ha configurado la aplicación para que sólo se muestre información detallada si la petición se realiza desde la máquina local. Si la petición es de un cliente remoto se mostrará una página genérica de error diseñada con la misma apariencia que el resto de la aplicación.

Para conseguir este comportamiento utilizaremos el elemento customErrors dentro del archivo de configuración web.config:

```
<customErrors mode="RemoteOnly" defaultRedirect="frmErrores.aspx" />
```

La página `frmError.aspx` es la que contiene el mensaje genérico.

6.1.14. Globalización.

E-Commerce, está desarrollado en 2 idiomas diferentes, Español e Inglés, para seleccionar el idioma simplemente debe hacer clic en el menú superior, junto a la

opción "Cerrar Sesión", cuando la interfaz de usuario esté en Español, la opción aparecerá como "English", de lo contrario aparecerá como "Español"

Figura 6.10. Globalización.

6.2 Manual de Usuario.

6.2.1. Agencias de Carga.

En esta sección el Administrador del Sistema podrá Listar las Agencias de Carga que se hayan guardado en la Base de Datos, además de poder Ingresar, Modificar y Eliminar.

Figura 6.11. Listado de Agencias de Carga.

Al hacer clic en el Vínculo Nueva Agencia de Carga se presentara la siguiente pantalla:

Figura 6.12. Nueva Agencia de Carga.

En está pantalla deberá ingresar:

- RUC (Registro Único de Contribuyente).
- Nombre de la Agencia de Carga

Una vez que haya ingresado estos datos debe hacer un clic en el botón Guardar para que los datos se guarden en la Base de Datos, al realizar esto aparecerá la siguiente pantalla:

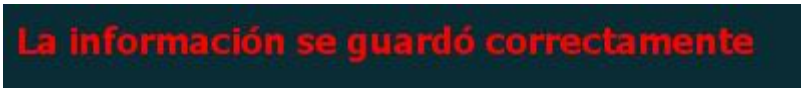


Figura 6.13. Datos Guardados.

Al hacer clic en el Icono , podrá acceder a la modificación de datos de la Agencia de Carga, aparecerá la siguiente pantalla:

Figura 6.14. Modificación de Agencias de Carga.

Aquí podrá modificar el nombre de la Agencia de Carga, una vez que haya ingresado los nuevos datos debe presionar el botón Guardar, aparecerá la siguiente pantalla:

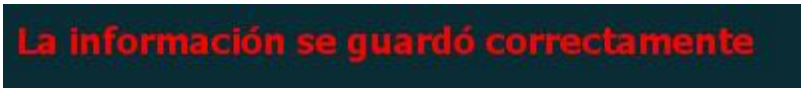


Figura 6.15. Datos Guardados.

Al hacer un clic en el Icono , podrá acceder a la eliminación de datos de la Agencia de Carga, aparecerá la siguiente pantalla:

Figura 6.16. Eliminación de Agencias de Carga.

Aquí podrá eliminar definitivamente los datos de la Agencia de Carga, si esta seguro de querer eliminar los datos debe presionar el botón Guardar, aparecerá la siguiente pantalla:

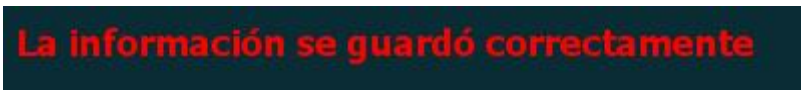


Figura 6.17. Datos Guardados.

6.2.2. Categorías de Productos.

En esta sección el Administrador del Sistema podrá Listar las Categorías de los Productos que se hayan guardado en la Base de Datos, además de poder Ingresar, Modificar y Eliminar.

Figura 6.18. Listado de Categoría de Productos.

Al hacer clic en el Vínculo Nueva Categoría de Producto se presentara la siguiente pantalla:

Figura 6.19. Nueva Categoría de Producto.

En está pantalla debe ingresar:

- Nombre de la Categoría del Producto.

Una vez que haya ingresado estos datos debe hacer un clic en el botón Guardar para que los datos se guarden en la Base de Datos, al realizar esto aparecerá la siguiente pantalla:

Figura 6.20. Datos Guardados.

Al hacer clic en el Icono , podrá acceder a la modificación de datos de la Categoría de Producto, aparecerá la siguiente pantalla:

Figura 6.21. Modificación de Categorías de Producto.

Aquí podrá modificar el nombre de la Categoría del Producto, una vez que haya ingresado los nuevos datos debe presionar el botón Guardar, aparecerá la siguiente pantalla:

Figura 6.22. Datos Guardados.

Al hacer un clic en el Icono ✕, podrá acceder a la eliminación de datos de la Categoría del Producto, aparecerá la siguiente pantalla:

Figura 6.23. Eliminación de Categoría de Productos.

Aquí podrá eliminar definitivamente los datos de la Categoría del Producto, si está seguro de querer eliminar los datos debemos presionar el botón Guardar, aparecerá la siguiente pantalla:

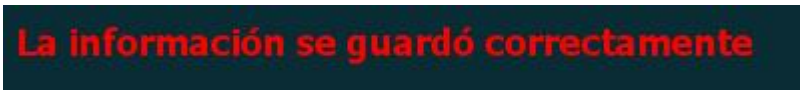


Figura 6.24. Datos Guardados.

6.2.3. Variedades de Productos.

En esta sección el Administrador del Sistema podrá Listar las Variedades de los Productos que se hayan guardado en la Base de Datos, además de poder Ingresar, Modificar y Eliminar.

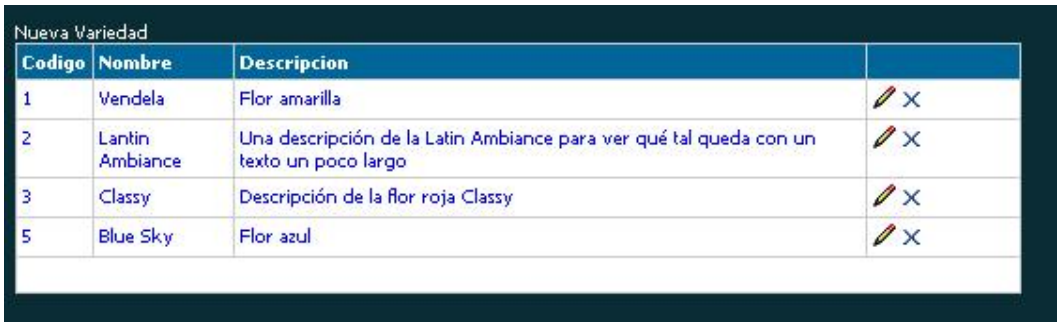


Figura 6.25. Listado de Variedades de Productos.

Al hacer clic en el Vínculo Nueva Variedad se presentara la siguiente pantalla:

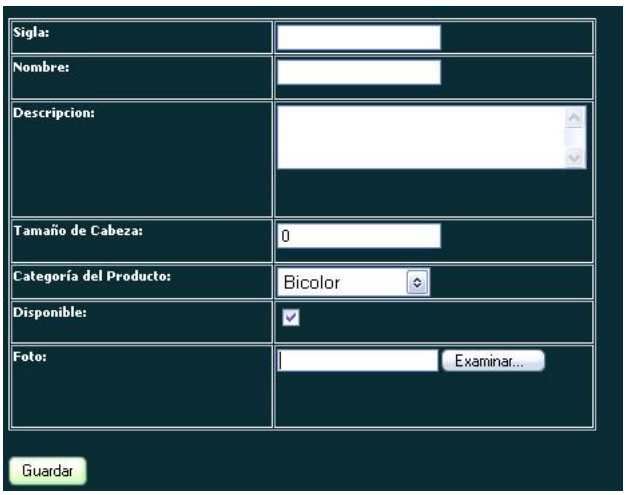


Figura 6.26. Nueva Variedad de Producto.

En está pantalla deberá ingresar:

- Sigla.
- Nombre de la Variedad.
- Descripción.
- Tamaño de Cabeza.
- Categoría del Producto a la cual pertenece la Variedad.
- Disponible.
- Foto.

Una vez que haya ingresado estos datos debemos hacer un clic en el botón Guardar para que los datos se guarden en la Base de Datos, al realizar esto aparecerá la siguiente pantalla:

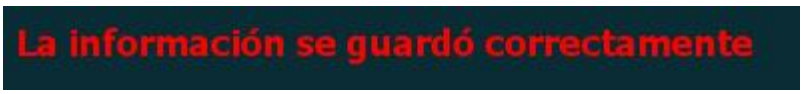


Figura 6.27. Datos Guardados.

Al hacer clic en el Icono , podrá acceder a la modificación de datos de la Variedad del Producto, aparecerá la siguiente pantalla:

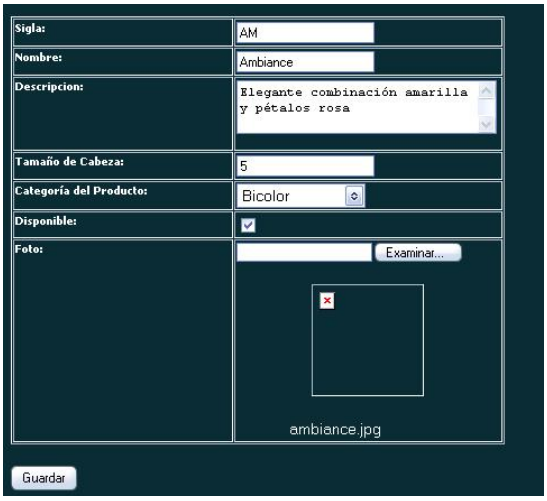


Figura 6.28. Modificación de Variedades de Productos.

Aquí podrá modificar la sigla, nombre, descripción, tamaño de cabeza, categoría del producto, disponibilidad y la imagen de la Variedad del Producto, una vez que haya ingresado los nuevos datos debe presionar el botón Guardar, aparecerá la siguiente pantalla:

Figura 6.29. Datos Guardados.

Al hacer un clic en el Icono , podrá acceder a la eliminación de datos de la Variedad del Producto, nos aparecerá la siguiente pantalla:

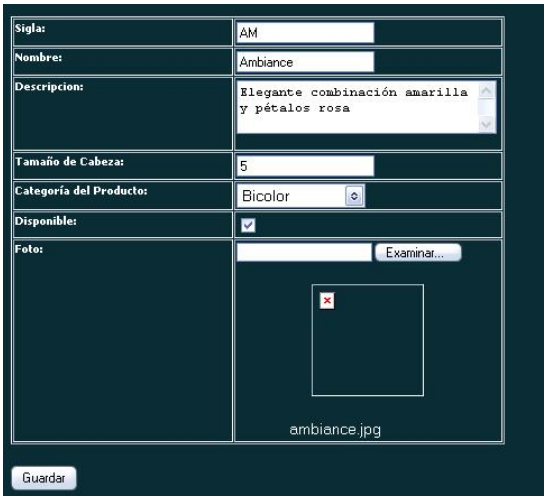


Figura 6.30. Eliminación de Variedades de Productos.

Aquí podrá eliminar definitivamente los datos de la Variedad del Producto, si está seguro de querer eliminar los datos debe presionar el botón Guardar, aparecerá la siguiente pantalla:

Figura 6.31. Datos Guardados.

6.2.4. Productos.

En esta sección el Administrador del Sistema podrá Listar los Productos que se hayan guardado en la Base de Datos, además de poder Ingresar, Modificar y Eliminar.

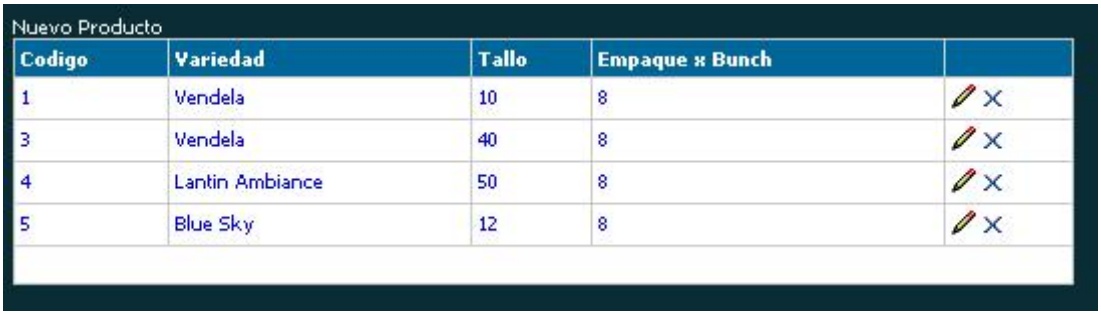


Figura 6.32. Listado de Productos.

Al hacer clic en el Vínculo Nuevo Producto se presentara la siguiente pantalla:

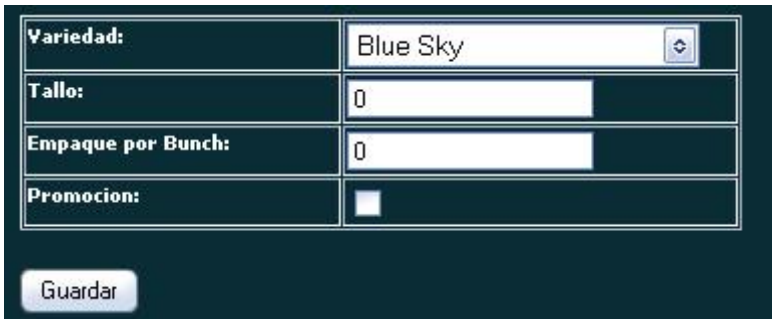


Figura 6.33. Nuevo Producto.

En está pantalla deberá ingresar:

- Variedad.
- Tamaño del Tallo.
- Empaque por Bunch.
- Promoción

Una vez que haya ingresado estos datos debe hacer un clic en el botón Guardar para que los datos se guarden en la Base de Datos, al realizar esto aparecerá la siguiente pantalla:

Figura 6.34. Datos Guardados.

Al hacer clic en el Icono , podrá acceder a la modificación de datos de Productos, aparecerá la siguiente pantalla:

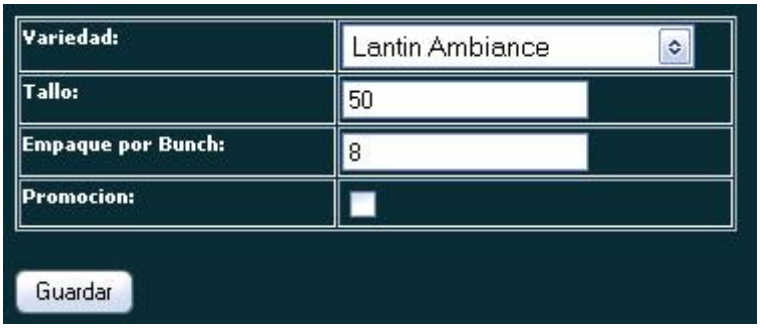


Figura 6.35. Modificación de Productos.

Aquí podrá modificar la variedad, el tamaño del tallo, el empaque por bunch, la promoción del Producto, una vez que haya ingresado los nuevos datos debe presionar el botón Guardar, aparecerá la siguiente pantalla:

Figura 6.36. Datos Guardados.

Al hacer un clic en el Icono , podrá acceder a la eliminación de datos del Producto, aparecerá la siguiente pantalla:

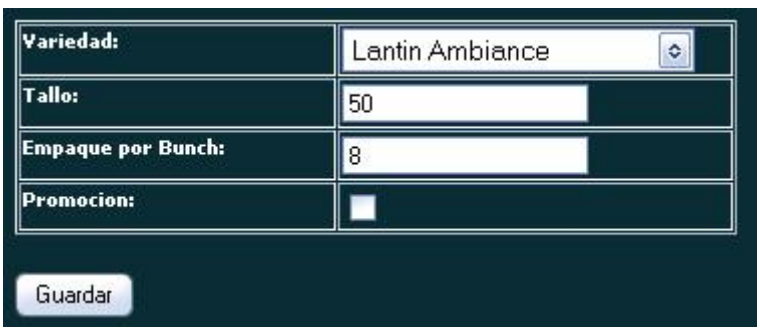


Figura 6.37. Eliminación de Productos.

Aquí podrá eliminar definitivamente los datos del Producto, si está seguro de querer eliminar los datos debe presionar el botón Guardar, aparecerá la siguiente pantalla:

Figura 6.38. Datos Guardados.

6.2.5. Tipos de Mercado.

En esta sección el Administrador del Sistema podrá Listar los Tipos de Mercado que se hayan guardado en la Base de Datos, además de poder Ingresar, Modificar y Eliminar.

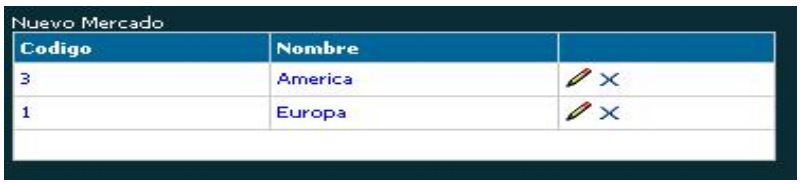


Figura 6.39. Listado de Tipos de Mercado.

Al hacer clic en el Vínculo Nuevo Mercado se presentara la siguiente pantalla:

Figura 6.40. Nuevo Tipo de Mercado.

En está pantalla deberá ingresar:

- Nombre del Mercado.

Una vez que haya ingresado estos datos debe hacer un clic en el botón Guardar para que los datos se guarden en la Base de Datos, al realizar esto aparecerá la siguiente pantalla:

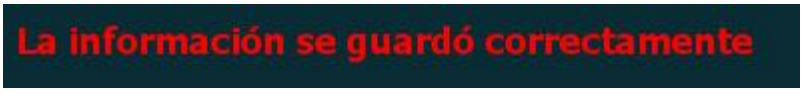


Figura 6.41. Datos Guardados.

Al hacer clic en el Icono , podrá acceder a la modificación de datos del Tipo de Mercado, aparecerá la siguiente pantalla:

Figura 6.42. Modificación de Tipos de Mercado.

Aquí podrá modificar el Nombre del Tipo de Mercado, una vez que haya ingresado los nuevos datos debe presionar el botón Guardar, aparecerá la siguiente pantalla:

Figura 6.43. Datos Guardados.

Al hacer un clic en el Icono ✕, podrá acceder a la eliminación de datos del Producto, aparecerá la siguiente pantalla:

Figura 6.44. Eliminación de Tipos de Mercado.

Aquí podrá eliminar definitivamente los datos del Tipo de Mercado, si está seguro de querer eliminar los datos debe presionar el botón Guardar, aparecerá la siguiente pantalla:

Figura 6.45. Datos Guardados.

6.2.6. Clientes.

En esta sección el Administrador del Sistema podrá Listar los Clientes de la empresa que se hayan guardado en la Base de Datos, además de poder Ingresar, Modificar y Eliminar.

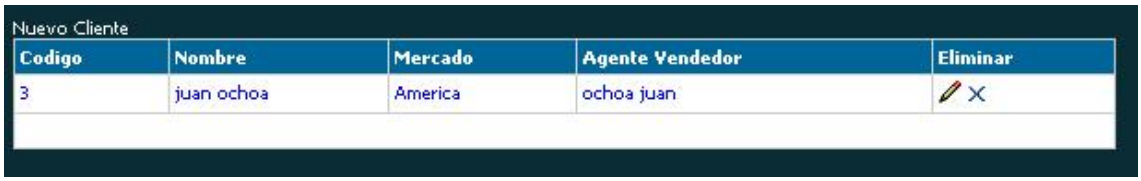


Figura 6.46. Listado de Clientes.

Al hacer clic en el Vínculo Nuevo Cliente se presentara la siguiente pantalla:

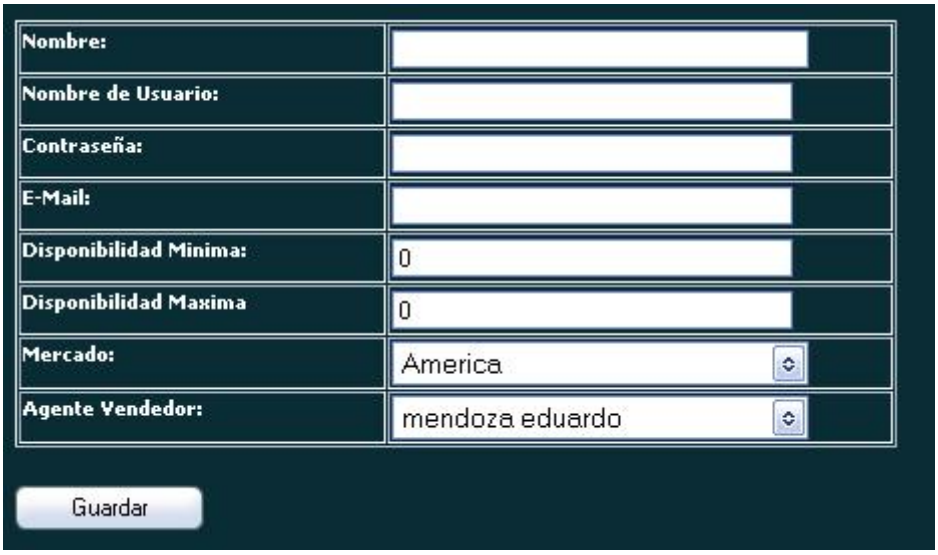


Figura 6.47. Nuevo Cliente.

En está pantalla deberá ingresar:

- Nombre.
- Nombre de Usuario.
- Contraseña.
- E-Mail.
- Disponibilidad Mínima.
- Disponibilidad Máxima.
- Mercado.
- Agente Vendedor.

Una vez que haya ingresado estos datos debe hacer un clic en el botón Guardar para que los datos se guarden en la Base de Datos, al realizar esto aparecerá la siguiente pantalla:

Figura 6.48. Datos Guardados.

Al hacer clic en el Icono , podrá acceder a la modificación de datos de Clientes, aparecerá la siguiente pantalla:

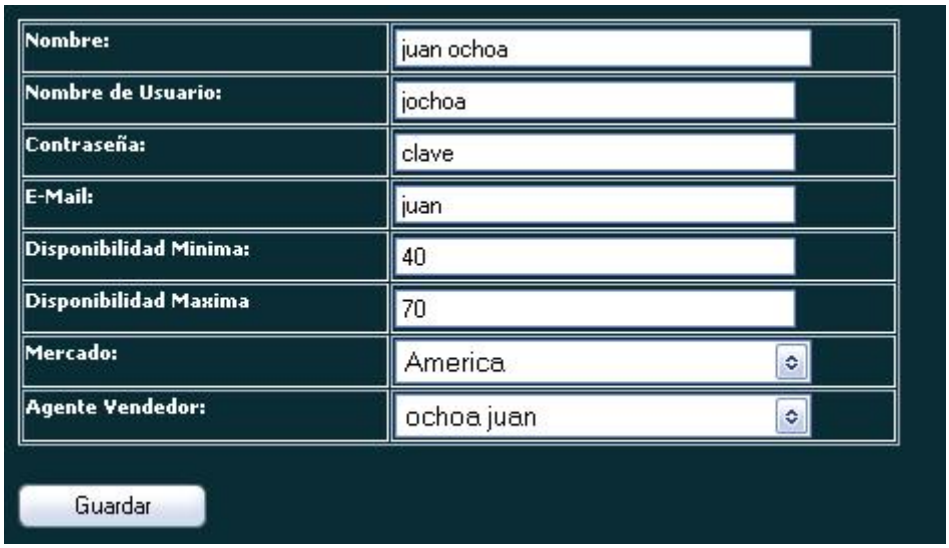


Figura 6.49. Modificación de Clientes.

Aquí podrá modificar el nombre, nombre de usuario, contraseña, e-mail, disponibilidad mínima, disponibilidad máxima, mercado, agente vendedor una vez que haya ingresado los nuevos datos debe presionar el botón Guardar, aparecerá la siguiente pantalla:

Figura 6.50. Datos Guardados.

Al hacer un clic en el Icono , podrá acceder a la eliminación de datos del Producto, aparecerá la siguiente pantalla:

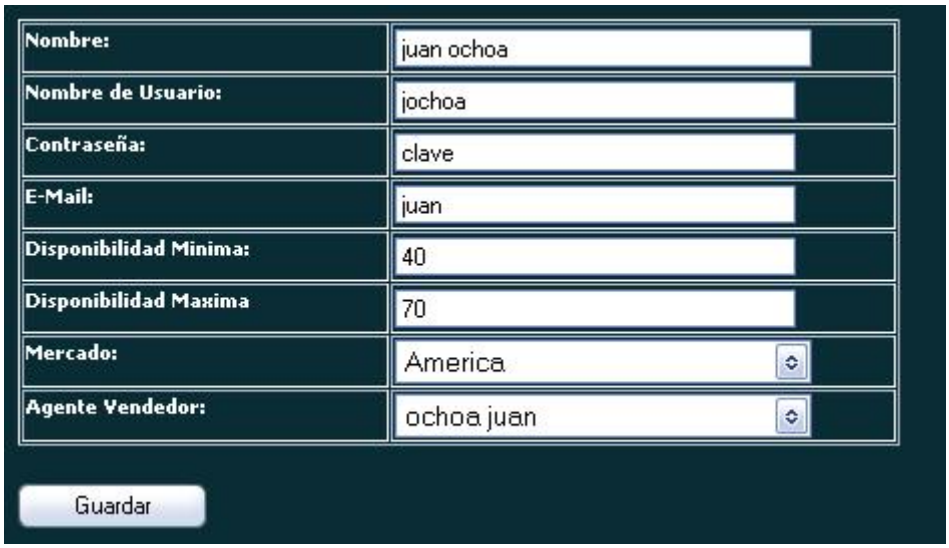


Figura 6.51. Eliminación de Clientes.

Aquí podrá eliminar definitivamente los datos del Cliente, si está seguro de querer eliminar los datos debe presionar el botón Guardar, aparecerá la siguiente pantalla:

Figura 6.52. Datos Guardados.

6.2.7. Disponibilidad.

En esta sección el Administrador del Sistema podrá ingresar las disponibilidades de cada uno de los Productos que posee la empresa para la venta.

Figura 6.53. Listado de Variedades.

En esta pantalla el Administrador del Sistema deberá escoger la variedad del producto de la cual desea ingresar la disponibilidad, una vez que haya escogido la variedad deberá hacer un clic en el botón Cargar, al realizar esto se presentara la siguiente pantalla:

Figura 6.54. Ingreso de Disponibilidad por Variedad.

En esta pantalla se presenta cada uno de los tamaños de los tallos de la variedad que se ha escogido previamente, aquí se debe ingresar los siguientes datos:

- Empaque x Caja.
- Existencia.
- Precio.

Una vez que haya ingresado estos datos en los tamaños de la Variedad que se necesite debe hacer clic en el botón Guardar, aparecerá la siguiente pantalla:

Figura 6.55. Datos Guardados.

6.2.8. Asignación de Roles.

En esta sección el Administrador del Sistema podrá asignar los Roles a cada uno de los usuarios del sistema, para que pueden tener acceso al mismo.

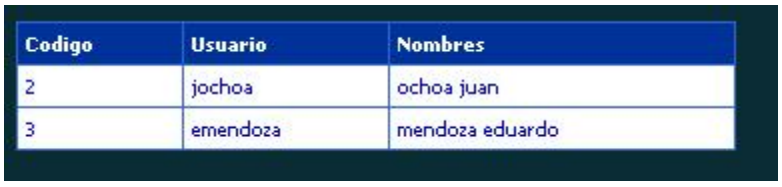
Figura 6.56. Asignación de Roles de Usuario.

En esta pantalla debe escoger el Usuario al cual deseamos asignar el Rol y el Tipo de Rol que se desea asignar, al realizar esta tarea debe hacer un clic en el botón Guardar, y aparecerá la siguiente pantalla:

Figura 6.57. Datos Guardados.

6.2.9. Roles de Usuario.

En esta sección el Administrador del Sistema podrá listar los Roles que han sido asignados a cada uno de los Usuarios del Sistema, para que los mismos puedan realizar las acciones que sean necesarias dentro del Sistema.



Codigo	Usuario	Nombres
2	jochoa	ochoa juan
3	emendoza	mendoza eduardo

Figura 6.58. Listado de Usuarios.

En esta pantalla se presenta un listado de Usuarios a los cuales ya se han asignado ciertos roles para el manejo del sistema, al hacer un clic en el Código del Usuario podrá acceder a un listado con los roles que se ha asignado al Usuario que se ha escogido:

Figura 6.59. Listado de Roles de Usuario Asignados.

En esta pantalla podrá acceder a la información de los Roles que han sido asignados a un Usuario determinado, si deseamos eliminar un cierto Rol a un usuario debe hacer un clic en el Ícono ✕, al realizar esto aparecerá la siguiente pantalla:

Figura 6.60. Eliminación de Roles.

Aquí podrá eliminar definitivamente los Roles de Usuario, si está seguro de querer eliminar los datos debe presionar el botón Guardar, aparecerá la siguiente pantalla:

Figura 6.61. Datos Guardados.

6.2.10. Usuarios.

En esta sección el Administrador del Sistema podrá Listar los Usuarios que hayan sido guardados en la Base de Datos, además de poder Ingresar, Modificar y Eliminar.

Figura 6.62. Listado de Usuarios.

Al hacer clic en el Vínculo Nuevo Usuario, se presentará la siguiente pantalla:

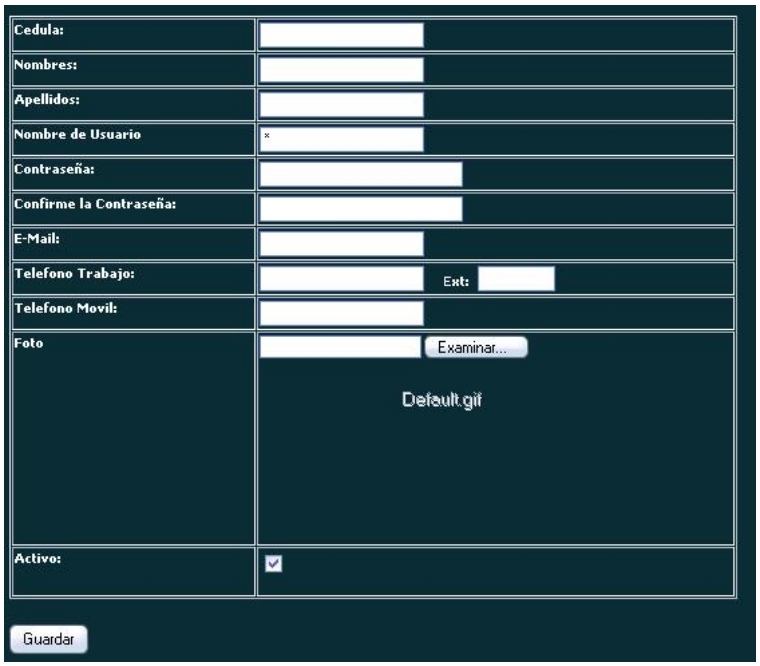


Figura 6.63. Nuevo Usuario.

En esta pantalla debe ingresar:

- Cedula.
- Nombres.
- Apellidos.
- Nombre de Usuario.
- Contraseña.
- E-Mail.
- Teléfono del Trabajo.
- Extensión.
- Teléfono Móvil.
- Foto.
- Activo.

Una vez que haya ingresado estos datos debe hacer un clic en el botón Guardar para que los datos se guarden en la Base de Datos, al realizar esto aparecerá la siguiente pantalla:

Figura 6.64. Datos Guardados.

Al hacer clic en el Icono , podrá acceder a la modificación de datos de Usuarios, aparecerá la siguiente pantalla:

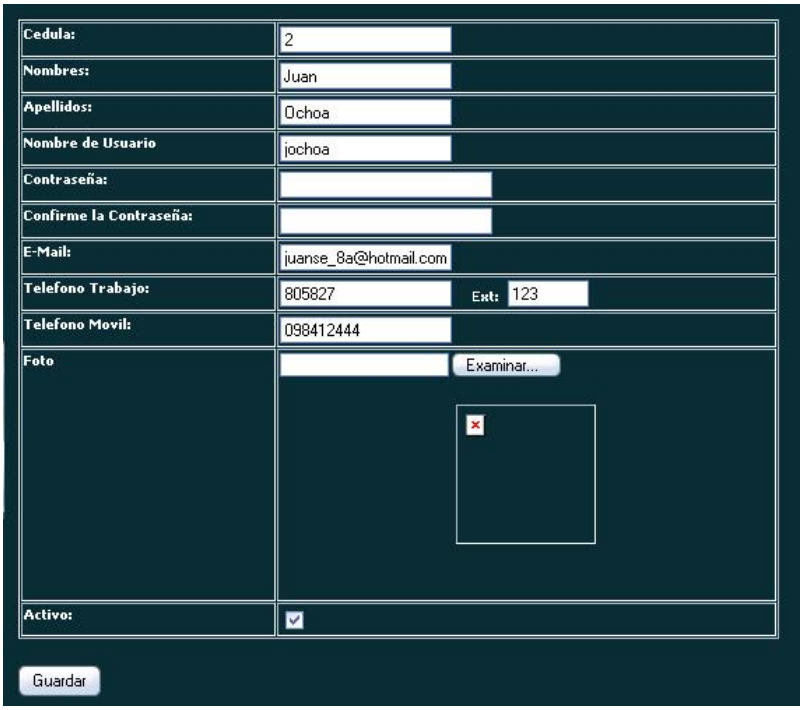


Figura 6.65. Modificación de Usuarios.

Aquí podrá modificar los nombres, apellidos, nombre de usuario, contraseña, e-mail, teléfono del trabajo, extensión, teléfono móvil, foto, activo, una vez que haya ingresado los nuevos datos debe presionar el botón Guardar, aparecerá la siguiente pantalla:

Figura 6.66. Datos Guardados.

Al hacer un clic en el Icono , podrá acceder a la eliminación de datos del Producto, aparecerá la siguiente pantalla:

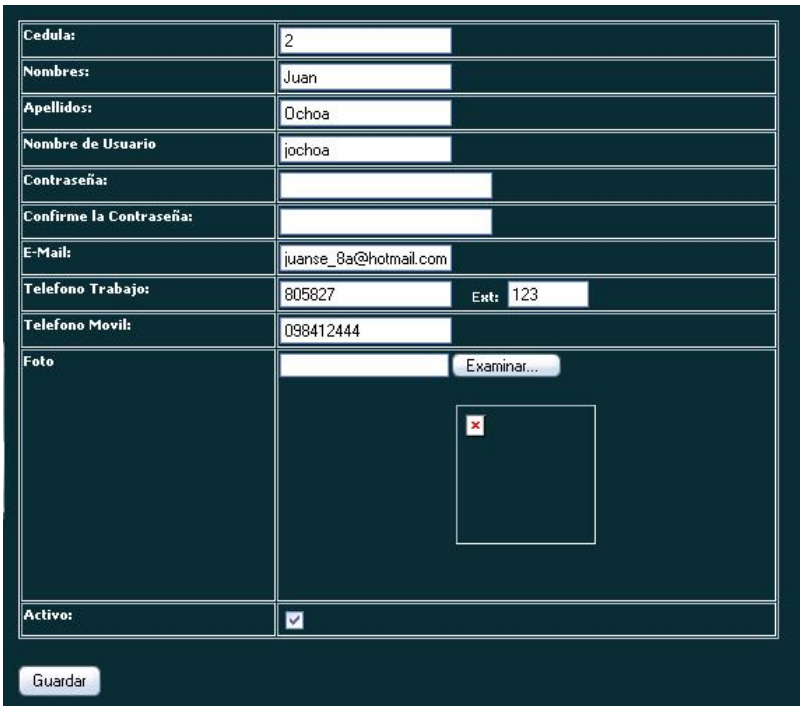


Figura 6.67. Eliminación de Usuarios.

Aquí podrá eliminar definitivamente los datos del Usuario, si está seguro de querer eliminar los datos debe presionar el botón Guardar, aparecerá la siguiente pantalla:

Figura 6.68. Datos Guardados.

6.2.11. Suscripciones de Usuario.

En esta sección el Administrador del Sistema podrá Activar o Desactivar las suscripciones realizadas por los Usuarios del Sistema.

Figura 6.69. Listado de Suscripciones de Usuario.

Al hacer clic en el Código de la Suscripción podrá activar o desactivar la Suscripción realizada por alguno de los Usuarios del Sistema, al realizar esto se nos presentara la siguiente pantalla:

Figura 6.70. Activación Desactivación de Suscripciones de Usuario.

Aquí podrá Activar o Desactivar la Suscripción realizada por el Usuario con tan solo hacer clic en la casilla de verificación de Activo, una vez que haya Activado o Desactivado la Suscripción debe hacer un clic en el botón Guardar, aparecerá la siguiente pantalla:

Figura 6.71. Datos Guardados.

6.2.12. Pedidos.

En esta sección los Clientes de la Empresa podrán realizar sus pedidos de acuerdo a sus necesidades.

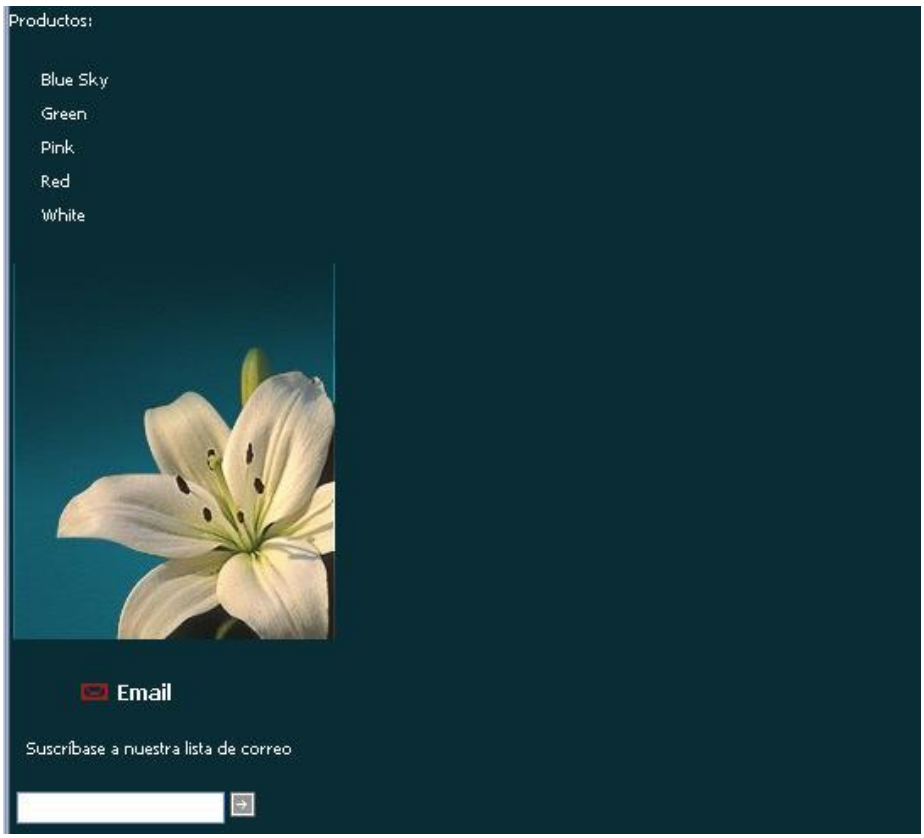


Figura 6.72. Listado de Variedades de Productos.

El Cliente deberá escoger los Productos con los que desea realizar su pedido, al realizar esto aparecerá la siguiente pantalla:

Figura 6.73. Variedades de Producto con sus respetivos detalles.

Aquí el Cliente deberá escoger la Variedad del Producto que desea comprar, se debe hacer un clic en la Imagen de la Variedad o en el Vínculo Comprar, al realizar esto aparecerá la siguiente pantalla:

Figura 6.74. Login.

Aquí se deberá ingresar el Nombre de Usuario y Contraseña para poder realizar el Pedido del Producto, si los datos han sido correctamente ingresados aparecerá la siguiente pantalla:

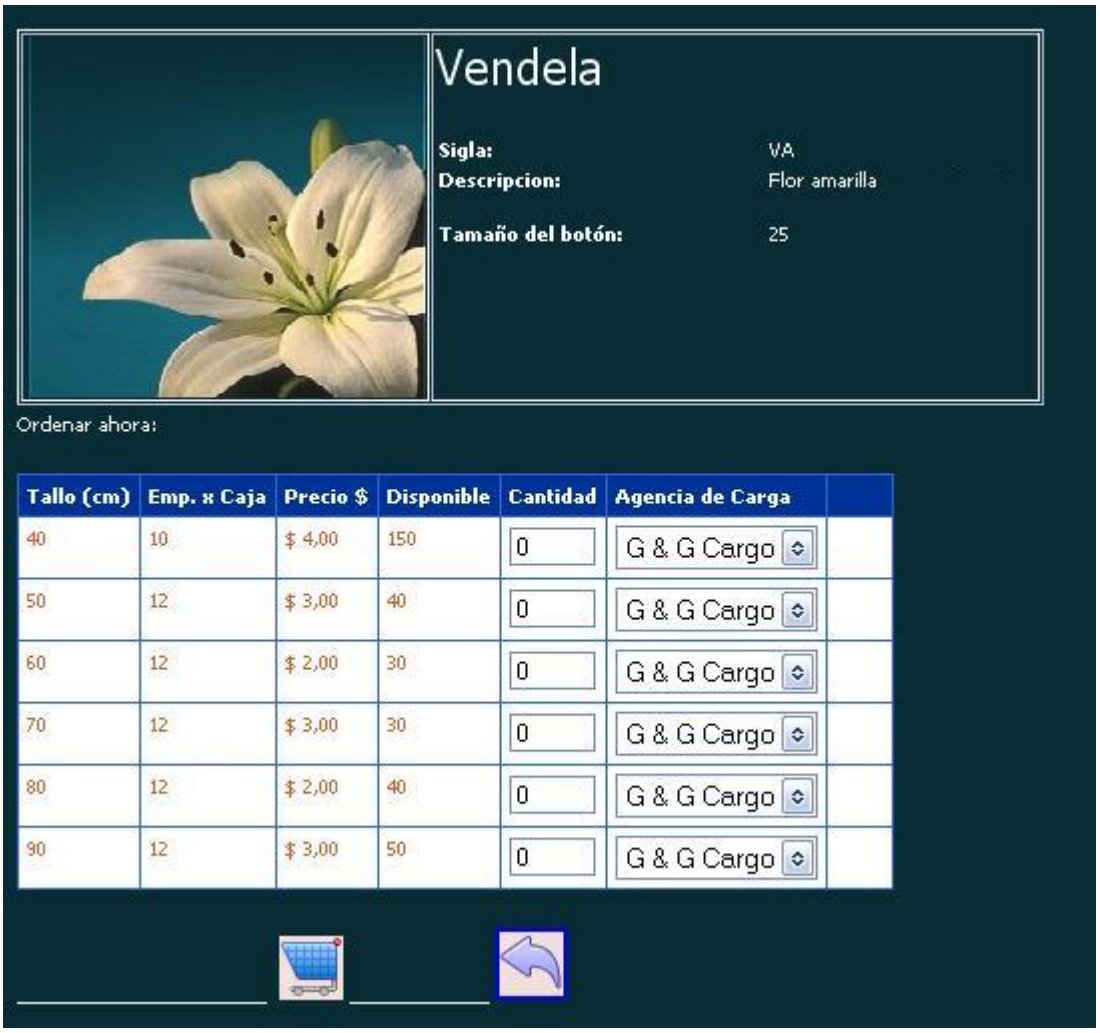


Figura 6.75. Variedad de Producto con sus respectivos tallos.

En esta pantalla se muestra los detalles del Producto que se desea comprar, además el Cliente deberá ingresar las cantidades que desea comprar de este producto así como la Agencia de Viajes por la cual desea que el Producto sea enviado, una vez que se haya realizado esto debe hacer un clic en  , aparecerá la siguiente pantalla:

Figura 6.76. Carro de Compras.

Aquí el Cliente podrá ver el Total de su Pedido, Quitar alguno de los Productos que está adquiriendo, modificar las cantidades de los Productos que está comprando, si cambia alguno de los datos del pedido deberá hacer un clic en el vínculo Actualizar Pedido, caso contrario deberá hacer un clic en el Vínculo Finalizar Compra, aparecerá la siguiente pantalla:

Figura 6.77. Confirmación Cancelación de la Orden.

Aquí el Cliente podrá Confirmar la Orden o Cancelar la misma, para confirmar la orden deberá hacer clic en el vínculo Confirmar la Orden, aparecerá la siguiente pantalla:

Figura 6.78. Detalle de la Orden.

En caso de que el Cliente haya hecho un clic en el vínculo Cancelar la Orden, aparecerá la siguiente pantalla:

Figura 6.79. Carro de Compras Vacío.

6.2.13. Historial de Compras.

En esta sección el Cliente podrá acceder a la información de todas las compras que haya realizado en la empresa.

Figura 6.80. Historial de Compras.

Para acceder a la información del pedido se deberá hacer un clic en el vínculo Ver detalles del número de orden que se desee ver la información, al realizar esto aparecerá la siguiente apantalla:

Figura 6.81. Detalle de la Orden.

En esta pantalla se presentará toda la información de la orden que el Cliente haya realizado.

6.2.14. Consultar Orden de Compra.

En esta sección el cliente podrá acceder a la información de una orden de compra específica.

6.82. Consulta de Orden de Compra

Aquí el Cliente deberá ingresar el número de orden de compra de la cual desea ver la información y luego hacer clic en  , aparecerá la siguiente pantalla:

Figura 6.83. Detalle de la Orden.

6.2.15. Mensajes.

En esta sección tanto el Cliente, los Vendedores y el Administrador del Sistema podrán realizar el envío de mensajes.

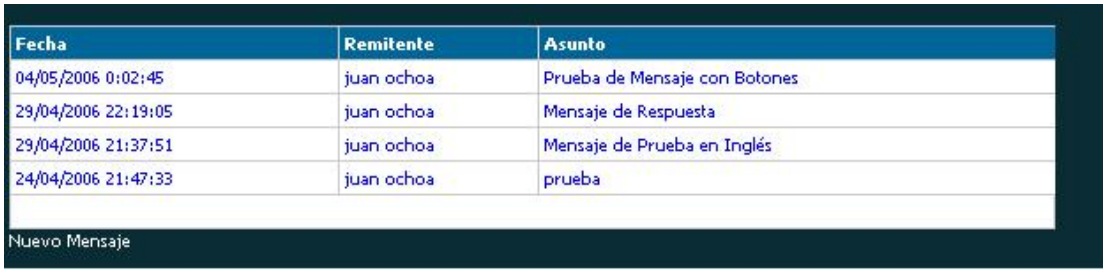


Figura 6.84. Lista de Mensajes.

Aquí el Cliente, Vendedor o Administrador del Sistema podrá visualizar los mensajes que le han enviado, para leer uno de los mensajes deberá hacer clic en el vínculo que contiene la fecha en la que el mensaje ha sido enviado, al realizar esta

acción en caso de ser un Vendedor o Administrador del Sistema aparecerá la siguiente pantalla:

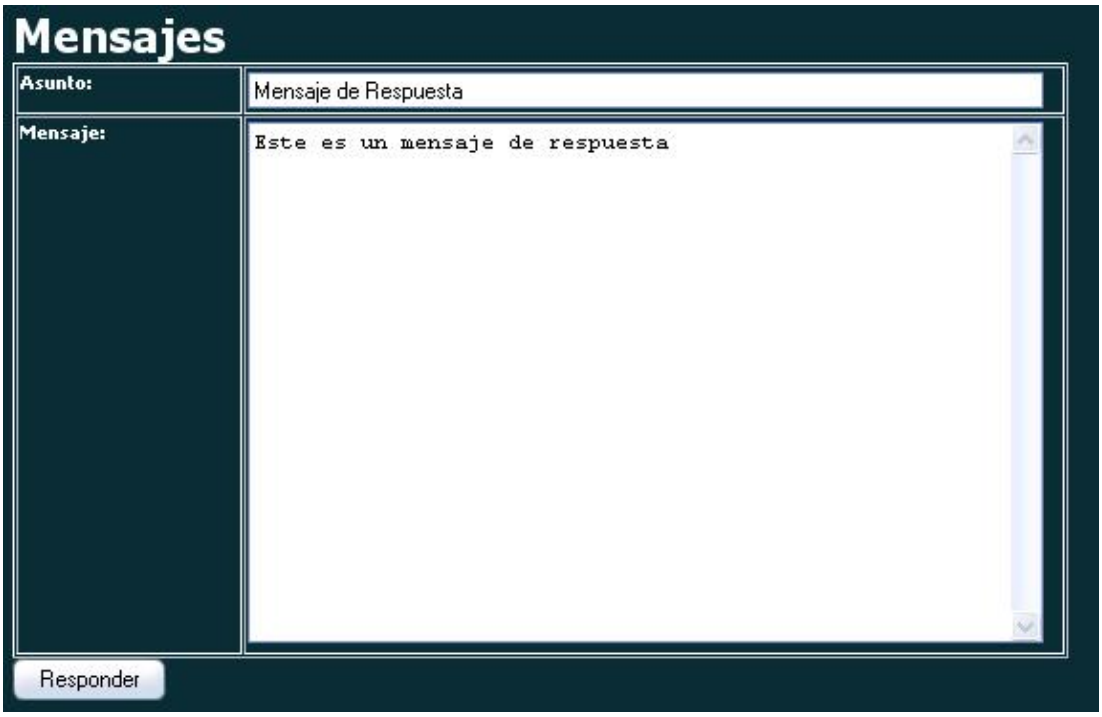


Figura 6.85. Revisión de Mensajes.

Si se desea responder el mensaje deberá hacer clic en el botón Responder, al realizar está acción aparecerá la siguiente pantalla:

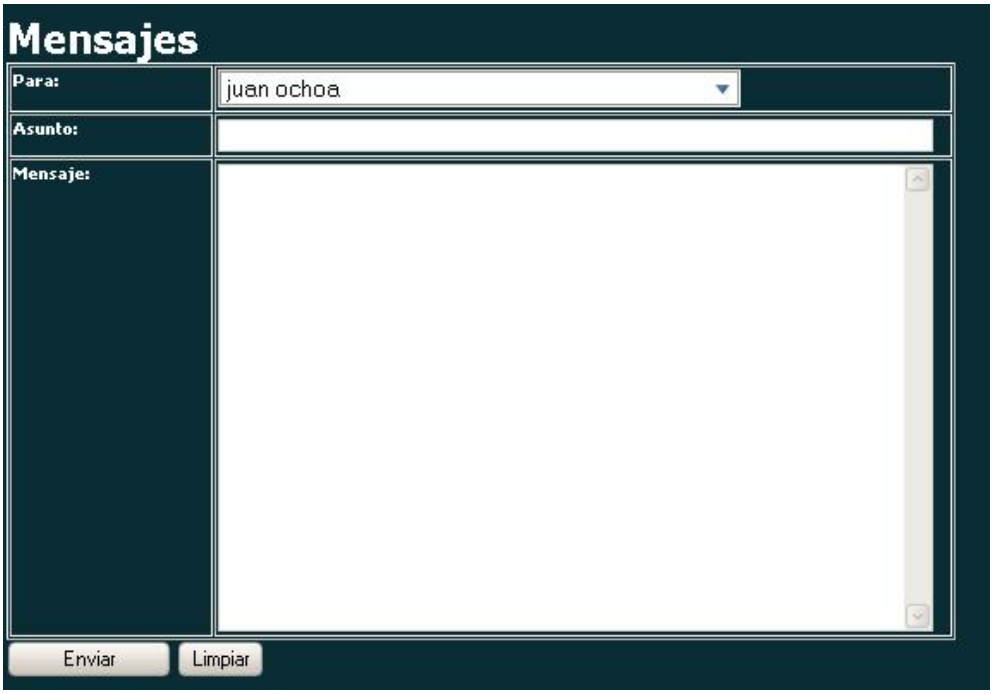


Figura 6.86. Envío de Mensajes (Vendedor o Administrador).

Para proceder al envío de los mensajes, se deberá escoger el cliente al cual se desea enviar el mensaje, llenar los campos de Asunto y Mensaje, una vez que se hayan realizado estas acciones, debe hacer clic en el botón Enviar, aparecerá la siguiente pantalla:

Figura 6.87. Mensaje Enviado.

En caso de ser Cliente, aparecerá la siguiente pantalla:

Figura 6.88. Envío de Mensajes (Cliente).

Para proceder al envío de los mensajes, se deberá llenar los campos de Asunto y Mensaje, ya que el Sistema mediante un proceso interno se encarga de enviar el Mensaje al Agente Vendedor asignado al Cliente, una vez que se hayan realizado estas acciones, debe hacer clic en el botón Enviar, aparecerá la siguiente pantalla:

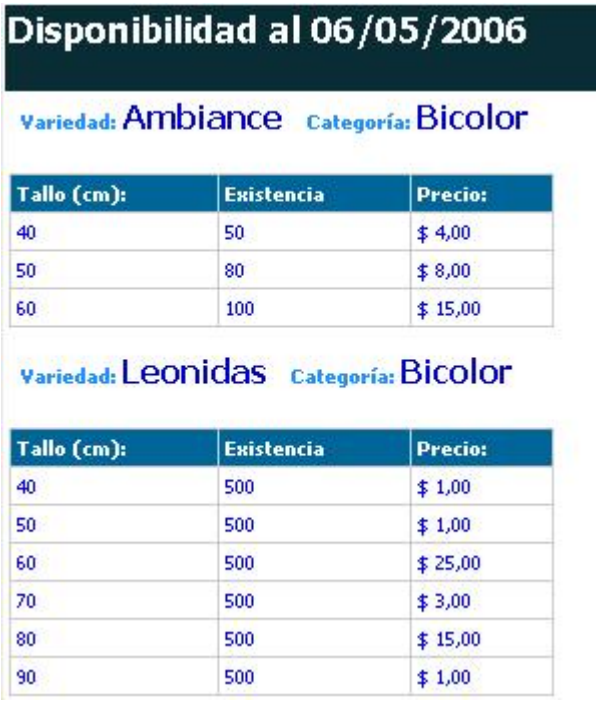
Figura 6.89. Mensaje Enviado.

6.2.16. Informes.

En esta sección se podrá tener accesos a los informes que el Sistema genera.

Figura 6.90. Listado de Informes.

Al hacer clic en el vínculo Disponibilidad se presentará la siguiente pantalla:



Disponibilidad al 06/05/2006

Variedad: Ambiance Categoría: Bicolor

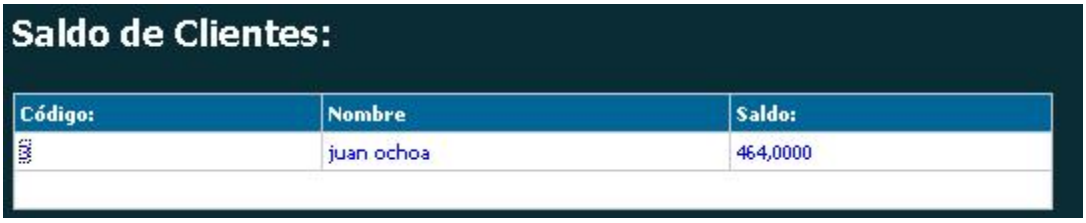
Tallo (cm):	Existencia	Precio:
40	50	$ 4,00
50	80	$ 8,00
60	100	$ 15,00

Variedad: Leonidas Categoría: Bicolor

Tallo (cm):	Existencia	Precio:
40	500	$ 1,00
50	500	$ 1,00
60	500	$ 25,00
70	500	$ 3,00
80	500	$ 15,00
90	500	$ 1,00

Figura 6.91. Informe de Disponibilidad.

Al hacer clic en el Vínculo Saldo de Clientes aparecerá la siguiente pantalla:



Saldo de Clientes:

Código:	Nombre	Saldo:
[illegible]	juan ochoa	464,0000

Figura 6.92. Informe de Saldo de Clientes.

Al hacer clic en el vínculo de Código, se podrá acceder a la información referente al Saldo del Cliente, al realizar está acción aparecerá la siguiente pantalla:

Historial de Compras

Código	Fecha	Valor	Fecha Vuelo	Ver detalles
23	30/04/2006 1:05:24	$ 8,00	30/04/2006 1:05:24	Ver detalles
22	30/04/2006 0:38:21	$ 8,00	30/04/2006 0:38:21	Ver detalles
19	29/04/2006 17:28:27	$ 4,00	29/04/2006 17:28:27	Ver detalles
18	25/04/2006 19:49:59	$ 460,00	25/04/2006 19:49:59	Ver detalles

Figura 6.93. Historial de compras.

Al hacer clic en el vínculo Ver Detalles, se tendrá acceso al detalle de cada una de las compras efectuadas por los clientes, al realizar esta acción aparecerá la siguiente pantalla:

Detalle de su orden

Código:	15	Fecha:	12/04/2006 9:33:08
Salida de Finca:	12/04/2006 9:33:08	Vuelo:	12/04/2006 9:33:08
Aprobado:	☐	Observaciones:	

Producto	Descripcion	Cantidad	Precio	Total	Agencia Carga	Marcacion	Comentario
3	VA40	30	$ 4,00	$ 120,00	G & G Cargo		Pedido completo
7	VA60	15	$ 2,00	$ 30,00	G & G Cargo		Pedido completo
11	VA70	30	$ 3,00	$ 90,00	G & G Cargo		Pedido completo

Total $: 240

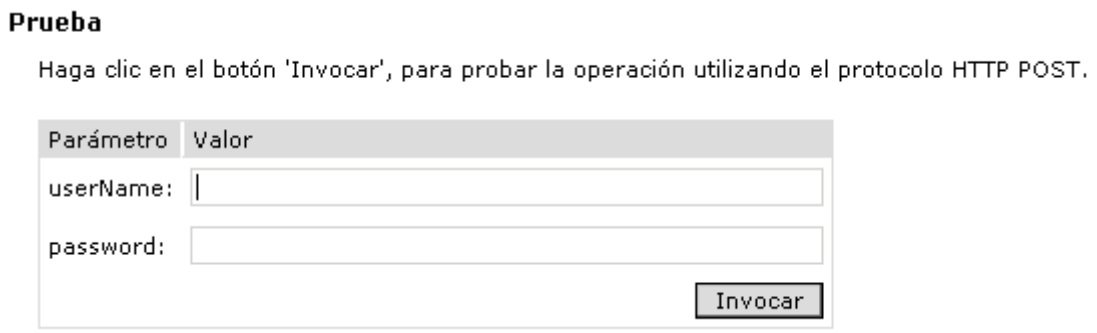
Figura 6.94. Detalle de Compras de Clientes.

6.2.17. Servicios Web.

Ecommerce provee facilidades de programación remota mediante servicios Web, para lo cual expone algunos métodos a través de http, permitiendo a cualquier cliente, construir su propia utilidad para la plataforma que desee y conectarse al sitio Web de Ecommerce y acceder a los diferentes servicios.

Los servicios Web que expone Ecommerce están relacionados estrictamente con los pedidos de los clientes, por lo que se les ha enmarcado dentro del espacio de nombres EcommerceWS y consta de los siguientes elementos:

GetOrdersList.- Este método permite a un cliente remoto acceder programáticamente a su historial de compras. Requiere el nombre de usuario y contraseña del cliente. La siguiente imagen ilustra la interfaz de prueba del servicio:



Prueba

Haga clic en el botón 'Invocar', para probar la operación utilizando el protocolo HTTP POST.

Parámetro	Valor
userName:	
password:	

Invocar

Figura 6.95. Interfaz de Prueba de GetOrdersList.

Login.- Este método permite a un cliente remoto validar programáticamente sus credenciales, para el acceso a Ecommerce. Requiere el nombre de usuario y la contraseña del cliente como parámetros. La siguiente imagen ilustra la interfaz de prueba del servicio:

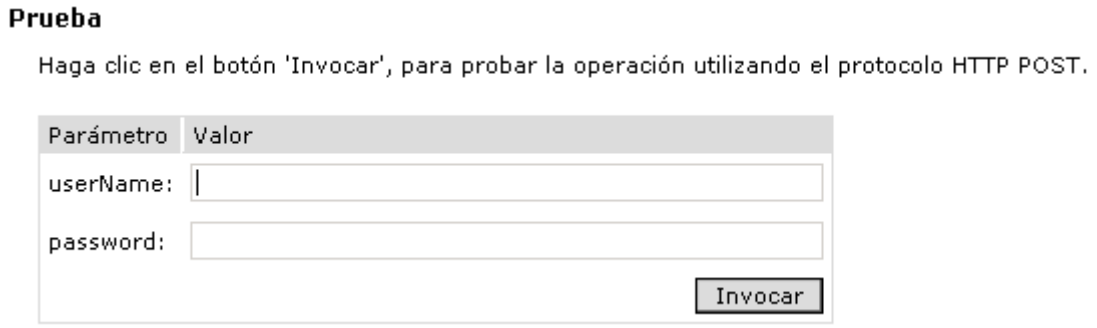


Figura 6.96. Interfaz de Prueba de Login.

GetClientDetails.- Este método permite a un cliente remoto, obtener programáticamente su información almacenada en la base de datos de Ecommerce. Requiere el nombre de usuario y la contraseña del cliente como parámetros, la siguiente imagen ilustra la interfaz de prueba del servicio:

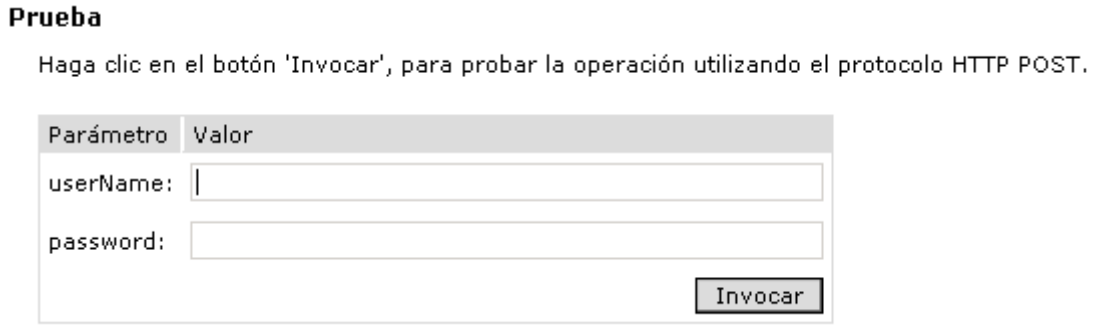


Figura 6.97. Interfaz de Prueba de GetClientDetails.

CheckStatus.- Este método permite a un cliente remoto consultar programáticamente el estado de sus órdenes. Requiere el nombre de usuario, la contraseña del cliente y el código de la orden que se desea consultar. La siguiente imagen muestra la interfaz de prueba del servicio:

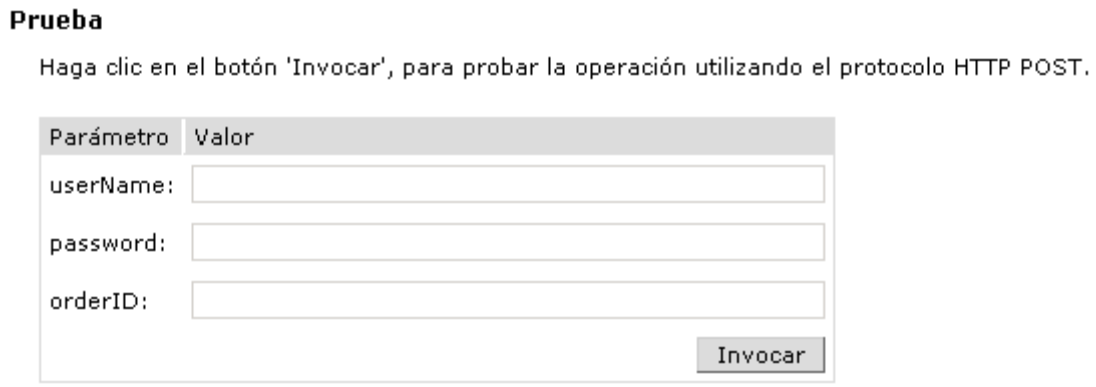


Figura 6.98. Interfaz de Prueba de CheckStatus.

La apariencia de OrdenWS es la siguiente:

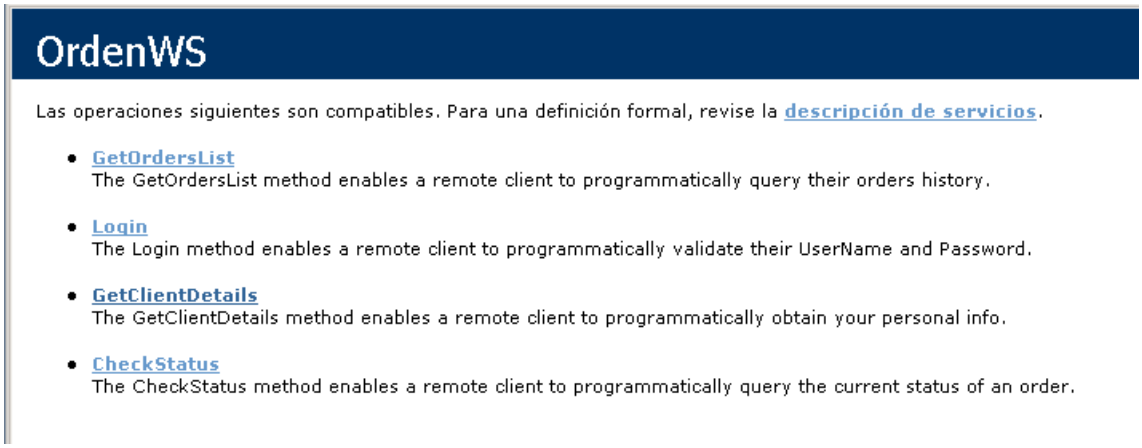


Figura 6.99. Apariencia de OrderWS.

Junto a Ecommerce se provee de una sencilla interfaz Windows denominada "Ecommerce for Windows", la cual permite hacer uso de los servicios Web, y que sirve de ejemplo para que los clientes remotos puedan desarrollar sus propias aplicaciones de conexión. A continuación se explica cómo Ecommerce for Windows aprovecha las facilidades de programación remota de Ecommerce.

Ecommerce for Windows sólo está disponible en idioma inglés y puede obtenerlo gratuitamente desde la sección de descargas del sitio Ecommerce, consta de un archivo ejecutable llamado EcommerceWin.exe, y está comprimido en formato zip. Puede descomprimirlo en cualquier lugar de su disco duro, para que Ecommerce for Windows funcione se requiere tener instalado .Net Framework 1.1.

Al ejecutar Ecommerce for Windows, aparecerá la pantalla que le solicita sus credenciales de usuario:

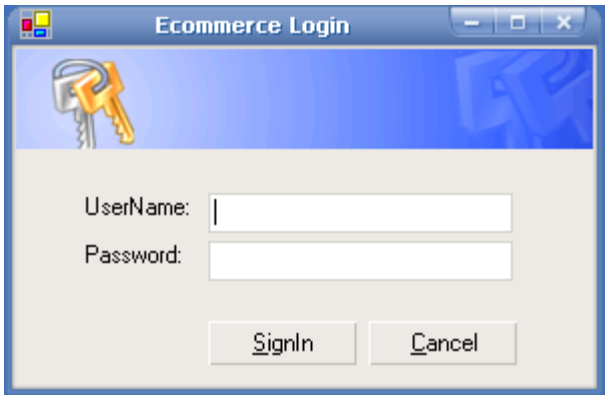


Figura 6.100. Ecommerce Login.

Al validar las credenciales del usuario (botón SignIn), se hace uso del servicio Web Login. Si la validación fue incorrecta, aparecerá el siguiente mensaje de error:

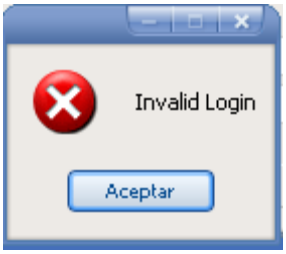


Figura 6.101. Login Inválido.

Si la validación se realizó correctamente, se mostrará la siguiente pantalla:

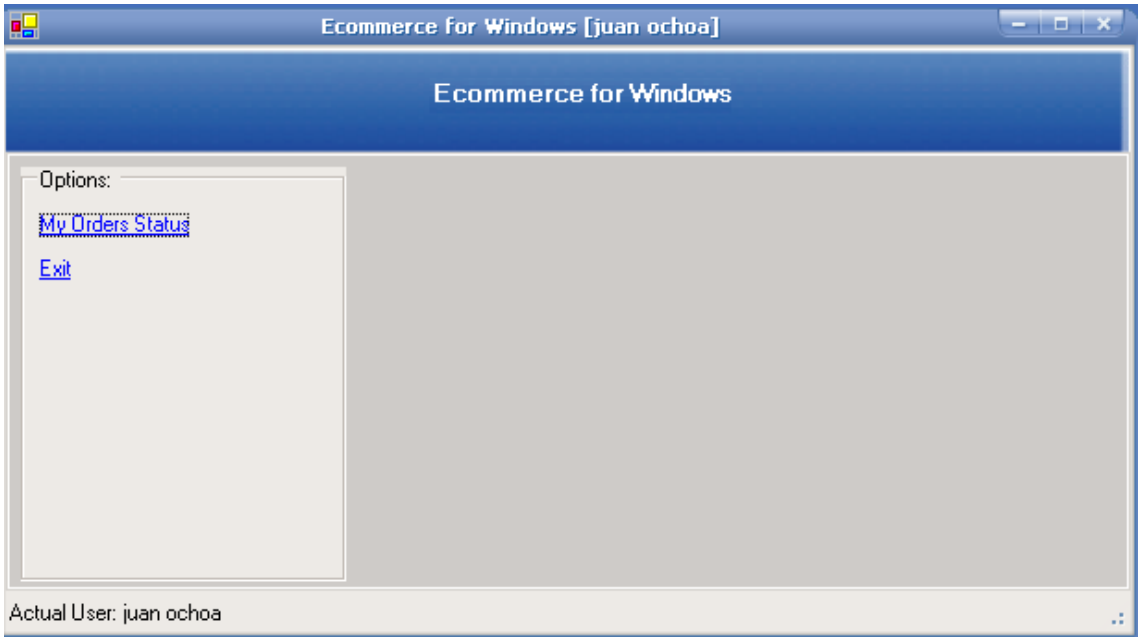


Figura 6.102. Ecommerce for Windows.

Durante la carga de esta pantalla se hace uso del servicio GetClientDetails, para obtener la información relacionada al cliente y mostrarla en la barra de estado y en la barra de título, tal como se aprecia en la imagen anterior.

En el cuadro Options, haga clic sobre “My Order Status”, se desplegará la pantalla que se muestra a continuación:

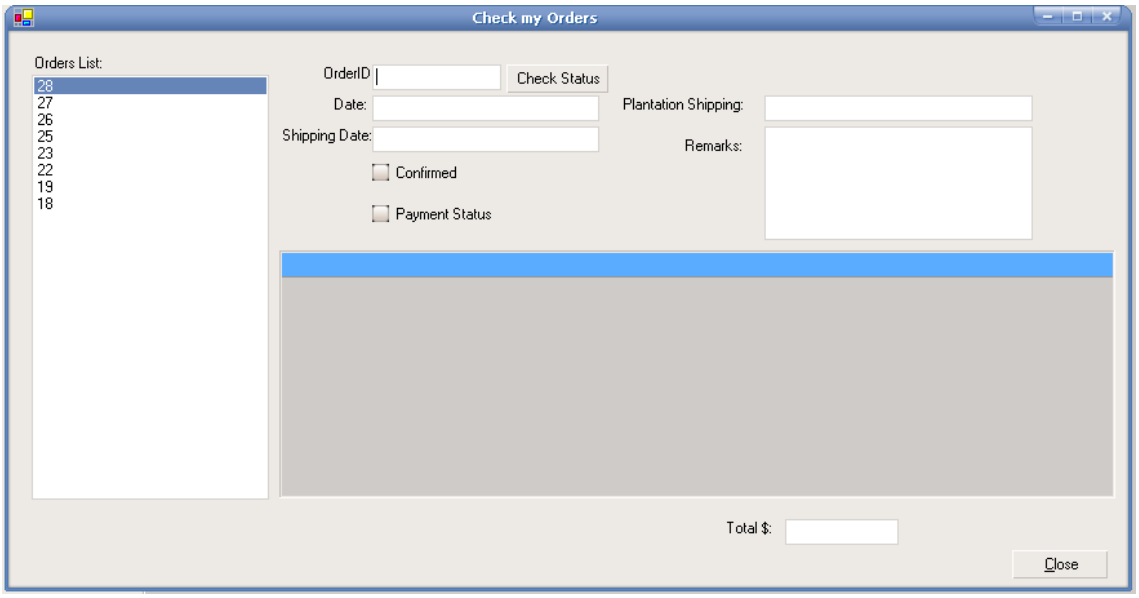


Figura 6.103. Chequear Órdenes.

Para el despliegue de esta pantalla, se hace uso del servicio GetOrdersList, el mismo que sirve para cargar el listado de órdenes de compras del cliente. Finalmente se utiliza el servicio CheckStatus, al hacer clic en el botón del mismo nombre, con lo cual se carga toda la información correspondiente al número de orden seleccionado o digitado en el cuadro de texto OrderId.

Capítulo 7.
Conclusiones y Recomendaciones

El rápido cambio de las tecnologías de información, la globalización e integración de aplicaciones y servicios mediante el Internet, y el acelerado crecimiento de las industrias, exigen al profesional de sistemas utilizar herramientas y metodologías que incrementen su productividad, acortando el ciclo de desarrollo del software y permitiéndole flexibilidad para reducir el impacto de los cambios que se deban efectuar durante el ciclo de vida del software.

El lenguaje de modelado unificado UML, el análisis, diseño y programación orientada a objetos, en combinación con una herramienta adecuada para la construcción del software, constituyen la metodología adecuada para alcanzar estos cometidos, permitiéndonos ventajas como:

- Reutilización, ya que las clases se diseñan para que se reutilicen en muchos sistemas
- Estabilidad.- Las clases diseñadas para una reutilización repetida se vuelven estables.
- Encapsulamiento.- Oculta los detalles y hace que las clases complejas sean fáciles de utilizar.
- Clases Complejas.- Se construyen clases a partir de otras clases, esto permite construir componentes de software complejos.
- Calidad.- Los diseños suelen tener mayor calidad, puesto que se integran a partir de componentes probados, que han sido verificados y pulidos varias veces.
- Incremento de la productividad.- Se logran mejores tiempos totales de desarrollo.
- Escalabilidad.- Encaminar el desarrollo del escalamiento en sistemas complejos de misión crítica, permitiendo un fácil crecimiento de los mismos.
- Control.- Mejor soporte a la planeación y al control de proyectos.

- Minimización de costos.- Efecto de lograr los puntos anteriores.
- Trabajo en Equipo.- Permite la fácil integración de grupos de trabajo, para agilizar el desarrollo.

Así mismo, la elección de una arquitectura de software adecuada, que permita dividir los aspectos esenciales de los requerimientos del cliente en capas distintas, aisladas e integrables convirtiéndolos en servicios, permite construir una solución sólida, segura, y multiplataforma, acentuando esta última, si consideramos el gran auge de los sistemas operativos como Linux, que está logrando una gran concentración de nuevos usuarios.

Todos estos objetivos, han sido logrados durante el desarrollo de esta tesis, utilizando como herramienta de desarrollo la plataforma Microsoft .NET, la misma que nos permitió la mezcla de varias tecnologías y Lenguajes, como ASP.NET, Visual Basic .NET, JavaScript, VBScript, etc., dentro de la misma solución, por lo que consideramos una metodología muy recomendable para los profesionales de sistemas que desarrollan proyectos de software.

Anexo 1. Glosario

.NET Framework.- Conjunto de servicios de programación diseñados para simplificar el desarrollo de aplicaciones en el entorno altamente distribuido de Internet

B2C Business - to Consumer.- Comercio electrónico que surge entre compañías y consumidores

OMC Organización Mundial de Comercio.- Organización internacional que se ocupa de las normas que rigen el comercio entre los países

POO Programación Orientada a Objetos.- una forma de programar basada en la reutilización de código mediante herencia, encapsulación y polimorfismo.

SOAP Simple Object Acess Protocol. - Protocolo que permite mover los datos entre aplicaciones y sistemas

UDDI Lenguaje que permite publicar.- encontrar y usar los Servicios Web basados en XML

UML Unified Modelling Language.- Notación estándar utilizada en el análisis y diseño orientado a objetos, basado en el trabajo de Grady Booch, James Rumbaugh, e Ivar Jacobson.

WSDLWeb Services Description Language.- un formato XML que se utiliza para describir servicios Web

XML Extended Markup Languaje.- Formato universal para representar los datos

Bibliografía

UML Y PATRONES INTRODUCCION AL ANALISIS Y DISEÑO ORIENTADO A OBJETOS.

LARMAN, Craig.

Prentice Hall, México, 1999.

LENGUAJE UNIFICADO DE MODELADO.

BOOCH, Grady; **RUMBAUGH**, James; **JACOBSON**, Ivar

Addison Wesley Iberoamericana, Madrid, 1999.

ASP.NET WEB DEVELOPERS GUIDE

AHMED, Mesbah; **GARRETT**, Chris; **FAIRCLOTH**, Jeremy; **PAYNE**, Chris; **LEE**, Wei Meng; **ORTIZ**, Jonothon

Syngress Publishing Inc, USA, 2002.

ASP.NET DATABASE PROGRAMMING WEEKEND CRASH COURSE.

BUTLER, Jason

CAUDILL, Tony

Hungry Minds Inc, USA, 2002.

INGENIERIA DE SOFTWARE UN ENFOQUE PRACTICO QUINTA EDICION

PRESSMAN, Roger S.

Editorial Mc-Graw Hill, 2002.

www.microsoft.com

www.vbdotnetheaven.com

www.elguille.info

www.asp.net

www.startdotnet.com

www.msdn.microsoft.com

www.hotscripts.com

www.webspots.co.uk

www.macromedia.com

www.sqlmax.com

www.google.com

www.altavista.com