UNIVERSIDAD POLITÉCNICA SALESIANA

SEDE QUITO-CAMPUS SUR

CARRERA DE INGENIERÍA EN SISTEMAS

MENCIÓN TELEMÁTICA – ROBÓTICA E INTELIGENCIA ARTIFICIAL

ANÁLISIS, DISEÑO E IMPLEMENTACIÓN DE UN CRM (CUSTOMER RELATIONSHIP MANAGEMENT) Y UN MÓDULO WAP DESTINADO A VENTAS, PARA MEJORAR LA GESTIÓN DE ATENCIÓN AL CLIENTE DE LA DESPULPADORA DE FRUTAS "LA PULPITA"

TESIS PREVIA A LA OBTENCIÓN DEL TITULO DE INGENIERO DE SISTEMAS

BORIS MARCO MORENO GUALLICHICO
SANDINO ANIBAL DÁVILA MOLINA

DIRECTORA ING. DORIS MEZA

Quito, Noviembre2011

DECLARACIÓN

Nosotros Boris Marco Moreno Guallichico y Sandino Anibal Dávila Molina, declaramos bajo juramento que el trabajo aquí descrito es de nuestra autoría; que no ha sido previamente presentado para ningún grado o calificación profesional y, que hemos consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedemos nuestros derechos de propiedad intelectual correspondientes a este trabajo, a la Universidad Politécnica Salesiana, según lo establecido por la Ley de Propiedad Intelectual, por su reglamento y por la normativa institucional vigente.

_____ _____

Boris Marco Moreno Guallichico Sandino Anibla Dávila Molina

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por Boris Marco Moreno Guallichico y Sandino Anibal Dávila Molina, bajo mi dirección.

_____

Ing. Doris Meza

Directora de tesis

AGRADECIMIENTOS

Al culminar la elaboración de esta Tesis nos queda la deuda de gratitud para muchas personas.

En primer lugar a nuestras familias, que en todo momento nos supieron acompañar en cada proyecto de nuestras vidas. Gracias por estar siempre a nuestro lado.

Gracias a aquellos amigos que estuvieron presentes a lo largo de nuestra formación, visible u ocultamente, aportando en diferente forma: material, espiritual o moral que muchas veces parecerían simples palabras de aliento.

Gracias a nuestra tutora, que supo guiarnos en la elaboración de este trabajo, de la manera más profesional y amigable posible.

Gracias a La Pulpita por aportar con todo lo que necesitábamos en el desarrollo de la Tesis, a su Gerente General por abrirnos las puertas de la empresa, por facilitarnos la documentación, el tiempo del personal y todos los otros detalles que sería largo enumerar.

Y por encima de todo: GRACIAS A DIOS por ser parte de nuestras vidas.

DEDICATORIA

Los resultados de este proyecto, están dedicados a mis padres y a mi hijo quienes han sido pilares fundamentales en mi vida. Mis padres por su lucha incansable se han convertido en mi ejemplo a seguir y destacar, no solo para mí sino también para mi hermana y familia en general. Mi hijo, fruto de un amor inmenso, por ser la fuerza y combustible en mi vida, quiero decirte que cada hora que no paso a tu lado, hay ansiedad en mi corazón por estar junto a ti y gozar cada una de tus aventuras e inquietudes, TE AMO MUCHÍSIMO MATEITO. A ellos les dedico el presente trabajo, ya que sin ellos no lo hubiese podido lograr.

BORIS MARCO MORENO GUALLICHICO

DEDICATORIA

El esfuerzo realizado y el logro conseguido en este largo proceso de trabajo y dedicación se lo ofrezco en primer lugar, a mi Madre y mi Hermana, que son mi familia y la base de mi vida. En segundo lugar un gracias, aquellos amigos que supieron apoyarme cuando en verdad los necesitaba. Esta satisfacción y triunfo los comparto con ustedes, que al ser un motivo de inspiración y lucha, me permitieron finalizar esta etapa de mi vida.

SANDINO ANIBAL DÁVILA MOLINA

RESUMEN

La presente tesis de grado constituye el producto final de un trabajo de investigación aplicado para implementar un sistema CRM en la empresa La Pulpita.

Dicha empresa maneja áreas como Ventas, Facturación, Compras, Producción, Almacenamiento, entre otras, las cuales administradas en conjunto logran dar atención de calidad a los clientes pero se vio la necesidad de implementarun CRM con el cual se espera que el desempeño del personal mejore y por ende se vea reflejado en satisfacción del cliente.

Este trabajo se realizó tomando en cuenta las dos etapas importantes en el desarrollo de un sistema. Primero se realizó un levantamiento de información el cual permitió analizar detenidamente los requerimientos de La Pulpita para poder así presentar un modelo. Y segundo se investigó acerca de las metodologías de software actuales para escoger la más adecuada que lleve a obtener un sistema calificado para este negocio.

El desarrollo de este proyecto, proveerá a la empresa de una herramienta tecnológica, para la atención a los clientes, quienes se beneficiarán al recibir información de manera ágil y oportuna. Además el portal Wap del módulo de ventas, que será de uso exclusivo de los vendedores de La Pulpita y apoyará en el proceso de toma de pedidos. Actualmente la empresa ha dotado a su personal de ventas de celulares con acceso a internet. La herramienta Wap da utilidad a las características que ofrecen los celulares y con esto se podrá realizar las ventas en tiempo real ya que la información alimentará directamente a la base del sistema; esto ayudará a controlar el stock y dará valor agregado a la imagen de la empresa.

ABSTRACT

The present thesis constitutes the final product of an investigation work applied for implementa CRM system in the La Pulpitacompany.

The above mentioned company handles areas such as Sales, Invoicing, Purchasing, Production, Storage, among others, which run together to achieve quality care to clients, but managers saw the need to implement a CRM with which it hopes the performance of the personnel improves and it will be reflected in satisfaction of the client.

This work was performed taking into account the two important stages in the development of a system. First there was realized an information survey which allowed to analyze La Pulpita´s requirementsto present a model. And second was investigated about the current software methodologies to choose the most appropriate to take to get a qualified business system.

The development of this project will provide to the company of a technological tool, for the attention to the clients, who will be benefit by having informationin an agile and opportune way. In addition the portal Wap of the module of sales, which will be of exclusive use of LaPulpita sellers, will rest on the process of capture of orders. Nowadays the company has provided to this personnel of cellular with access to Internet. The tool Wap gives usefulness to the characteristics that the cellular ones offer and with this it will be possible to realize sales in real time since the information will feed directly to the base of the system; this will help to control the stock and will give value added a the image of the company.

INDICE

CAPÍTULO I

1.1. INTRODUCCIÓN

El mercado de pulpa de fruta congelada en el país, se ha visto involucrado en un crecimiento continuo, lo cual se refleja en el nivel de competencia que existe entre las empresas. La calidad del producto hace la diferencia entre una despulpadora de fruta con alto nivel de producción y otra con poca producción, sumado a un conjunto de estrategias utilizadas para mantener y obtener nuevos clientes.

Un Sistema CRM,ordena las relaciones con el cliente (datos, ofertas, contactos personales, agenda, precios, productos, etc.).Por tal motivo llega a convertirse en una herramienta principal de la empresa junto con la gestión de los recursos humanos.

El cliente siempre debe ser tomado como parte importante del negocio, por lo que los representantes de ventas o vendedores internos deben estar capacitados para ofrecer una atención excelente, ya que ellos son la imagen del negocio y una mala actitud generada ante el cliente puede ser costosa para la empresa,pues probablementelos clientes mal atendidos se encargarán de hacerle mala publicidad.

No se trata solamente de conocer el teléfono, la dirección donde vive, sino también qué compra, dónde lo compra, por qué compra un determinado producto, con qué frecuencia lo hace, si es a crédito o de contado, es decir, conocer al cliente de tal manera que la empresa se adelante a determinar cuáles son sus necesidades,le permita definir las estrategiasmás efectivas paraalcanzara ofrecer lo que el cliente realmente necesita, que se sienta satisfecho del negocio realizado y que se fidelice con el producto. CRM es la herramienta que apoyará y permitirá lograr conseguir un cliente satisfecho.

1.1.1. PLANTEAMIENTO DEL PROBLEMA

La Pulpita, es una empresa dedicada a la comercialización de pulpa de fruta, tanto en el mercado nacional como internacional y que mantiene una relación directa con el cliente.

En el caso de La Pulpita, se presentan problemas tales como:

- Demora en la consulta del producto destinado a la venta, solicitado por el cliente.

- Dificultad en el cálculo de conversión de la materia prima a producto terminado, en unidades de medidas de peso, para tener conocimiento de la cantidad de producto que se tiene disponible para la venta después de la producción.

- Dificultad en archivar y registrar los pedidos de productos, las ventas y compras realizadas.

- Problemas al tomar decisiones de mercadotecnia de manera más rápida y efectiva tales como determinación de precios y estrategia de canales.

- La facturación se la realiza de forma manual, lo cual no da una buena presentación ante los clientes, por lo que es necesario que el proceso de facturación se lo realice en forma digital.

Los problemas descritos afectan la gestión administrativa, comercial y productiva de la compañía, ya que, al ser procesos que se los realiza manualmente, generan pérdida de tiempo, incremento en los costos de producción, y por ende se ve impactada la posición de la empresa en el mercado.

1.1.2. OBJETIVOS

1.1.2.1. Objetivo General

Analizar la gestión de atención al cliente por parte de La Pulpita y diseñar e implementar un sistema CRM, que permita agilitar procesos como ventas, compras, almacenamiento, procurando un mejor servicio para el cliente.

1.1.2.2. ObjetivosEspecíficos

a) Ofrecer al usuario de la empresa un sistema de fácil manejo para acceder a la información necesaria, con el fin de llegar a satisfacer los requerimientos del cliente y de esta manera mantenerlo fiel a la empresa.

b) Diseñar el sistema con sus módulos para la optimización de los procesos de: ventas, compras, almacenamiento, y producción, utilizando metodología OMT.

c) Desarrollar una aplicación Wap, utilizando la metodología OOHDM. Que integrada en el sistema base, al módulo de ventas, permita un mejor control de los pedidos y manejo de stock.

d) Manejar información centralizada a través del sistema, la misma que va a ayudar a los ejecutivos de la empresa a tomar decisiones más efectivas sobre eventos específicos.

e) Controlar correctamente las transacciones procesadas y almacenadas que se vean involucradas en los procesos de la empresa mediante el uso de una efectiva base de datos, de esta manera el soporte para la toma de decisiones será veraz y oportuno.

1.2. JUSTIFICACIÓN DEL PROYECTO

La Pulpita maneja áreas como Ventas, Facturación, Compras, Producción, Almacenamiento, entre otras, las cuales administradas en conjunto logran dar atención de calidad a los clientes. Con la implementación de un CRM, se espera que el desempeño del personal mejore y se vea reflejado en la satisfacción del cliente, dándole un valor agregado a la empresa.

El ambiente de mercado en el cual se maneja La Pulpita, demanda que empresas como ésta, requieran sistemas informáticos que ayuden al servicio que se presta al cliente como por ejemplo: mayor agilidad en consulta de datos, información de los productos disponible para la venta, e incluso datos de la empresa.

El desarrollo de este proyecto, proveerá a la empresa de una herramienta tecnológica, para la atención a los clientes, quienes se beneficiarán al recibir información de manera ágil y oportuna, además el portal Wap del módulo de ventas, que será de uso exclusivo de los vendedores de La Pulpita apoyará en el proceso de toma de pedidos, actualmente la empresa ha dotado a su personal de ventas celulares con acceso a internet que están siendo subutilizados. La herramienta que se desarrollará, dará utilidad a las características que ofrecen los celulares y con esto se podrá realizar las ventas en tiempo real ya que la información alimentará directamente a la base del sistema; esto ayudará a controlar el stock y dará valor agregado en la imagen de la empresa.

1.3. DESCRIPCIÓN DEL MERCADO DE PULPA

La producción de pulpa de fruta congelada ha tenido un crecimiento significativo en el país. Las empresas en sus inicios, procesaban 20 toneladas de fruta congelada

alaño. Ahora son un promedio de 200 toneladas anuales que son procesadas; en un lapso de siete años la producción creció 10 vecesen su capacidad.

Según estudios realizados se ha llegado a calcular que el mercado nacional de pulpa de fruta, mueve cerca de USD 20 millones al año. El producto, está dirigido a un segmento socioeconómico medio y alto, “por lo que la posibilidad de crecer en otros sectores es amplia”.

Con esto se determinó que el 70% del consumo se concentra en Quito. Esto ocurre por la cantidad de autoservicios (solo Supermaxi tiene 15 locales), y también porque el producto es demandado por firmas de catering (comida a domicilio) que trabajan para el sector público.

Además la pulpa congelada tiene doble función, explicanlos estudios. En la elaboración de jugos funciona como producto final, pero también sirve como materia prima para industrias que elaboran mermeladas, conservas y helados.

Existe una ventaja muy importante del producto, su facilidad de preparación. “El tiempo para el ama de casa es más escaso y el producto ayudaa su trabajo”.[1]
Otra ventaja, es que las productoras aprovechan la abundancia de una fruta en determinada temporada para procesarla y congelarla, aunque se advierte que hay que tener una infraestructura adecuada (cuartos fríos y maquinaria).

Un inconveniente que se llegó a determinar, es el espacio que la pulpa congelada ocupa, frente a otros productos elaborados también con frutas. Se comprueba que frente al envasado aséptico (que no requiere refrigeración, ni congelación), la pulpa congelada ocupa más espacio. En el mismo espacio en que caben 100 toneladas de pulpa congelada, con envases asépticos se pueden almacenar 1.000.

[1]Revista Líderes
Mercados | Líderes | Lunes, 3 de Agosto del 2009
Pulpa de fruta: este producto hizo del frío su principal socio
http://www.revistalideres.ec/Generales/solo_texto.aspx?gn3articleID=9808

La pulpa congelada producidanacionalmente se ha logrado exportar a distintos países como por ejemplo: Estados Unidos, España y la Región Andina, y a mas de eso existen mercados abiertos, según los consultados, a los cuales se puede apuntar para la venta de este producto. También hay planes para llegar a Asia y Centroamérica.

1.4. CRM (CUSTOMER RELATIONSHIP MANAGEMENT)

1.4.1. INTRODUCCIÓN

El mundo actual es sumamente complejo y exigente; por esto, enmarcadas en los márgenes de exigencia, calidad y atención, que impone la denominada "nueva economía", las empresas competitivas de hoy en día han comprendido y aprendido que el éxito hay que buscarlo en el manejo de una exitosa Relación con los Clientes, más allá de la rentabilidad por producto o líneas de producción del pasado. El éxito, y por ende la rentabilidad empresarial, vienen entonces de la mano con la interacción con el cliente, del conocimiento del cliente, de reconocer su valor actual y potencial, de saber qué productos le gustan, de escuchar sus quejas y sugerencias y saber cómo utilizarlas en beneficio de la compañía, en definitiva se trata de poder conocer y predecir el comportamiento actual y futuro de los clientes para garantizar su completa y plena satisfacción, para finalmente conseguir su lealtad hacia la empresa[2].

El surgimiento del CRM ha llevado a las empresas hacia un enfoque basado en los procesos importantes que se relacionan con el cliente que busca extraer de todas las fuentes empresariales: las metodologías, las estrategias, el software, y las capacidades que ayudan a organizar y gestionar las relaciones con los clientes y potenciales clientes.

[2]http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=crm1#_ftn1

Los objetivos que busca el CRM son:

- Definir los objetivos y estrategias de entendimiento, venta y seguimiento del Cliente, evaluando los recursos disponibles, tanto en hora hombre como en recursos económicos.

- Optimizar el proceso de venta (venta de producto/sistema/servicios o venta de proyectos/soluciones) para reducir gastos/recursos e incrementar la satisfacción del cliente.

- Mejorar la eficiencia del proceso de ventas (venta de productos/sistemas/servicios o venta de proyectos/soluciones) a través de la satisfacción de necesidades puntuales del cliente, con mira a futuros negocios.

- Mejorar el contacto con el Cliente de acuerdo al orden del proceso para incrementar la satisfacción del cliente.

1.4.2. DEFINICIÓN

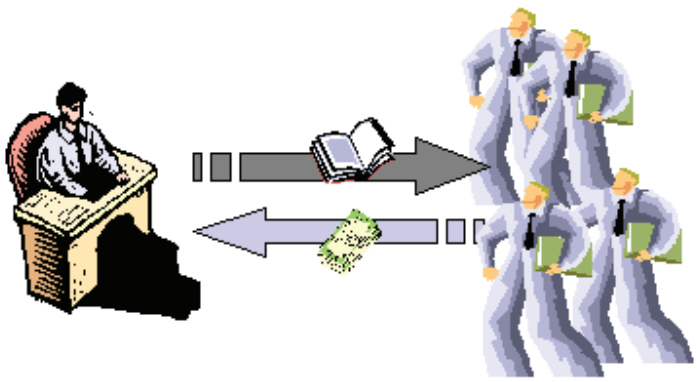


Fig. 1.1CRM – Representación Atención al cliente.
Fuente:http://winred.com/images/eicrm2.gif

El CRM es una importante estrategia de negocio, tanto para empresas de e-business como para compañías tradicionales, a la hora de hacer frente a un mercado cada vez más competitivo y exigente. La evolución de Internet ofrece infinitas oportunidades a las empresas, pero también ha abierto un nuevo mundo a los clientes. “Es una estrategia de negocio que sitúa al cliente como el corazón de su compañía.

Se centra en proporcionar una serie de procesos orientados al cliente a través de los distintos departamentos y canales, de manera que permitan atraerlos y retener a los clientes.

El principal objetivo es incrementar la rentabilidad del cliente. Esta es la razón por la que las compañías están fomentando el conocimiento sobre sus clientes, porque les permite identificar oportunidades de ventas cruzadas y ventas complementarias.

Se define como CRM, a un programa informático que una empresa utiliza como apoyo a la gestión de la relación con los clientes. De esta forma es posible lo siguiente:

- Almacenar toda la información relativa al cliente, de manera organizada para facilitar su acceso y mantener su integridad.

- Realizar estadísticas y seguimiento de objetivos de ventas.

- Consultar cualquier dato o persona de contacto en el cliente.

CRM es una estrategia para identificar, atraer y mantener a los clientes, con procesos eficaces que ayuden a satisfacer las necesidades actuales y conocer las necesidades potenciales de los mismos.

En la filosofía del CRM, el centro de atención es el cliente, y por ello las áreas de la empresa que se dedican a la comercialización, marketing y atención al cliente, son las encargadas de fidelizar y fortalecer las relaciones con los clientes.

Las principales características del CRM son las siguientes:

- Maximiza la información del cliente.
- Identifica nuevas oportunidades de negocio.
- Mejora el servicio al cliente.
- Procesos optimizados y personalizados.
- Mejora de ofertas y reducción de costos.
- Identifica los clientes que generan mayor beneficio a la empresa.
- Fideliza al cliente.

Las empresas que cuentan con un CRM, presentan una ventaja competitiva, respecto a sus competidores, anticipándose a las necesidades de los clientes, a la vez que aumentan el nivel del servicio y generan muchas más oportunidades de contacto entre cliente / proveedor.

El CRM es un desafío de negocio y no tecnológico. La tecnología ayuda a gestionar las relaciones con los clientes de una manera operativa pero sin

elcorrecto enfoque tanto estratégico como de personas y procesos, el proyecto nunca alcanzará el éxito.[3]

Fig. 1.2 Relación con el Cliente

Fuente:http://www.growit.com.ar/images/stories/GrowIT/imagen%20t%20para%20joomla.jpg

1.4.3. VENTAJAS Y DESVENTAJAS DE UN CRM.

La principal ventaja de la implementación de un CRM, es el aumento de información que posee de sus actuales y potenciales clientes, lo que permite direccionar sus ofertas y necesidades, aumentado así el nivel de satisfacción y optimizando su ciclo de vida.

Otra de las ventajas es el aumento de las ventas y la reducción del ciclo de venta.

[3] http://www.monografias.com/trabajos21/customer-relationship/customer-relationship.shtml

Las pequeñas y medianas empresas tienen la ventaja de contar con pocos clientes y menos personal lo que facilita el manejo de la información y por lo tanto los problemas para compartirla son menores.

Un problema que se puede presentar en las empresas de menor tamaño es la falta de formalización de los procedimientos y la falta de interés de los usuarios por adentrarse en la importancia que tienen estos y en la filosofía de trabajo de la empresa.[4]

En resumen, CRM es una solución informática, que permite a la empresa la captura fácil de datos del cliente y el uso generalizado de forma rápida en el diario vivir del personal de la empresa con los clientes.

El concepto de CRM en sí, no está relacionado directamente con tecnología, sino que es una filosofía corporativa en la que se busca conocer y anticipar las necesidades del cliente, es decir, se trata de una estrategia de negocio enfocada en el cliente y sus necesidades.

Una correcta implementación de CRM, debe contar con un proceso de elaboración en el cual esté integrada toda la empresa, implicando cambios en los procesos, estrategias y funciones.[5]

[4] http://www.gerenciaynegocios.com/teorias/crm/03%20-%20ventajas%20y%20desventajas.htm
[5] http://www.gestiopolis.com/canales/gerencial/articulos/20/crm.htm

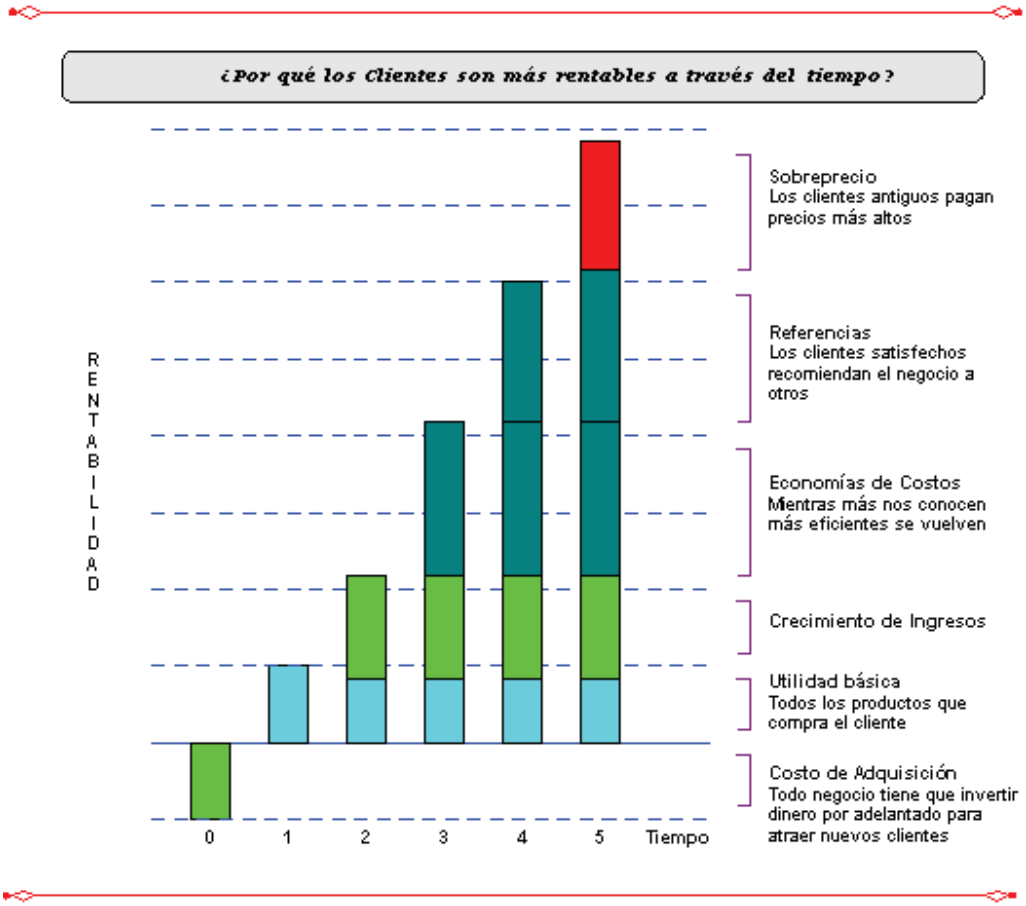


Fig. 1.3 Rentabilidad vs Tiempo (Rentabilidad de los Clientes con el tiempo.)
Fuente: http://www.adictosaltrabajo.com/tutoriales/crm1_archivos/image010.gif

1.4.4. FASES DE UN CRM

El proceso de CRM consta de cuatro etapas principales, que tal como muestra la figura 1.4, obedecen a un orden natural y cuyo objetivo es tratar clientes diferentes de una manera diferente, estas etapas son:

- **Identificar:** Consiste en dar a conocer el sistema, lograr que los usuarios se identifiquen con éste, para conocer a cada cliente y su historia, de forma individual.

- **Diferenciar:** Cuantificar el valor neto presente de las utilidades de una relación futura del cliente individual con la organización. Para encontrar los clientes de mayor valor real y los clientes de mayor potencial o valor estratégico y los clientes sin probabilidad de utilidad.

- **Interactuar:** Durante esta etapa, se busca conocer cada vez más a los clientes y sus necesidades, y así crear una relación de aprendizaje que conlleve a una personalización de los servicios o productos.

- **Personalizar:** Finalmente, esta etapa, basada en el conocimiento de los clientes, pretende crear una relación individualizada según las necesidades del consumidor (personalización masiva).

Fig. 1.4 Personalización de cada Cliente

Fuente: http://es.scribd.com/doc/55058680/7/Figura-2-2-%E2%80%93-Tres-etapas-de-CRM

1.4.5. EN QUÉ EMPRESAS SE DEBE APLICAR EL CRM

La buena relación con los clientes se debe a un factor decisivo al momento de realizar un negocio, dice Bhullar[6]. Telecomunicaciones, seguros, banca, corretaje y otros servicios financieros llevan mucho tiempo comprometidos con el CRM y han podido comprobar un retorno continuo de la inversión. Pero, dice, “hay todavía enormes oportunidades en el negocio business-to-business, especialmente para fabricantes por proceso y por fabricación discreta que

[6]El CRM como estrategia de negocio, Referencia al Centro de negocios y profesionales de Microsoft, publicado en Web: http://www.microsoft.com/spain/empresas/soluciones/guia_crm.mspx

realmente nunca han sido expertos en las relaciones con el cliente." Incluso si la empresa tiene pocos clientes, puede beneficiarse al replantear sus procesos a favor de las buenas relaciones con el cliente. "Deberían fortalecer lo que han hecho ya. Siempre hay espacio para la mejora. Las cosas pueden cambiar". Pero la mayoría de las compañías, si se les pregunta, probablemente dirían que ellos ya tratan al cliente como el número uno. Según Bhullar, la realidad es bastante diferente. "Ellos siempre se han organizado internamente alrededor del producto. Ahora la ventaja competitiva la tendrán las empresas que pongan al cliente en primer lugar. Y esto no significa simplemente comprar una aplicación CRM."

1.5. WAP

1.5.1. INTRODUCCIÓN

El mercado de las telecomunicaciones está experimentando cambios trascendentales en los últimos tiempos. Estos cambios están propiciados por diversos factores, unos provocados por la liberalización, otros por la rápida evolución tecnológica, o bien por las variaciones en el estilo de vida de los profesionales o los particulares.

Quizás el cambio principal lo está representando la irrupción masiva de los datos en los negocios y en la sociedad en general, impulsada en buena medida por el fenómeno Internet. Diversos estudios y análisis contrastan el crecimiento exponencial del tráfico de datos a nivel mundial (crecimiento mensual del 8%), con el crecimiento lineal del tráfico de voz (7% al año). Esto hará que, de seguir esta tendencia, el tráfico de datos supere al de voz.

La explicación a este fenómeno es sencilla: mientras que el tráfico de voz está limitado al número de personas que, por término medio, hablan de 10 a 15

minutos por día, el tráfico de datos está basado en el número de máquinas y diversos dispositivos interconectados, que crece exponencialmente y que podrían estar las 24 horas del día intercomunicadas.

En el siguiente gráfico se puede ver de mejor manera esta evolución:

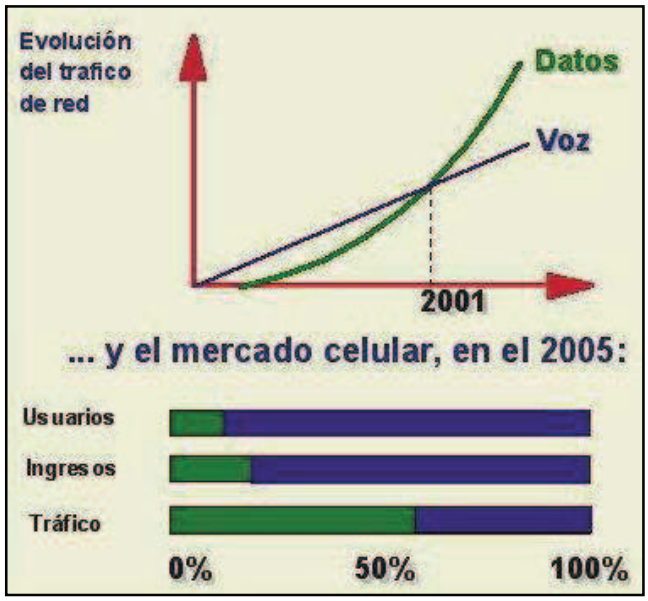


Fig. 1.4Evolución de los mercados de voz y datos en telefonía móvil

Fuente: http://www.coit.es/publicac/publbit/bit117/datos2.html

A este fenómeno no son ajenos los operadores, hasta ahora tradicionalmente denominados móviles, que han demostrado que dominan perfectamente el negocio de la voz y están perfectamente posicionados para ser, en nuestra opinión, los ganadores finales en el negocio global de las telecomunicaciones. Para ello tienen un gran reto por delante: el dominar e introducirse fuertemente en el negocio de datos e Internet, para lo cual la evolución tecnológica, con las nuevas generaciones de comunicaciones móviles de banda ancha (GPRS, UMTS) o de terminales (WAP, Bluetooh), les posiciona muy adecuadamente. El resto es cuestión de tarifas adecuadas, aplicaciones e imaginación.

1.5.2. DEFINICIÓN

La combinación de la telefonía móvil e Internet no satisfacía las expectativas creadas, sobre todo por las limitaciones de las redes de comunicación inalámbricas y la inoperatividad de los terminales. Dichos obstáculos son ahora superados por la tecnología WAP (WirelessApplicationProtocol o Protocolo de Aplicaciones Inalámbricas), la cual viene a superar estas limitaciones para convertirse en un nuevo estándar lleno de posibilidades.

Esta tecnología posibilitará el acceso y la manipulación de la información que se encuentra en un WEB Server sin la necesidad de un ordenador y un módem, pudiéndose realizar ésta a través de una terminal móvil de una forma inalámbrica utilizando la cobertura celular con la tecnología disponible dentro del país.

La tecnología WAP puede funcionar tanto en redes de transporte GSM como en GPRS o UMTS. Está por tanto, optimizado para superar las deficiencias de la telefonía móvil actual. Como consecuencia de ello, los entornos creados expresamente en tecnología *HTML*[7] no son legibles e interpretables por este nuevo protocolo, para ello, se ha creado un nuevo lenguaje de marcas, un lenguaje de script y un interfaz de aplicaciones, *WML*[8], *WMLScript*[9] y *WTAI*[10].

Para conectarse a Internet mediante WAP es necesario disponer de un teléfono móvil que soporte esta tecnología. Hoy en día la mayoría de los terminales que se comercializan en el Ecuador soportan WAP.

Los teléfonos móviles con WAP deben disponer de un Micro-Browser (Micro Navegador compatible con WAP) que es similar a los navegadores que se

[7] HTML, Siglas de HyperText Mark-up Language
[8] WML, Siglas de Wireless Markup Language
[9] WMLScript, Siglas de Wireless Markup Language Script
[10] WTAI, Siglas de Wireless Telephony Application Interface

utilizan en Internet, como IE o Netscape. A través de este Micro Browser el terminal solicita la información al servidor Web, del mismo que se realiza en Internet cuando el usuario teclea una dirección Web.

La estructura necesaria que una Operadora de Telefonía móvil debe crear para ofrecer los servicios vinculados a WAP está formada, principalmente, por tres elementos: Teléfono WAP, Gateway WAP y Servidor Web.

El diseño de WAP fue creado para trabajar bajo restricciones de memoria y procesadores, pequeñas pantallas monocromáticas capaces de desplegar muy pocas líneas de texto y conexiones irregulares debido al ancho de banda reducido. Gracias al apoyo de varios cientos de vendedores de estos dispositivos, el WAP Forum, está convirtiendo a WAP en el estándar, permitiendo que cualquier usuario con un dispositivo de mano pueda tener acceso a la información, lo que ha traído por consiguiente que cada vez más información se transmita por redes inalámbricas.

Actualmente muchos sitios están creando sus versiones en WML para ser transmitidos por medio de WAP. Para conocer esta tecnología sin necesidad de adquirir un dispositivo inalámbrico de alta tecnología, puede utilizarse un emulador como el localizado en el sitio de Gelon.net (Funciona desde el navegador), o el primer navegador WML para Windows: WinWap. Las páginas que implementan WAP sólo pueden verse a través de un dispositivo compatible o de estos emuladores. En realidad, actualmente aparecen más páginas web con tecnología Wap que aparatos. O sea que suele ser difícil conseguir un teléfono WAP.

1.5.3. COMUNICACIÓN POR WAP.

Para realizar esta comunicación se deben seguir estos pasos:

- El usuario teclea en su terminal WAP la dirección que desea visualizar.

- El Micro Browser del móvil crea una solicitud que contiene la dirección y le añade información que identifica al cliente. A continuación envía toda esta información al Gateway WAP.

- El Gateway WAP recoge la solicitud de información, la convierte a una solicitud *HTTP*[11] o *HTTPS*[12] y la envía al Servidor Web.

- El Servidor Web recoge la solicitud y diferencia si la dirección solicitada es estática, en este caso la recupera, o dinámica, lanza el programa necesario para ofrecer los datos.

- El servidor Web coloca el encabezado HTTP o HTTPS dependiendo si la página es normal o segura y la envía al Gateway WAP.

- El Gateway WAP recoge la página, genera una respuesta para el terminal quitando el encabezado y la envía.

- El Micro Browser del terminal interpreta la respuesta enviada por el Gateway WAP y la muestra al usuario en pantalla.

En ciertos casos es conveniente montar un gateway propio, todo depende del objetivo planteado.

Montar un gateway WAP propio requiere mantener las conexiones con la red inalámbrica de telefonía móvil, de alguna forma. Por ejemplo, se debe poder hacer marcación dentro del sistema *SMS*[13], tener módems para acceso CSD

[11] HTTP, Siglas de Hypertext Transfer Protocol
[12] HTTPS, Siglas de Hypertext Transfer Protocol Secure
[13] SMS, Siglas de Short Messenger Server

(eso significa que un móvil debería poder marcar el número de teléfono asignado al modem como si nuestro sistema fuese un dispositivo de la red de telefonía móvil más). Esta opción es la más adecuada si la intención es actuar como un ISP (Servidor y almacenador de páginas) para telefonía móvil.

En este caso, se debe comenzar por contactar con los operadores de telefonía móvil para integrar, de alguna forma, en sus redes.

Por otro lado, si lo que se quiere es simplemente crear algunas páginas o servicios comerciales, se puede conformar con almacenar las páginas estáticas (o las aplicaciones que las generen) en un servidor WEB típico (Apache, IIS, Netscape, etc) y configurarlo para que sirva páginas WML (añadir unos "tipos MIME").

1.5.4. ESTRUCTURA WAP.

Los protocolos WAP son una especificación abierta de protocolos, regida por un consorcio (similar al W3C en HTML) de partes interesadas: el Forum WAP.

Los protocolos WAP son análogos a los existentes en la tecnología utilizada en Internet, solo que optimizados para la "potencia" que ofrecen los dispositivos portátiles, y limitados por el ancho de banda que ofrece la tecnología inalámbrica y cubren tanto la capa de aplicación (WML y WMLScript, conocidos como capa de aplicación), y las capas de transporte (WSP, WTP, WTLS y WDP).

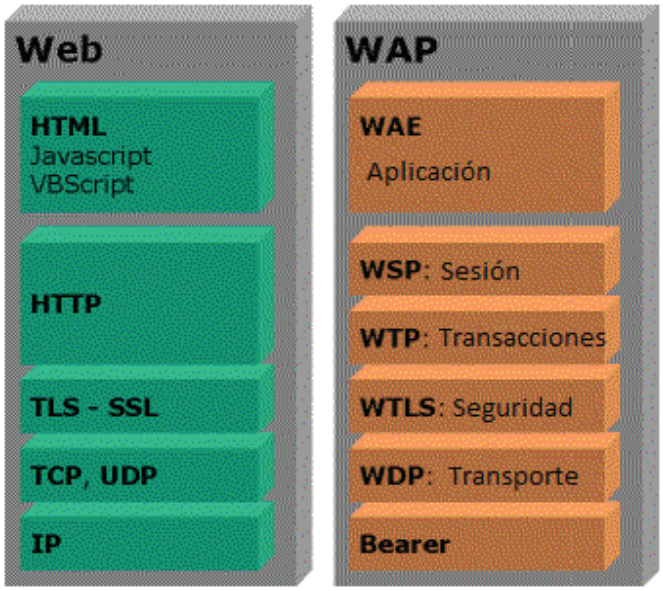


Fig. 1.5Estructura WAP

Fuente: http://geneura.ugr.es/~maribel/wap/introduccion/index.shtml

La última capa (en la estructura) depende del tipo de red para móviles.

1.5.5. SERVIDORES WAP

Internet se compone de miles de servidores que almacenan las páginas a los que acceden los clientes (los navegadores, generalmente).

El sistema seguido por WAP no es muy diferente. Ahora los clientes son los dispositivos móviles y los servidores deben seguir almacenando las páginas y aplicaciones.

La única consideración que hay que hacer es que hay que conectar la red inalámbrica (telefonía móvil) a Internet, de forma que el dispositivo portátil pueda hacer una petición de página WML al servidor.

Para conectar ambas redes las especificaciones WAP asumen que habrá un Gateway WAP que convierte las peticiones WAP a peticiones WEB, y las respuestas WEB a respuestas WAP, como vemos en la siguiente figura:

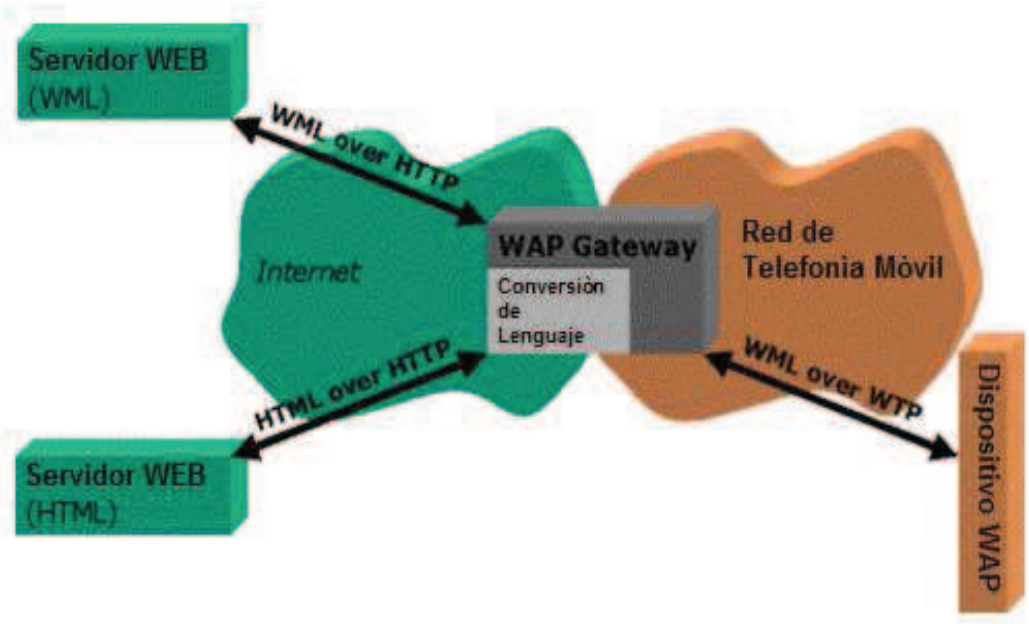


Fig. 1.6Servidor WAP

Fuente: http://geneura.ugr.es/~maribel/wap/introduccion/index.shtml

En teoría, el Gateway debe poder convertir páginas HTML, sobre la marcha, en páginas WML.

Sin embargo, puesto que WML está pensado y diseñado para presentar información en pantallas muy pequeñas, y no es tan potente, visualmente, como HTML, esa transformación de información HTML a WML puede ser complicada de hacer (más aún si hay scripts, animaciones, imágenes grandes, etc.).

1.5.6. UTILIDADES DE LA TECNOLOGÍA WAP

1.5.6.1. Monitor de centros neurálgicos (redes neuronales)

Este tipo de monitor podrá colocarse en cualquier punto de la casa y acceder a él de manera directa gracias a la interfaz estilo Wap. Además, permitirá ver la televisión, dejar notas o apuntar la lista de la compra con una agenda especial, acceder a un servicio de teleasistencia, navegar por Internet, controlar los niveles de consumo de los electrodomésticos e incluso enviar al horno recetas de cocina para que programe automáticamente el tiempo de cocción.

El monitor funcionará con electrodomésticos digitales, pero también con máquinas tradicionales a las que se incorpora un pequeño módem.

1.5.6.2. Mercadotecnia Dinámica

Consiste en utilizar las repetidoras del servicio inalámbrico como puntos de referencia para enviar publicidad: Si el aparato inalámbrico se encuentra utilizando la repetidora "X" y existe un negocio cerca de esta repetidora "X", se le enviará un banner dinámico con: "Está a menos de 100 mt de nosotros VISÍTENOS!"

1.5.6.3. Cadena de Distribución

Con un sistema de información WAP es posible aumentar el rendimiento de la cadena de distribución a través de aplicaciones orientadas a tener información en tiempo real. Esto se lografacilitando al encargado de distribución aparatos inalámbricos en los cuales pueda llenar y/o enviar información; proporcionando másinformaciónal sistema de la empresa.

CAPÍTULO II
MARCO TEÓRICO

2.1. INTRODUCCIÓN

VISIÓN.-

Ser reconocidos como líderes en el mercado que brinde un producto de aceptación global, posicionando la marca La Pulpita en el mercado nacional e internacional un producto de calidad.

MISIÓN.-

Fabricar y comercializar pulpa de fruta para satisfacer las necesidades de los consumidores en nuestro mercado objetivo, aportando a la vez con fuentes de trabajo y el desarrollo de nuestro país.

2.1.1. RESEÑA HISTÓRICA DE JUGOS NATURALES LA PULPITA

La Pulpita fue creada aproximadamente hace cinco años, por una idea que nació del Señor Lucas Córdova en conjunto con su familia. La pulpa de fruta ha comenzado a comercializarse desde sus inicios como producto terminado y materia prima, tomando en consideración que los clientes que han ganado, reconocen la calidad del producto yse ha logrado un crecimiento del 20 al 30% en los dos últimos años en empresas como La Pulpita cuyo objetivo es ingresar en un mercado competitivo guiado por una estrategia comercial.

2.1.2. RAZÓN SOCIAL

JUGOS NATURALES LA PULPITA
RUC: 1101401840001

2.1.3. ORGANIGRAMA DE LA EMPRESA

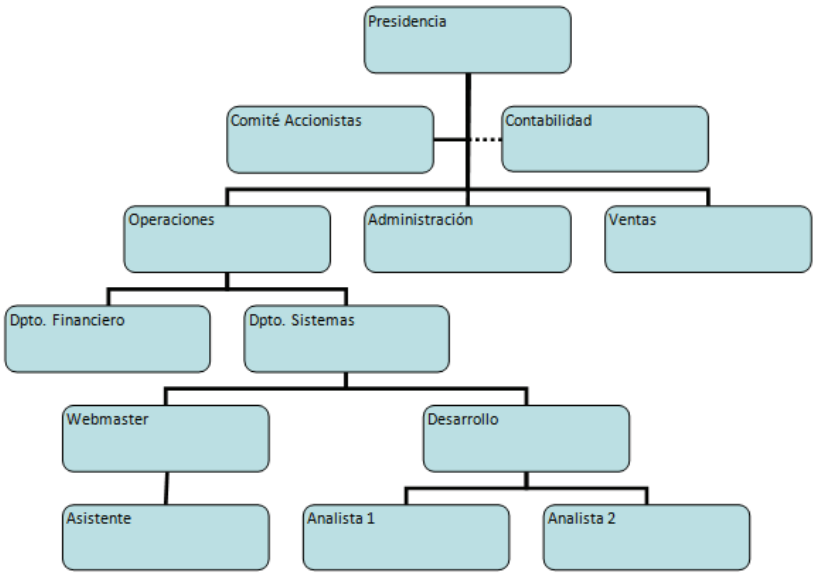


Fig. 2.1 Organigrama de la Empresa
Fuente: Tesistas – La Pulpita

2.1.4. LOCALIZACIÓN Y POLÍTICAS DE LA EMPRESA

Calle Primera N58-01 y Pasaje Décimo, Barrio LA QUINTANA, ubicado en el SectorNorte deQuito – Ecuador.

Las políticas más importantes que jugos naturales La Pulpita tiene para la conservación de la materia prima, producto en proceso y producto terminado son las siguientes:

- Adquisición materia prima de primera calidad para que su producto terminado sea también de calidad.

- Mantenimiento de limpieza enlas áreas en las que se trabaja.

- Optima conservación de los productos.

- Políticas de BPM (buenos procesos de manufactureo “Producción de la materia prima”).

- Buena calidad interna.

- Producción con hojas de trabajo.

- Producción por lotes.

2.2. DESCRIPCIÓN DE LOS PROCESOS DE LA EMPRESA

La Pulpita maneja cuatro procesos para elaborar su producto: Ventas, Compras, Producción, Almacenamiento; los cuales se detallan a continuación:

2.2.1. PROCESO DE VENTAS

El módulo de Ventas es fundamental para la empresa, ya que en este proceso, se toma el pedido de los clientes y se le informa la fecha en la que el producto final será entregado; se realiza de esta manera ya que la mayor parte de los clientes ordena un alto volumen de producto final.

Para los clientes que solicitan una cantidad pequeña de producto final (promedio de 15 o 20 kilos), se realiza la entrega inmediatamente o como máximo al siguiente día.

Además en este proceso se da a conocer las promociones de temporada y también se ofrece al cliente los nuevos productos que salen al mercado.

En el proceso de Ventas se considera tres aspectos, que son los siguientes:

Clientes

La empresa tiene como clientes a Grandes, Medianas y Pequeñas Empresas, con las cuales existe una relación directa por lo tanto posee información completa de las empresas, lo que permite tener una estrecha relación de trabajo.

La información de los clientes mayoristas se la maneja en papel, y cuando se necesita algún dato de dicha información, la búsqueda es lenta y con el simple hecho de tenerla en papel, se corre el riesgo de una pérdida de información.

Pedidos

La empresa ofrece a sus clientes tres formas de receptar los pedidos:

- Receptar un pedido en el momento de una venta anterior.
- Por medio de correo electrónico.
- Mediante vía telefónica.

Para comodidad de los clientes, se aceptan dos formas de pago: Contado y Crédito a 8, 15, 22 o 30 días. La cantidad de días de crédito depende del monto de la compra.

Los pedidos realizados se los registra únicamente en papel, para luego revisar el stock disponible que se posee, este proceso actualmente se lo realiza con hojas de trabajo que consiste en registrar la cantidad de producto que se va a procesar.

Entregas

La entrega de los productos se realiza por medio de los despachadores, que van de empresa en empresa entregando los pedidos. En varios casos son ellos los que toman los próximos pedidos solicitados por los clientes.

2.2.2. PROCESO DE COMPRAS

El proceso de Compra de materia prima para la elaboración del producto final, se realiza directamente con las siguientes fincas:

- Agrícola Santiago Javier
- Agricultores Santa Rosa
- Rosa Cherres
- Rancho San Gabriel Fernando Moreta
- Exofrut

- Agroficial
- Javier Figeroa

Las fincas antes mencionadas son las encargadas de hacer llegar el pedido, mediante camiones de su propiedad, hasta las instalaciones de La Pulpita.

2.2.2.1. Proveedores

La buena relación que mantiene la empresa con sus proveedores, permite manejar cualquier tipo de inconveniente en el menor tiempo posible, con el fin de que ninguna de las dos partes salga afectada.

Se tiene información detallada de cada uno de los proveedores, la misma que se la registra únicamente en papel, y cuando se necesita algún dato de dicha información, la búsqueda a momentos es lenta y además que al tenerla solo en papel, se corre el riesgo de que la información se pierda.

2.2.2.2. Pedidos

El pedido a los proveedores, se realiza por medio de correo electrónico, vía telefónica o pre-venta.

Al igual que en el proceso de Ventas, los proveedores ofrecen a “La Pulpita” dos formas de pago, de acuerdo al monto facturado: Contado y Crédito a 8, 15, 22 o 30 días.

2.2.2.3. Recepción

Para la recepción de la materia prima se tienen gavetas de aproximadamente 25 kilos de capacidad, la cual es inmediatamente verificada y calificada para el proceso industrial.

Dependiendo del volumen de pedidos que se tenga y el grado de maduración de la fruta, ésta puede ser almacenada o procesada de inmediato.

2.2.3. PROCESO DE PRODUCCIÓN

Para el proceso de producción se requieren varios pasos, o fases, las cuales se detallan en el siguiente cuadro:

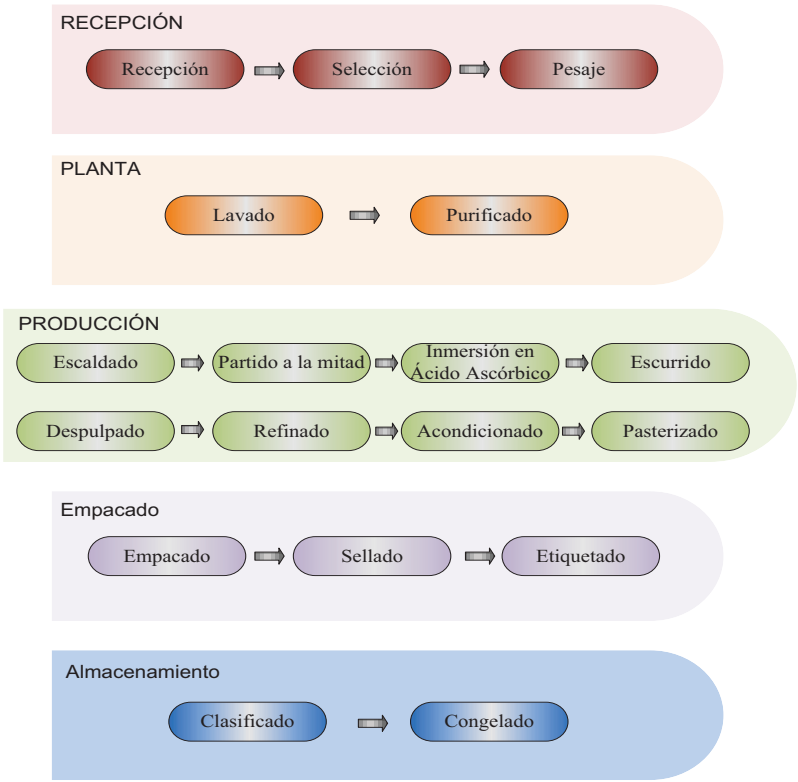


Fig. 2.2Diagrama de procesos de Producción

Fuente: Tesistas – La Pulpita

2.2.3.1. Preparación de la Materia Prima

La empresa tiene un ciclo definido para la preparación de la materia prima, el cual se lo detallaa continuación.

a) **Seleccionar**.- Escoger solamente la fruta completamente madura y que no tenga daño microbiano.

b) **Lavar**.- El proceso para lavar la fruta requiere de agua potable y la técnica utilizada es inmersión o chorro a presión.

c) **Desinfectar**.- Rociar con una solución desinfectante a concentraciones de 15 ppm.

d) **Escaldar.-** Consiste en someter la fruta a un calentamiento corto y posterior enfriamiento. Se realiza para ablandar un poco la fruta y con esto aumentar el rendimiento de pulpa; también se reduce un poco la carga microbiana que aún permanece sobre la fruta y también se realiza para inactivar enzimas que producen cambios indeseables de apariencia, color, aroma, y sabor en la pulpa, aunque pueda estar conservada bajo congelación.

e) **Enfriar**.- Con agua potable, en la marmita o tanque anexo hasta tener la fruta a una temperatura interna de 28 °C.

2.2.3.2. Fabricación

Para el proceso de fabricación la empresa, sigue estos pasos:

a) **Despulpar**.- Pasar la fruta por un tamiz de acero inoxidable.

El tamizado consiste en obtener la pulpa de las frutas en diversas formas y tamaños. Hay varios tipos de tamices para las diferentes formas de obtención de pulpa, por ejemplo: tamices para refinar, tamices para sacar pulpa fibrosa y tamices para despulpar.

Refinar. Consiste en poner el tamiz de menor diámetro, para las frutas que tengan sus semillas demasiado finas, con el fin de que la pulpa de fruta salga con ausencia de materias extrañas. Resultado de esto se obtiene una pulpa sin alteraciones en su color, aroma y textura.

Pulpa Fibrosa. Para la obtención de materia prima que es despachada a empresas que requieren de un producto terminado, el cual será utilizado en la elaboración de mermeladas, pasteles, mousses, etc.

Despulpar. Retira toda aquella materia fibrosa que contienen varias frutas.

b) **Balance de la Pulpa**.- Ajustar los grados brix, agregando un 0.1% de ácido ascórbico hasta lograr un Ph apropiado para el sabor procesado.

	Rangos dePH	
Pulpas de Frutas	**Máximo**	**Mínimo**
Taxo	3.5 +	-38
Fresa	3.8 +	-0.2
Guanabana	4.0	-43
Guayaba	3.8+	-0.2
Mango	3.9	-4.2

	Rangos dePH	
Pulpas de Frutas	**Máximo**	**Mínimo**
Maracuyá	2.8	-3.2
Mora	2.8	-3.2
Papaya	2.8	-3.2
Piña	3.6	-3.8
Tamarindo	0	0
Tomate de Árbol	0	0

c) **Empacado**.- Se empaca en bolsa plástica de alta densidad, opaca, con capacidad de 1000 gramos para el empaque individual o de un kilo para el empaque institucional. Al llenar se debe evacuar el aire al máximo y sellar herméticamente, para luego almacenar a una temperatura de -18°C.

2.2.3.3. Producto Final

El producto final es empacado en las diferentes cajas según su peso, los cuales son:

a) 450 gramos
b) 500 gramos
c) 1000 gramos
d) 5000 gramos
e) Tambores de 50 kilos o 215 kilos

2.2.4. PROCESO DE ALMACENAMIENTO

La selección de almacenamiento tanto para la materia prima, como para el producto terminado se lo realiza dependiendo de algunos factores.

2.2.4.1. Ubicación del Producto Comprado

En el caso de la materia prima comprada, se selecciona para enviarla según su estado de maduración, a dos puntos específicos: fruta lista para proceso y aquellas que tienen un grado bajo de maduración.

2.2.4.2. Ubicación del Producto de Venta

Para ubicar la pulpa final, se hace una división de lotes de producción según su sabor, variedades, volúmenes, rendimiento, y demanda que exista en el mercado.

2.3. HERRAMIENTAS A UTILIZAR PARA EL DESARROLLO DEL CRM

Para la elección del lenguaje de programación que se va a utilizar en el desarrollo del CRM, se tomó en consideración los siguientes aspectos:

2.3.1. VISUAL STUDIO.NET 2010

En Visual Studio 2010 se han realizado varias mejoras al lenguaje más popular, C#, para que los desarrolladores tengan todas las herramientas necesarias para construir un gran software.

Las principales mejoras de C# 4.0 son la interoperabilidad con paradigmas de programación dinámica y la mejora de la programación en Office.

La palabra clave dynamic, permite que el tipo se resuelva dinámicamente en tiempo de ejecución, y no de forma estática como se lo hace en tiempo de compilación.

La búsqueda dinámica, una nueva característica de C# 4.0, le permite utilizar y manipular un objeto desde IronPython, IronRuby, JScript, HTML DOM o desde una biblioteca estándar de .NET de la misma manera, independientemente de donde proceda.

El acceso a las interfaces COM, incluidas las API de automatización de Office, se ha mejorado considerablemente ya que se han agregado argumentos con nombre y opcionales, el tipo dynamic, propiedades indizadas y modificadores refopcionales.[1]

Ya que las máquinas en las que va a funcionar el sistema, tienen instalado Windows XP, .Net Framework es un componente integral de Windows y por lo tanto la compilación y la ejecución es mucho más rápida que Java.

También se puede implementar una aplicación que contiene información de tipos incrustada en lugar de información de tipos que se importa desde un ensamblado de interoperabilidad primario (PIA). Con la información de tipos incrustada, la aplicación puede utilizar los tipos en un motor en tiempo de ejecución sin necesidad de una referencia al ensamblado en tiempo de ejecución. Si se publican varias versiones del ensamblado del runtime, la

[1]http://walterrojasa.blogspot.com/2009/09/nuevas-caracteristicas-en-vb-y-c-en-vs.html

aplicación que contiene la información de tipos incrustada puede funcionar con las diferentes versiones sin que sea necesario volver a compilarla.[2]

Por otro lado se tiene ClickOnce que es una tecnología de implementación que permite crear aplicaciones basadas en Windows de actualización automática, que se pueden instalar y ejecutar con interacción mínima por parte del usuario.

La implementación ClickOnce supera tres problemas importantes de la implementación:

- **Dificultades para actualizar las aplicaciones.** Con la implementación de Microsoft Windows Installer, siempre que se actualiza una aplicación, el usuario debe volver a instalar toda la aplicación; con la implementación de ClickOnce, es posible suministrar actualizaciones automáticamente. Sólo se descargan las partes de la aplicación que hayan cambiado y, a continuación, la aplicación completa y actualizada vuelve a instalarse desde una nueva carpeta en paralelo.

- **Impacto en el equipo del usuario.** Con la implementación de Windows Installer, las aplicaciones normalmente utilizan componentes compartidos, con sus posibles conflictos de versiones; gracias a la implementación ClickOnce, cada aplicación es autosuficiente y no puede interferir con otras aplicaciones.

- **Permisos de seguridad.** La implementación de Windows Installer requiere permisos administrativos y sólo permite una instalación de usuario limitada; la implementación de ClickOnce habilita a los usuarios que no tienen derechos administrativos para realizar la instalación y

[2] http://msdn.microsoft.com/es-es/library/we86c8x2.aspx

sólo concede los permisos de seguridad de acceso del código necesarios para la aplicación.Antes, estos problemas hacían que los desarrolladores decidieran crear aplicaciones Web en lugar de aplicaciones basadas en Windows, sacrificando la interfaz de usuario enriquecida que posibilita una instalación más fácil. Con aplicaciones implementadas mediante ClickOnce, puede aprovechar lo mejor de ambas tecnologías. [3]

Para el módulo WAP, serán instalados en los teléfonos celulares Windows Mobile 6.5, además se desarrollará con las herramientas que ofrece Visual Studio 2010 para aplicaciones móviles, que entre otras facilidades tiene, (DeviceEmulator Manager) que permite trabajar con los distintos emuladores existentes en el sistema, desde emuladores de PhoneEdition 2003-SE o WM 5.0 hasta dispositivos Windows CE 5.0. Este 'Manager' permite trabajar con los emuladores como si realmente se lo estuviera haciendo con un dispositivo real.

A todo esto se le suma que Visual Studio 2010 ofrece mucha información para consultar por medio de MSDN y su soporte técnico.

2.3.2. SQL SERVER 2008

Para la base de datos, se ha seleccionado Microsoft SQL Server 2008, ya que la empresa “La Pulpita” cuenta con la licencia, además viene con nuevas funcionalidades que potencian las aplicaciones y brindan una amplia variedad de posibilidades de solución a los desafíos del desarrollo.

Entre las principales mejoras se tiene las siguientes:

[3] http://msdn.microsoft.com/es-es/library/t71a733d.aspx

El tipo de dato FILESTREAM soluciona aquellos problemas de almacenar datos no estructurados como por ejemplo guardar en la base de datos la ruta donde el archivo se aloja en el sistema de archivos (file system); permitiéndole a las aplicaciones utilizar las API de streaming con el mismo rendimiento que el sistema de archivos, y a su vez mantener una consistencia transaccional y el mismo esquema de seguridad entre la información estructurada y la no estructurada.

En cuanto a la seguridad de los objetos almacenados, el campo de tipo FILESTREAM tiene las mismas características y privilegios que cualquier otro tipo de datos.

El tipo de dato FILESTREAM es implementado como una columna varbinary(max), y totalmente integrado en el motor. Lo cual permite utilizar todas las técnicas de copias de respaldo y restauración para resguardar los datos.[4]

Con SQL Server 2008 en el lenguaje T-SQL es aceptada la inicialización y asignación de variables, es decir, se puede declarar y asignar la variable en la misma línea, por ejemplo:

```
declare @fecha = getdate()
```

[4] http://bernardorobelo.blogspot.com/2010/10/caracteristicas-del-nuevo-tipo-de-datos.html

SQL Server 2008 posee un nuevo comando que permite hacer inserciones y actualizaciones dependiendo si el registro existe o no, a este comando se lo conoce como Comando Merge.

SQL Server 2008 tiene un nuevo tipo de parámetro, el tipo Tabla, el cual puede ser declarado como una tabla previamente definida. Este parámetro puede ser utilizado para enviar varias filas de información a una sentencia T-SQL o rutina.

Estos parámetros reemplazan a la ya acostumbrada técnica de creación de tablas temporales, además con las siguientes ventajas:

- Son fuertemente tipificadas
- Se puede especificar un ordenamiento y clave única
- Reduce las llamadas al servidor
- Evita la re compilación de la sentencia
- No necesita de un bloqueo desde el cliente para inicializar los valores de la tabla.

El Lenguaje de Consultas Integrado LINQ, permite realizar consultas a una fuente de datos utilizando lenguajes de programación manejados tales como C# (en nuestro caso), en lugar de usar sentencias SQL. Se debe utilizar una sintaxis consistente para obtener datos de diversas fuentes, incluyendo datos

relacionales, entidades, XML, DataSets de ADO.NET, y colecciones de objetos en memoria.[5]

2.4. METODOLOGÍA OMT (OBJECT MODELING TECHNIQUE)

2.4.1. INTRODUCCIÓN

La metodología OMT fue creada por James Rumbaugh y Michael Balaha en 1991 y es considerada una de las metodologías de análisis y diseño orientados a objetos, más maduros y eficientes que existen en la actualidad. Lo más importante que se tiene en esta metodología, es su carácter de abierta (no propietaria), esto facilita su evolución para adaptarse a las necesidades actuales y futuras de la ingeniería de software.

Las fases que conforman la metodología OMT son las siguientes:

2.4.2. ANÁLISIS

Implicarála realización de un modelo del dominio del problema, con lo que se llegará a mostrar las características más importantes del negocio de la empresa. Aquí se detallará la forma en que trabaja al momento la empresa y como lo hará después con el sistema. El modelo debe ser entendible y en lo posible criticado por expertos en el dominio del problema que no tengan conocimientos informáticos, dentro de la empresa.

En esta fase, se obtiene la descripción inicial de los requerimientos de La Pulpita:

[5] http://msmvps.com/blogs/cwalzer/archive/2008/04/01/nuevas-caracter-237-sticas-para-desarrolladores-en-sql-server-2008.aspx

- Administración de usuarios que ingresarán al sistema de acuerdo a su área de trabajo.
- Manejo sistematizado de órdenes de compras de materia prima.
- Manejo sistematizado de órdenes de ventas y fabricación de producto terminado.
- Control de stock tanto de materia prima como de producto terminado

Además se construirá el modelo de clases y objetos con sus respectivos enlaces y asociaciones, identificando los atributos que intervienen en el CRM de La Pulpita, si es necesario se tendrá que organizar y simplificar las clases utilizando herencia.

2.4.3. DISEÑO DEL SISTEMA

Una vez tomada la decisión en base al análisis, se procederá con la arquitectura para el desarrollo del sistema. Durante esta fase el sistema se organiza en subsistemas basándose tanto en la estructura del análisis como en la arquitectura propuesta. Se seleccionará una estrategia para afrontar el problema.

Los pasos que se llevarán a cabo en la fase de Diseño del sistema para el caso de La Pulpita son los siguientes:

Organizar el sistema en subsistemas o módulos, que comparten alguna propiedad en común.

Se realizará un estudio técnico de la concurrencia del sistema.Se asignará cada subsistema a una tarea, buscando satisfacer las necesidades de rendimiento y minimizar las comunicaciones entre subsistemas.

Se llegó a la selección de la estrategia a utilizarse para la implementación y administración de los almacenes de datos.

Se implementará control de código para el desarrollo del sistema, en la actualidad Visual Studio 2010 ofrece su propio Source Control.

2.4.4. DISEÑO DE OBJETOS

Se construirá un modelo de diseño basado en el modelo de análisis, pero incorporando detalles de implementación. El diseño de objetos se centrará en las estructuras de datos y algoritmos que son necesarios para implementar cada clase. OMT describe la forma en que el diseño será implementado en el lenguaje orientado a objetos en este caso Visual Studio C# .Net 2010y la base de datosSql Server 2008.

2.4.5. IMPLEMENTACIÓN

Las clases de objetos y relaciones desarrolladas durante el análisis de objetos se traducirán finalmente en una implementación concreta. Durante la fase de implementación es importante tener en cuenta los principios de la ingeniería del software de forma que la correspondencia con el diseño sea directa y el sistema implementado sea flexible y extensible. Se reutilizará el código de manera que esta práctica potencialice la utilización del sistema, usando AOO y DOO, así como también la correspondencia entre el dominio del problema y el sistema informático. Tomando en consideración que si se realiza una implementación de mala calidad se perderán las ventajas antes descritas.

La metodología OMT emplea tres clases de modelos para describir el sistema:

- **Modelo de objetos.** Se describirá la estructura estática de los objetos del sistema (identidad, relaciones con otros objetos, atributos y operaciones). El modelo de objetos proporcionará el entorno esencial en el cual se podrá situar el modelo dinámico y el modelo funcional. El objetivo es llegar a capturar aquellos conceptos del mundo real que sean importantes para la aplicación. Se representará mediante diagramas de objetos.

- **Modelo dinámico.** Detallará los aspectos del sistema que tratarán de la temporización y secuencia de operaciones (sucesos que marcarán los cambios, secuencias de sucesos, estados que definirán el contexto para los sucesos) y la organización de sucesos y estados. Capturará el control, de aquel aspecto en que el sistema describirá las secuencias de operaciones que se producirán sin tener en cuenta lo que hagan las operaciones, aquello a lo que afectará o la forma en que se llegará a implementar. Se representará gráficamente mediante diagramas de estado.

- **Modelo funcional.** Describirá las transformaciones de valores de datos (funciones, correspondencias, restricciones y dependencias funcionales) que ocurren dentro del sistema. Capturará lo que hace el sistema, independientemente de cuándo se haga o de la forma en que se haga. Se representará mediante diagramas de flujo de datos.[6]

[6] http://www.monografias.com/trabajos13/metomt/metomt.shtml

2.5. METODOLOGÍA OOHDM (OBJECT ORIENTED HYPERMEDIA DESIGN METHODOLOGY)

2.5.1. INTRODUCCIÓN

OOHDM, Método de Diseño Hipermedia Orientado a Objetos es una metodología propuesta por Schwabe y Rossi en 1998, tiene por objetivo simplificar y a la vez hacer más eficaz el diseño de aplicaciones hipermedia.

OOHDM como metodología de desarrollo de aplicaciones de hipermedia, proporciona ventajas como:

- La recuperación de la información puede realizarse sin problemas.
- Se pueden crear enlaces entre nodos cualesquiera.
- La modularidad y la consistencia se potencian.
- Marco idóneo para la autoría en colaboración.
- Soporte a diferentes modos de acceso a la información.

Al igual que RMM, este método se inspira en el modelo HDM, pero lo que le distingue claramente del primero es el proceso de concepción orientado a objetos.

Cada etapa de la concepción define un esquema objeto específico en el que se introducen nuevos elementos (clases).

En la siguiente imagen se muestran las relaciones entre los esquemas conceptuales, navegacionales y los objetos de interfaz en OOHDM.

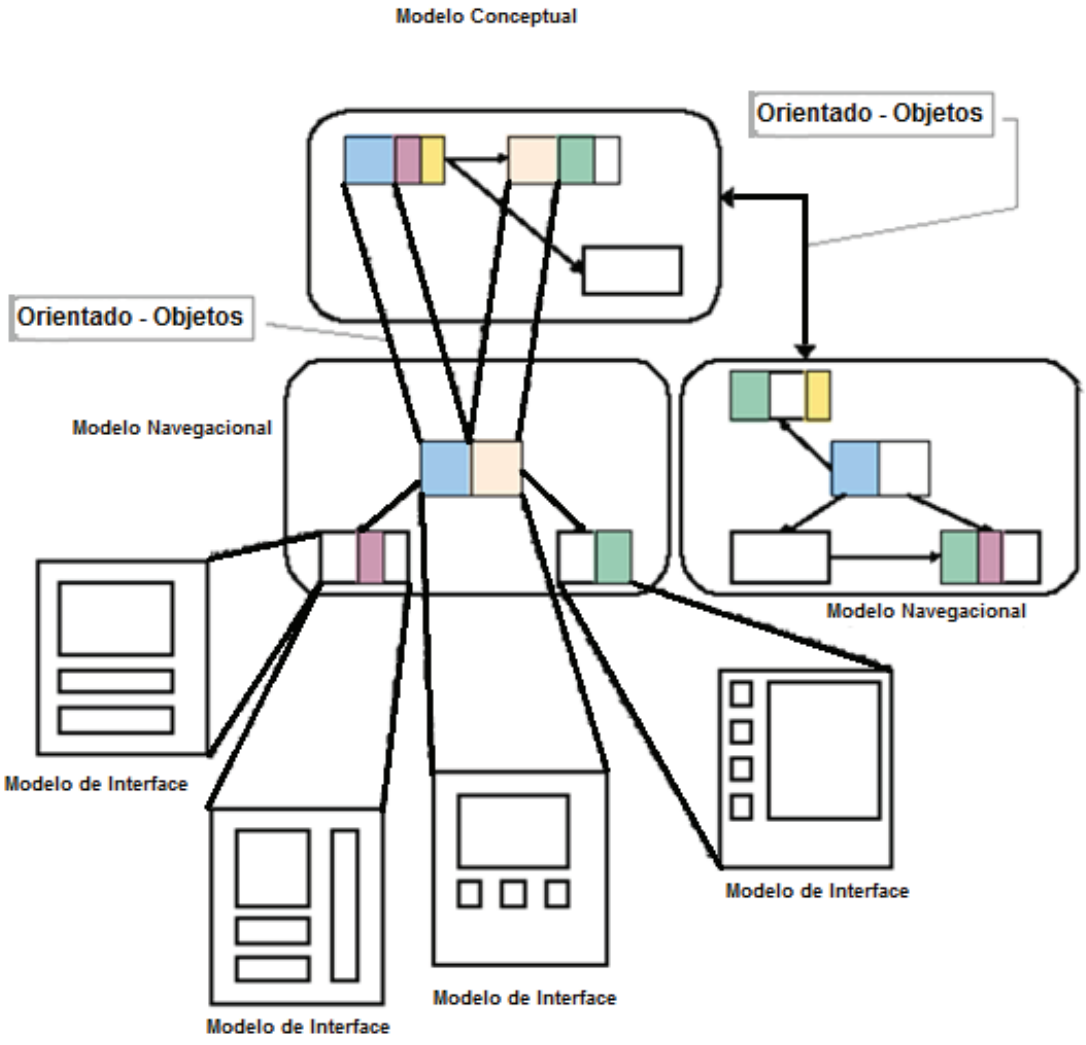


Fig. 2.3Relaciones entre los esquemas conceptuales
Fuente:Navegacional y los objetos de interfaz en OOHDM[7]

[7]Relación entre los esquemas y objetos OOHDM

2.5.2. DEFINICIONES

- **Nodos:** Los nodos son contenedores básicos de información de las aplicaciones hipermedia, un nodo es definido mediante la combinación de atributos de clases diferentes relacionadas en el modelo de diseño conceptual. Los nodos contendrán atributos de tipos básicos (donde se pueden encontrar varios tipos como imágenes o sonidos) y enlaces.

- **Enlaces:** Los enlaces muestran la relación de navegación que puede explorar el usuario. En un mismo esquema conceptual existirá diferentes esquemas navegacionales y los enlaces son imprescindibles para poder tener vistas diferentes.

- **Estructuras de Acceso:**Permite al usuario encontrar de forma rápida y eficiente la información deseada. Los índices y las guías de ruta son parte de esta estructura. La estructura de acceso se encontrará como clases, compuestas por un conjunto de referencias a objetos que son accesibles desde ella y una serie de criterios de clasificación de las mismas.

- **Contexto Navegacional:** Para diseñar bien la aplicación hipermedia, hay que prever los caminos que el usuario puede seguir, así es como únicamente se evitará información redundante o que el usuario se pierda en la navegación. En OOHDM el contexto navegacional está compuesto por un conjunto de nodos, de enlaces de clases de contexto y de otros contextos navegacionales.

Referencia Web: Daniel Schwabe y Gustavo Rossi: The Object-Oriented Hypermedia Design Model (OOHDM). http://www-di.inf.puc-rio.br/schwabe//papers/TAPOSRevised.pdf

- **Clase de Contexto:** Es otra clase especial que complementará la definición de un tipo de navegación. Por ejemplo, sirve para indicar qué información estará accesible desde un enlace y desde dónde se puede llegar a él.

2.5.3. FASES DE LA METODOLOGÍA OOHDM

OOHDM como técnica de diseño de aplicaciones hipermedia, plantea un conjunto de tareas que reducen notablemente los tiempos de desarrollo al tener como objetivo principal la reusabilidad de diseño, y así simplificar el costo de evoluciones y mantenimiento.

Esta metodología propone el diseño de una aplicación de este tipo a través de cinco fases que se desarrollan de un modo iterativo. Estas fases son:

Fig. 2.4Fases del OOHDM

Fuente: Tesistas

CAPÍTULO III
DISEÑO

3.1. INTRODUCCIÓN

Los diagramas son una representación gráfica de los pasos que se siguen en toda una secuencia de actividades, dentro de un proceso o un procedimiento, identificándolos mediante símbolos de acuerdo con su naturaleza, incluyendo a esto toda la información que se considera necesaria para el análisis.

La metodología OMT, emplea tres clases de modelos para describir el sistema:

- Modelo de Objetos.- describirá la estructura estática de los objetos del sistema, como identidad, relaciones con otros objetos, atributos y operaciones con el objetivo de capturar aquellos conceptos del mundo real que sean importantes para la aplicación.

- Modelo Dinámico.- se manejarán diagramas de estado que representarán gráficamente la temporización y secuencia de operaciones y la organización de sucesos y estados.

- Modelo Funcional.- se utilizarán diagramas de flujo de datos que describirán las transformaciones de los valores de los datos que ocurrirán dentro del sistema. Se registrará la información de lo que hace el sistema independientemente de cuándo y cómo lo haga.

3.2. MODELO DE OBJETOS Y SUS RELACIONES

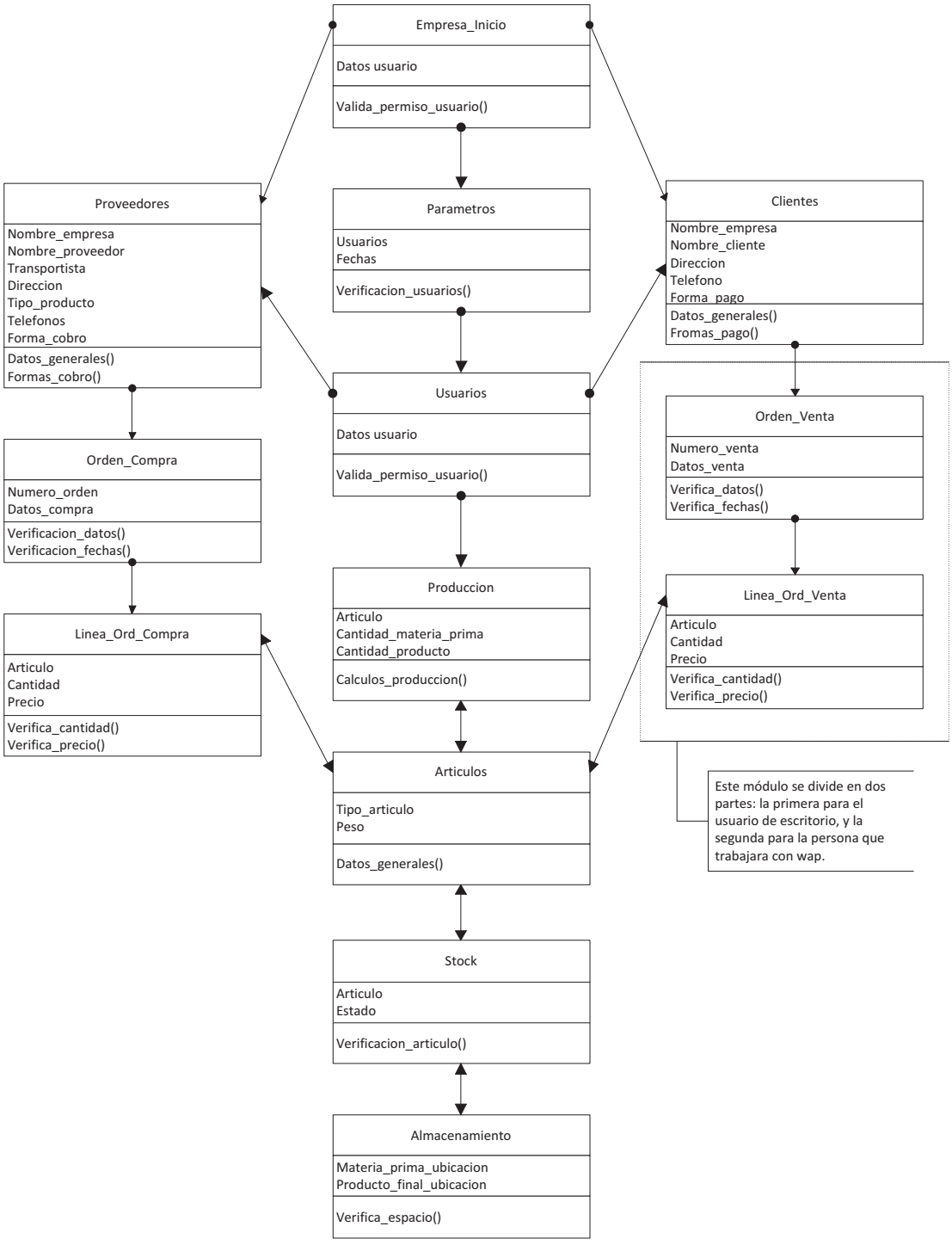


Fig. 3.1 Modelo de Objetos

Fuente: Tesistas

3.3. DIAGRAMA DE FLUJO DE DATOS

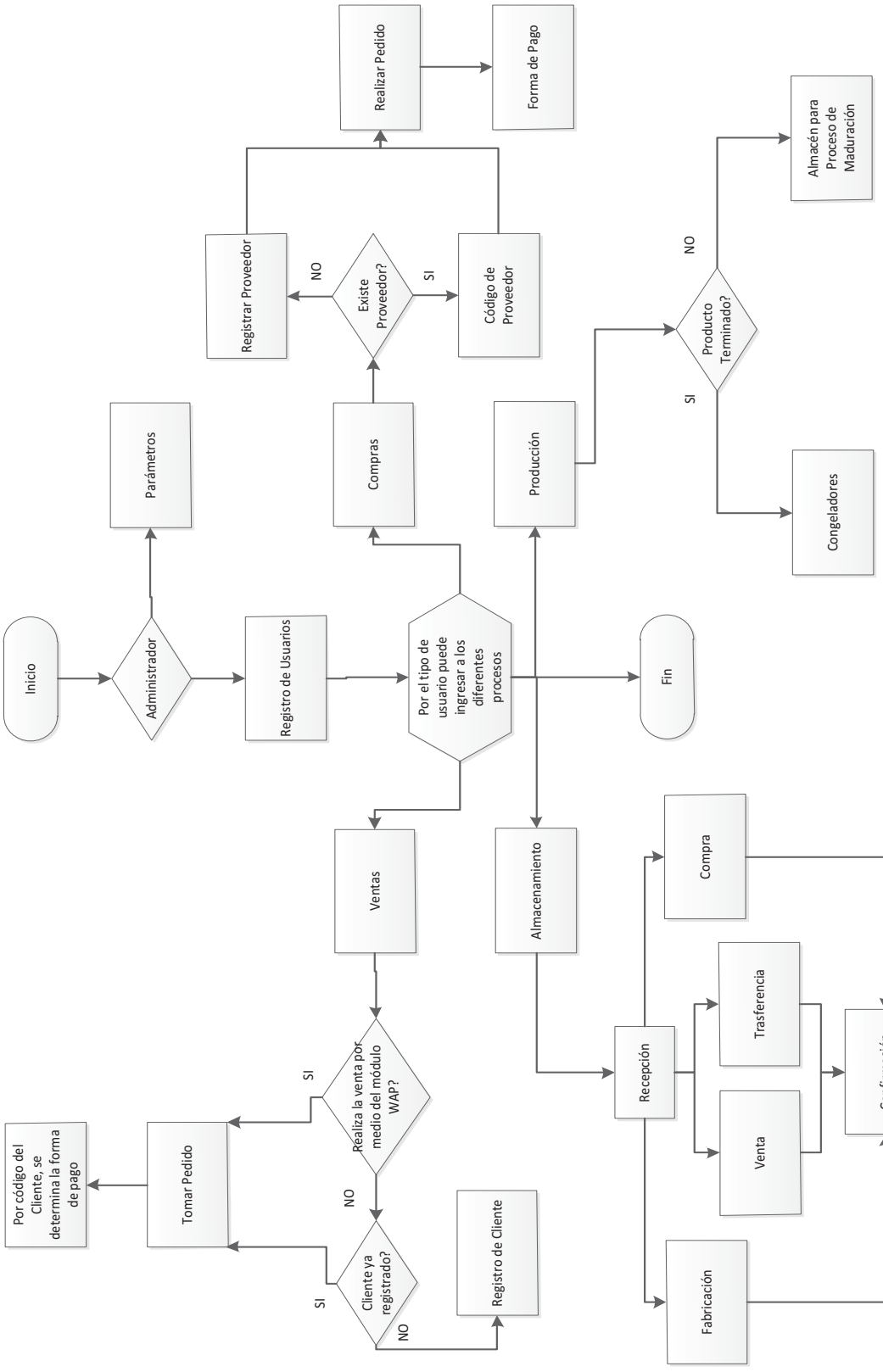


Fig. 3.2 Diagrama de Flujo de Datos

Fuente: Tesistas

3.3.1. MÓDULO DE VENTAS

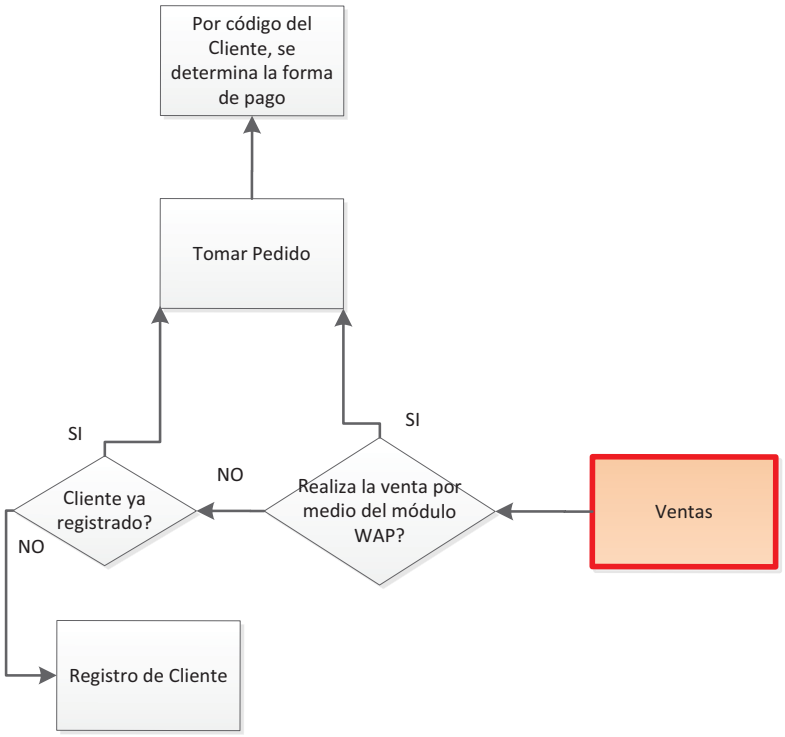


Fig.3.3 Diagrama de Ventas

Fuente: Tesistas

Las órdenes de venta son utilizadas para ingresar los pedidos, en éstas se colocan los datos generales del cliente así como las cantidades y precios de los artículos que desea.

El origen de las órdenes de ventas puede serde diferentes formas: por llamada telefónica, plan de ventas, cotizaciones e Intercambio electrónico de datos.

Para realizar cada orden de venta se necesita de varios datos previos, como:

a) Datos de Artículo.- Esto es necesario para completar la información del artículo a ser vendido. Los datos más importantes son:

- Datos de precios.- Incluye datos como las unidades de venta y de precio de venta.

- Datos de creación de órdenes.- Contiene datos como la unidad en que se vende, las cantidades mínimas que se pueden ordenar, el almacén, entre otros.

- Configuración por Artículo.- para futuros productos se debe poner si estos tienen IVA, y el porcentaje que debe tener el mismo.

b) Oficinas de Venta.- Una oficina de ventas es necesaria para realizar transacciones con los clientes. Por el momento sólo se tiene la matriz, que es la planta de producción, único punto de venta fijo, las demás ventas se realizan con vendedores que se desplazan al lugar del cliente.

c) Control de Márgenes.- El control de márgenes permite definir el tope máximo y el mínimo en el que se debe situar el precio de venta de un artículo al momento del ingreso de la orden de venta. Este control puede servir para tres efectos diferentes: Verificar si el precio de una orden o una cotización se encuentra dentro del margen permitido. Verificar el margen bruto de una línea de orden o cotización asegurándose de que exista un margen de utilidad con respecto al costo del artículo. Verificar si el margen bruto total de una orden de venta o cotización se encuentre arriba del costo total de los artículos que esta contiene.

Cuando se hacen verificaciones y se encuentra que los datos están fuera de los márgenes permitidos, el sistema puede realizar diferentes acciones de

aviso y de bloqueo dependiendo del perfil definido para cada usuario. En perfiles de usuario existe un parámetro que indica si un usuario determinado puede desbloquear órdenes que han sido bloqueadas por margen.

3.3.1.1. Cotizaciones

Las cotizaciones son utilizadas para proveer al cliente de los detalles suficientes de los productos para tomar su decisión de compra. Una cotización puede ser creada en respuesta a una requisición del cliente o como una herramienta de ventas para iniciar un proceso de venta con un cliente potencial. Una cotización define las fechas, términos y alternativas asociadas con él (los) artículos en cuestión.

3.3.1.2. Estatus de Líneas

En las líneas de cotización y de venta se define información específica del artículo que se está cotizando o vendiendo. Esta información incluye la cantidad, precio y condiciones de entrega además del código del artículo en cuestión.

Es posible proponer alternativas al cliente mediante líneas alternativas sólo en el caso de cotizaciones.

Dentro de las líneas de cotización y venta se puede observar el estatus de las mismas, estos estatus pueden ser los siguientes:

- **Libre** - Es el estatus que adquiere una línea cuando recién se crea.

- **Confirmada** - Este estatus lo adquiere cuando el cliente al que se le cotizó ha aceptado la cotización o venta y este resultado es registrado en el sistema.

- **Cancelada** - Cuando el cliente no aceptó.

- **Procesada** - Cuando la línea ha sido aceptada, ésta se convierte mediante un proceso en una línea de orden de venta, es entonces cuando adquiere este estatus. (Sólo en casos de cotizaciones)

- **Facturado –** En el momento que la orden de venta, ha pasado por todos los demás estatus, y se encuentra como ya facturada. (Sólo órdenes de venta)

3.3.1.3. Impresiones

Se necesita imprimir algunos reportes como:

- Listar cotizaciones, y órdenes de venta

- Listar líneas de cotizaciones y órdenes de venta.

En estas sesiones se tienen rangos de selección para delimitar la información que se requiere imprimir. Es posible seleccionar el estatus de lo que se desea listar.

3.3.2. MÓDULO ALMACENAMIENTO

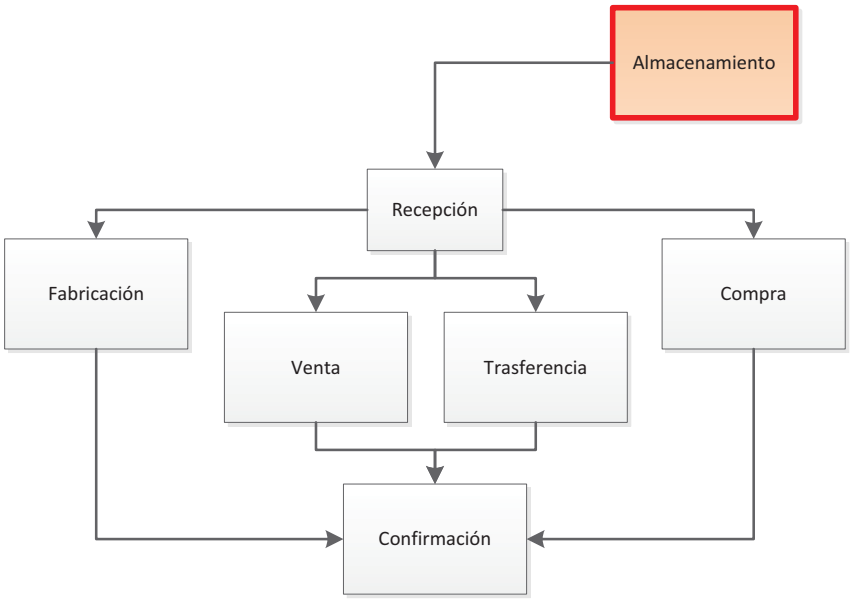


Fig. 3.4 Diagrama de Almacenamiento

Fuente: Tesistas

Por el momento existe un solo almacén, que es la matriz, pero en un futuro se van a tener más almacenes, los cuales deben ser registrados en el sistema. Cada almacén puede tener varias zonas, y dentro de estas zonas contendrá ubicaciones, las cuales se las debe identificar en el sistema.

El almacén puede contener zonas, y esta a su vez pueden tener ubicaciones, así mismo un almacén puede tener ubicaciones directamente. Se entiende a la zona como un lugar macro donde se coloca materia prima, producto terminado, o en proceso, y a la ubicación como espacio micro con las mismas instancias que tiene una zona.

3.3.2.1. Lotes

El control de lotes se utiliza para tener una clasificación muy específica de los artículos. Dependiendo de su origen, un lote puede ser de tipo comprado o fabricado. Este módulo permite la identificación de artículos mediante un código de lote.

El proceso de configuración de lotes será útil para realizar las siguientes actividades:

- Evaluación del inventario por lotes.
- Bloqueo y desbloqueo de lotes.
- Movimiento de entrada y salida de mercadería.

3.3.2.2. Transacciones

Los tipos de transacciones definen las formas básicas por las cuales los módulos del sistema pueden interactuar. Hay 4 posibles valores:

- Recepción.
- Entrega
- Transferencia
- Ítem de Transferencia

3.3.2.3. Órdenes de Almacenamiento

Una orden de almacén controla el movimiento de mercancías. Cada orden de almacén se compone de una cabecera y de líneas de la orden. La cabecera contiene información tal como el tipo de la orden, el socio de negocio y la información de envío. Esta sesión se puede utilizar para visualizar y para mantener órdenes existentes del almacén o para incorporar nuevas órdenes del mismo.

La fuente para este pedido de movimiento del inventario. Los valores posibles son:

- Ventas.
- Compra.
- Servicio.
- Fabricación.
- Transferencia.

3.3.2.4. Unidad de Almacenamiento

La Cantidad de la Orden indica la unidad de almacenaje (como piezas o libras) y la cantidad que se moverá.

3.3.2.5. Ajustes de Inventario

Órdenes Correcciónquesirven para reajuste de stock. El campo de razón del ajuste identifica la razón por la que el inventario fue ajustado. Este valor se lleva adelante a las líneas de orden de ajuste del inventario mientras que se agrega cada nueva línea. Sin embargo, cada

línea de la orden se puede fijar independientemente en caso de necesidad.

3.3.2.5. Inventario por Almacén, Ubicación, Artículo y Lote

Esta sesión permite obtener la información de cuál es la cantidad que se tiene en inventario. Se encuentra disponible para todos los usuarios, donde se tiene una punta común, que es el nivel mínimo del inventario registrado en el sistema. Las puntas comunes están determinadas por los diversos artículos o lotes dentro de una sola ubicación en un almacén. Un total de todo lo que contiene el almacén, la ubicación y el artículo se muestra enla parte inferior de la sesión.

3.3.3. MÓDULO COMPRAS

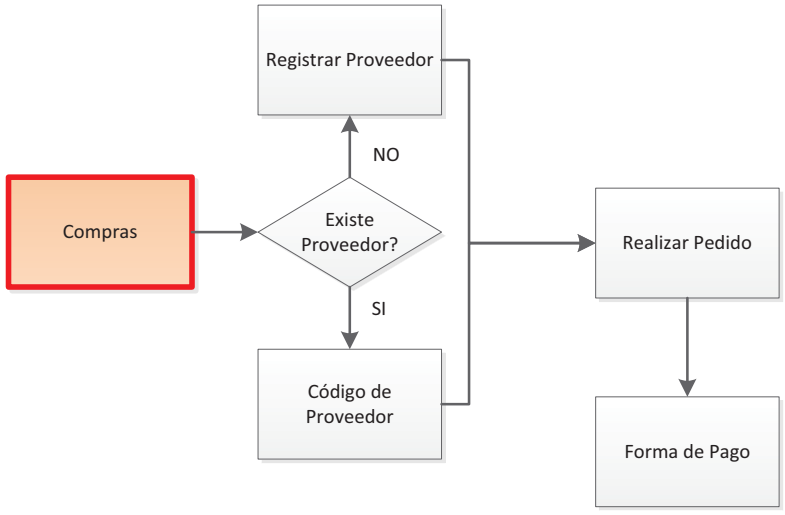


Fig. 3.5 Diagrama de Compras

Fuente: Tesistas

3.3.3.1. Módulo de Compras

La orden de compra es el documento mediante el cual se solicita mercancía a los proveedores, estableciendo las condiciones bajo las cuales se llevará a cabo la transacción de compra.

La ejecución de una orden de compra puede contener diversos pasos para llegar a completarse. Algunos de estos pasos son opcionales pero otros son obligatorios que dependerá del tipo de orden al que sean asociados.

A continuación se describen los pasos básicos para la ejecución de una orden así como algunos documentos y pasos de apoyo que servirán para que dependiendo de las necesidades de la empresa.

Una orden de compra al igual que la de venta se compone de dos partes, la cabecera y las líneas de la orden. La cabecera contiene información sobre el proveedor, tipo de orden de compra, fechas de recepción de mercancías, tanto la esperada como la confirmada y las condiciones generales de entrega.

La sesión de la cabecera consta de diferente pantalla en donde se capturan las condiciones de compra, envío, facturación y control de cambios de la orden.

- **Tipo de Orden -** El tipo de orden determina cuales actividades son parte del procedimiento de compra y su número de secuencia. Una vez que se ingresan las líneas de la orden, el tipo de orden ya no puede ser cambiado.

- **Condiciones de Entrega -** Es un acuerdo con el socio de negocio concerniente a la forma en que los bienes serán entregados.

- **Pantalla "Compras" -** Esta pantalla incluye información del proveedor y descuentos por el total de la orden.

- **Pantalla "Envíos" -** Esta pantalla incluye información sobre la compañía de envíos.

- **Pantalla "Facturación" -** Información sobre el proveedor que envía las facturas a su organización.

- **Pantalla "Pagos" –** Información sobre el proveedor al que tiene que enviar el pago.

3.3.3.2. Líneas de Orden de Compra

Una vez que la cabecera de la orden de compra es ingresada, es necesario especificar información en las líneas de la orden. Los datos más importantes son el código del artículo, cantidad y precio.

Existen datos adicionales que han sido predeterminados en el catálogo de proveedores y en el catálogo de artículos. Estos datos pueden ser modificados cuando se ingresan las líneas de la orden.

- **Importe -** Es el valor total de la línea de la orden de compra. Si no se ingresa una cantidad, este campo es obligatorio. El importe es calculado de acuerdo a la siguiente fórmula:

$$Importe = (cantidad \times precio) - \frac{cantidad \times precio \times descuento}{100}$$

- **Pantalla "Envío" –** Se pueden definir datos diferentes a los de la cabecera, como el Expedidor o el Almacén, para cada una de las líneas.

- **Pantalla "Finanzas" -** Solamente aplica para los artículos de costo.

3.3.3.3. Imprimir Órdenes de Compra

Cuando se desea imprimir una copia de una orden de compra existente, se puede hacer desde la sesión Listar órdenes de compra. Después de listar una orden de compra, algunos datos pueden ser modificados.

Durante el proceso de impresión es importante indicar de manera correcta la información que desea imprimir. Por ejemplo, sólo las líneas modificadas, sólo las recién ingresadas, todas las líneas, etc. También puede imprimir la cantidad pedida o sólo la retro orden

3.3.4. MÓDULO PRODUCCIÓN

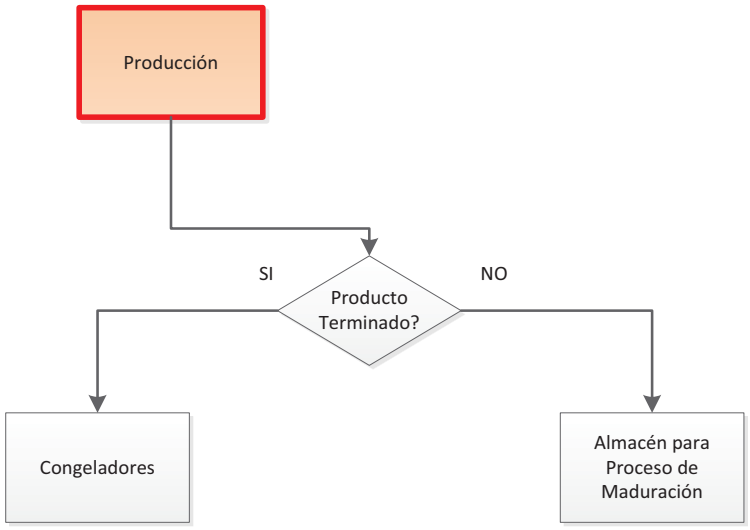


Fig. 3.6 Diagrama de Producción

Fuente: Tesistas

Por cada tipo de fruta hay uno o más procesos de industrialización, cada uno de los cuales presenta problemas específicos en relación con el control de procesos y generación de residuos.

El éxito en la obtención de pulpas de alta calidad comienza en la disponibilidad de frutas de excelentes características gustativas. Junto a esta disponibilidad está el cuidado que se tenga en mantener la alta calidad en los pasos previos a la llegada a la fábrica de procesamiento.

En la planta, la fruta es rodeada de condiciones óptimas para el favorecimientode sus características sensoriales. Si la fruta llegó a la planta de producción tierna, habrá que propiciar su maduración adecuada. Si ya está madura, se procurará evitar su deterioro microbiológico mediante la disponibilidad de un ambientecompletamente limpio.Durante todo el tiempoque permanezca la fruta expuesta a este ambiente, se controla los grados de

maduración, sin madurantes artificiales para que esté en óptimas condiciones para el despulpado.

3.3.4.1. Recepción

Esta operación tiene una importancia grande en cualquier actividad productiva de la empresa. Consiste en recibir del proveedor la materia prima requerida, de acuerdo a las especificaciones entregadas de antemano por la empresa. El acto de recepcionarimplica la aceptación de lo entregado de acuerdo a las especificaciones y políticas de la empresa dadas al proveedor. Se pueden dar recepciones completas o parciales, cualquiera que sea el caso, se debe realizar un pago por lo recibido.

3.3.4.2.Pesaje

Esta operación implica la cuantificación de varios aspectos, entre los cuales se cuenta, el volumen comprado, el volumen de la calidad adecuada para el proceso dedatos sobre el volumen para la cuantificación del rendimiento.Por último, lo más importante, el volumen por pagar al proveedor y el volumen que ha de ingresar al proceso.

Se efectúa con cualquier tipo de balanza de capacidad apropiada y de precisión a las décimas ycentésimas de gramo.

La forma de pesar puede ser en los mismos empaques en que la fruta llega a planta o pasándola con cuidado a los empaques adecuados de la fábrica que se puedan manejar y apilar cómodamente. Debe evitarse el manejo brusco de los empaques para evitar magulladuras o roturas de las frutas.

3.3.4.3.Selección

Se separa las frutas sanas de las descompuestas.Se puede efectuar sobre mesas, disponiendo de recipientes donde los operarios puedan colocar la fruta descartada.

Los instrumentos para decidir qué frutas rechazar son en principio la vista y el olfato de un operario. Él debe ser muy consciente de la responsabilidad de su trabajo e influencia en la calidad de la pulpa final.

Hay ciertas frutas costosas que por su tamaño grande pueden pasar la prueba pero deben ser "arregladas" retirando cuanto antes las fracciones dañadas.

3.3.4.4. Clasificación

Permite separar entre las frutas que pasaron la selección, aquellas que están listas para proceso, en razón de su grado de madurez y las verdes o aún pintonas que deben ser almacenadas.

Aquí también los instrumentos más ágiles y económicos son los sentidos de los operarios. El color, aroma o dureza de las frutas permiten elegir las frutas adecuadas. Estas características exteriores específicas de las frutas se pueden comprobar por controles en el laboratorio, que responden a un grado de madurez adecuado para la obtención de pulpas de alta calidad. Una guayaba amarilla, sana, olorosa y ligeramente blanda le indica al operario que es adecuada para proceso. Aquí no importan el tamaño o la forma.

3.3.4.5.Almacenamiento

Puede aplicarse para acelerar o retardar la maduración de las frutas en la fábrica. Se pueden someter a la primera, frutas sanas que han llegado a la fábrica, tiernas para que maduren. Otras veces es conveniente retardar la maduración un determinado tiempo a fin de procesar paulatinamente la fruta que por razones de cosecha se adquirió en grandes cantidades.

La aceleración de la maduración se logra generalmente ajustando la temperatura y humedad de una cámara donde se puede almacenar la fruta. Las condiciones del ajuste son específicas para cada especie, pero por lo general se acercan a los 25 ºC y la humedad relativa se eleva a 90%. En los casos de frutas climatéricas, también se puede ajustar la composición de la atmósfera de gases que rodean a las frutas.

El retardo de la madurez se hace principalmente con la disminución de la temperatura y ajuste de la humedad relativa de la cámara. Hay casos en que se puede controlar modificando la composición de la atmósfera que rodea las frutas. Se disminuye el contenido de oxígeno y aumenta el de anhídrido carbónico y nitrógeno. En cualquier caso es crítica la higiene y limpieza de la cámara.

Lograr resultados esperados de la maduración exige que se controlen las condiciones durante las cuales permanecen las frutas en almacenamiento. Es definitivo que las frutas ubicadas en la cámara puedan ser afectadas por las condiciones que existen a su alrededor. Para esto las frutas deben estar colocadas en cestillos por donde puedan circular los gases a la temperatura necesaria.

3.3.4.6. Enjuague

A la fruta desinfectada se le debe retirar los residuos de desinfectante y microorganismos mediante lavado con agua potable. Si es posible por aspersión con agua que corra y se renueve. No es conveniente enjuagarla sumergiéndola en tanques de agua que cada vez estará más contaminada.

3.3.4.7. Corte

Algunas frutas como el maracuyá debe ser cortado para extraer su masa interior antes de separar la pulpa. Este proceso se lorealiza de forma manual, donde los empleados se ayudan de cuchillos, para elcorte.

3.3.4.8. Pelado

A otras frutas hay necesidad de retirarles la cáscara como a la guanábana y papaya, por su incompatibilidad de color, textura o sabor al mezclarla con la pulpa. Esta operación se efectúa de manera manual.

El pelado manual se realiza con cuchillos comunes de cocina o con otros que presentan ciertas características que se ajustan al tipo de piel de algunas frutas. Estos son similares a los que hoy se emplean para pelar papas. Permiten cortar películas de cierto grosor, evita que el operario por descuido se corte, tienen formas especiales para acceder a superficies curvas y poseen empuñaduras ergonómicas, es decir que se ajustan muy bien a la mano del operario.

Los métodos físicos emplean calor y frío, por ejemplo el tomate de mesa.

3.3.4.9. Separación

Esta operación permite retirar la masa pulpa - semilla de frutas como el maracuyá.

Se efectúa generalmente de forma manual con la ayuda de cucharas de tamaños adecuados. El rendimiento aumenta si se hace dentro de recipientes plásticos para evitar las pérdidas de jugos.

Por eficiencia los operarios se colocan en grupos que se encargan unos de cortar la fruta y otros de separar la pulpa - semilla. Estas masas obtenidas se deben cubrir con tapas o materiales plásticos para prevenir contaminaciones u oxidaciones del medio ambiente.

3.3.4.10. Molido

Permite la desintegración de las estructuras de las frutas que facilitan operaciones como el escaldado y despulpado.

Se efectúa en molinos como el de martillos, con el que se logra un efecto similar al de la licuadora casera o industrial.

Este molido no es recomendado para frutas que poseen semillas grandes, oscuras, amargas y frágiles como el maracuyá, el mango o aún la guanábana. Las frutas de semillas pequeñas como la guayaba, mora, y tomate se desintegran muy bien sin romper las semillas.

El molido tiene la desventaja de incorporar aire a la masa obtenida, con lo que se pueden acelerar procesos de oxidación entre los que se hallan el cambio de color y formación de espuma, ambos causan inconvenientes en la calidad final de la pulpa.

3.3.4.11. Escaldado

Consiste en someter a la fruta a un calentamiento corto y posterior enfriamiento. Se realiza para ablandar un poco la materia prima y con esto aumentar el rendimiento de la pulpa; además se reduce la carga microbiana que aún permanece sobre elfruto y también se realiza para inactivar enzimas que producen cambios indeseables de apariencia, color, aroma, y sabor en la pulpa, aunque pueda estar conservada bajo congelación.

En la fábrica el escaldado se efectúa por inmersión de las frutas en una marmita con agua caliente, o por calentamiento con vapor vivo generado también en marmita. Esta operación se puede realizar a presión atmosférica o a sobrepresión en una autoclave. Con el escaldado en agua caliente se pueden perder jugos y componentes nutricionales. Bajo vapor puede ser más costoso y demorado pero hay menos pérdidas. En autoclave es más rápido pero costoso.

En todos los casos se producen algunos cambios. Baja significativamente la carga microbiana; el color se hace más vivo, el aroma y sabor puede variar a un ligero cocido y la viscosidad de la pulpa puede aumentar.

Un escaldado frecuente se hace en marmita agregando una mínima cantidad de agua, como para generar vapor y luego si se coloca la fruta. Se agita con vigor, tratando de desintegrar las frutas y volver el producto en una especie de "sopa". Cuando la mezcla alcanza cerca de 70 a 75º C se suspende el calentamiento.

3.3.4.12. Despulpado

Es la operación en la que se logra la separación de la pulpa de los demás residuos como las semillas, cáscaras y otros. El principio en que se basa es el de hacer pasar la pulpa - semilla a través de un tamiz. Esto se logra por el impulso que comunica a la masa pulpa - semilla, un conjunto de paletas (2 o 4) unidas a un eje que gira a velocidad fija o variable. La fuerza centrífuga de giro de las paletas lleva a la masa contra el tamiz y allí es arrastrada logrando que el fluido pase a través de los orificios del tamiz. Es el mismo efecto que se logra cuando se pasa por un colador una mezcla de pulpa - semilla que antes ha sido licuada. Aquí los tamices son el colador y las paletas es la cuchara que repasa la pulpa - semilla contra la malla del colador.

El proceso de despulpado se inicia introduciendo la fruta entera en la despulpadora perfectamente higienizada. Solo algunas frutas, como la mora, guayaba o fresa, permiten una introducciónd irecta a la despulpadora. Las demás exigen una adecuación como pelada (guanábana), corte y separación de la pulpa - semilla de la cáscara (maracuyá). Ablandamiento por escaldado (tomate de árbol).

Los residuos pueden salir impregnados aún en la pulpa, por lo que se acostumbra a repasar estos residuos. Los mismos se pueden mezclar con un poco de agua o de la misma pulpa que ya ha salido, para así incrementar el rendimiento de la pulpa. Esto se ve cuando el nuevo residuo sale más seco y se aumenta la cantidad de pulpa.

Se recomienda exponer lo menos posible la pulpa al medio ambiente. Para lograrlo inmediatamente que se obtiene la pulpa, se cubre, o se la envía por tubería desde la salida de la despulpadora hasta un tanque de almacenamiento.

3.3.4.13. Refinado

Consiste en reducir el tamaño de partícula de la pulpa, cuando ésta ha sido obtenida antes por el uso de una malla de mayor diámetro de sus orificios.

Reducir el tamaño de partícula da una mejor apariencia a la pulpa, evita una más rápida separación de los sólidos insolubles en suspensión, le comunica una textura más fina a los productos como mermelada o bocadillos preparados a partir de esta pulpa. De otra parte refinar baja los rendimientos en pulpa por la separación de material grueso y duro que está naturalmente presente en la pulpa inicial.

El refinado se puede hacer en la misma despulpadora, solo que se le cambia la malla por otra de diámetro de orificio más fino. Generalmente la primera pasada para el despulpado se realiza con malla 0,060" y el refinado con 0,045" o menor. La malla inicial depende del diámetro de la semilla y la malla final de la calidad de finura que se desee tenga la pulpa.

3.3.4.14. Empaque

Las pulpas ya obtenidas deben ser aisladas del medio ambiente a fin de mantener sus características hasta el momento de su empleo. Esto se logra mediante su empacado con el mínimo de aire, en recipientes adecuados y compatibles con las pulpas.

Debido a la tendencia que tiene el vapor de agua de sublimarse de las superficies de los alimentos congelados a las superficies más frías de

los congeladores y cámaras frigoríficas, los materiales de envasado empleados deben tener un alto grado de impermeabilidad al vapor de agua.

La mayoría de los alimentos se dilatan al congelarse, algunos de ellos hasta un 10% de su volumen. Por lo tanto los envases en que se congelan deben ser fuertes, hasta cierto punto flexible y no llenarse completamente. Como en el caso de todos los alimentos que pueden almacenarse durante meses, sus envases deben protegerse contra la luz y el aire. Ya que generalmente se les descongela dentro de sus envases, estos deben ser impermeables a fin de prevenir el escurrimiento durante la descongelación.

3.3.4.15. Pasteurización

Consiste en calentar un producto a temperaturas que provoquen la destrucción de los microorganismos patógenos. El calentamiento va seguido de un enfriamiento para evitar la sobrecocción y la supervivencia de los microorganismos termófilos.

La temperatura y el tiempo escogidos para pasterizar una pulpa dependerán de varios factores como su pH, composición, viscosidad y nivel de contaminación inicial. A menor pH, viscosidad y contaminación, se requerirá menor tiempo o temperatura de pasterización para disminuir el grado de contaminación hasta niveles en los que no se presentará un rápido deterioro de la pulpa.

Es el caso de la pulpa de maracuyá que posee un pH alrededor de 2.7, que no permite el crecimiento de muchos microorganismos y el calor a este pH los afecta más; baja viscosidad que permite un mayor

movimiento de la pulpa y por ello mejor y más rápida transmisión del calor, y por estar protegido por una cáscara tan resistente no se contamina fácilmente, este nivel de contaminación dependerá en gran medida en la higiene y cuidados mantenidos durante el procesamiento.

La esterilización es simplemente una pasterización más drástica que elimina mayor número de microorganismos. Se logra empleando equipos más complejos como una autoclave, en donde por la sobrepresión que se alcanza, la temperatura puede ascender a niveles superiores a los de ebullición del agua a condiciones de medio ambiente.

En el caso de las pulpas casi no se emplea esterilizarlas debido al bajo pH que caracteriza a la mayoría de las frutas.

3.3.4.16. Congelación

Se basa en el principio de que a menor temperatura más lentas son todas las reacciones. Esto incluye las reacciones producidas por los microorganismos, los cuales no son destruidos sino retardada su actividad vital.

La congelación disminuye la disponibilidad del agua debido a la solidificación del agua que caracteriza este estado de la materia. Al no estar disponible como medio líquido, muy pocas reacciones pueden ocurrir. Solo algunas como la desnaturalización de proteínas presentes en la pared celular.

Esto propicia la precipitación de los sólidos insolubles con lo que se favorece el cambio en la textura y la separación de fases, sobre todo cuando con estas pulpas se preparan néctares.

Durante la congelación se favorece la formación de cristales de hielo que crecen y causan roturas de las paredes celulares y pérdida de la capacidad retenedora de los jugos dentro de las células. Se ha notado también que la congelación produce una disminución de los aromas y sabores propios de las frutas.

A pesar de estos cambios, la congelación es la técnica más sencilla que permite mantener las características sensoriales y nutricionales lo más parecidas a las de las pulpas frescas y en el caso particular de la Empresa Industrial Procesadora y Comercializadora de Frutas será la técnica empleada.

Para el control microbiológico de calidad hay necesidad de descongelar la pulpa, con lo que se puede aumentar el recuento real del producto.

La conservación por congelación permite mantener las pulpas por períodos cercanos a un año sin que se deteriore significativamente. Entre más tiempo y más baja sea la temperatura de almacenamiento congelado, mayor número de microorganismos que perecerán. A la vez que las propiedades sensoriales de las pulpas congeladas durante demasiado tiempo irán cambiando.

3.4. MODELO NAVEGACIONAL

Consiste en definir la relación que existirá entre las distintas pantallas y los diversos contenidos del sitio WAP, perteneciente al módulo de Ventas.

En este sub-modelo del área de ventas, se podrá consultar el stock de las pulpas de fruta específicamente, es decir de aquellos artículos definidos como

de venta, y el valor que se mostrará es el stock disponible, siempre que sea mayor a cero.

También contendrá la toma de pedido de un cliente, es decir con campos específicos como el RUC o Cédula, artículo y cantidad, se creará una orden de venta, la cual será procesada de inmediato para que lo pedido por el cliente se convierta en stock bajo pedido.

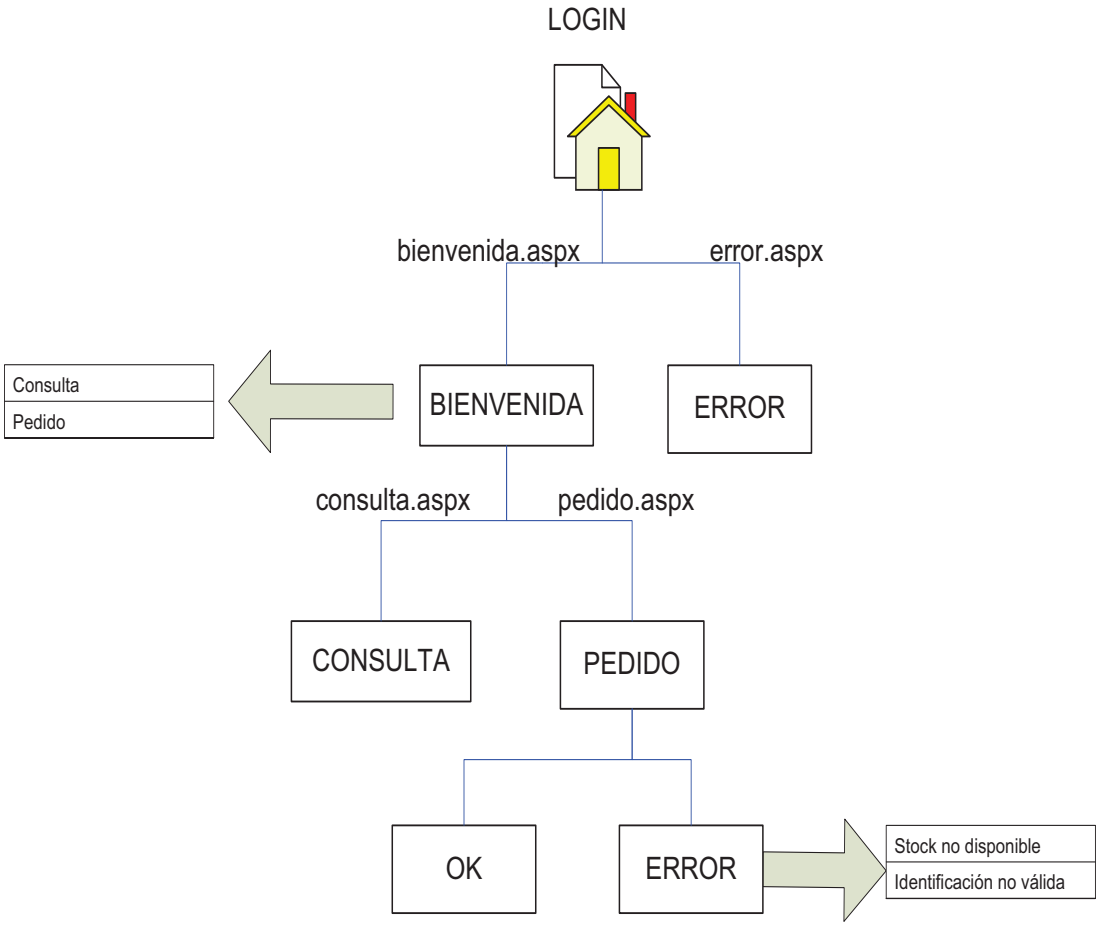


Fig. 3.7Diagrama Navegacional

Fuente: Tesistas

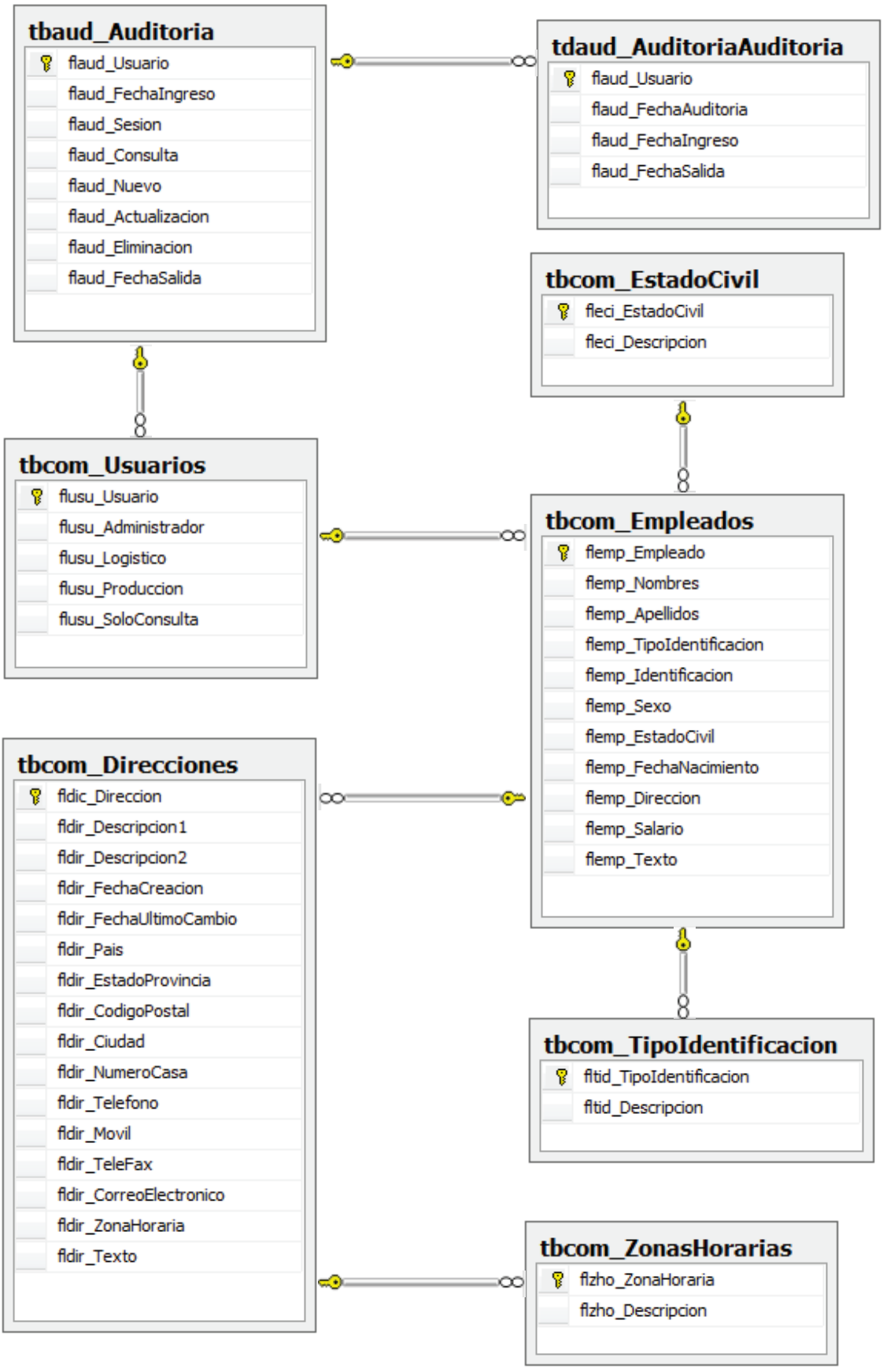
3.5. ESTRUCTURA ENTIDAD RELACIÓN

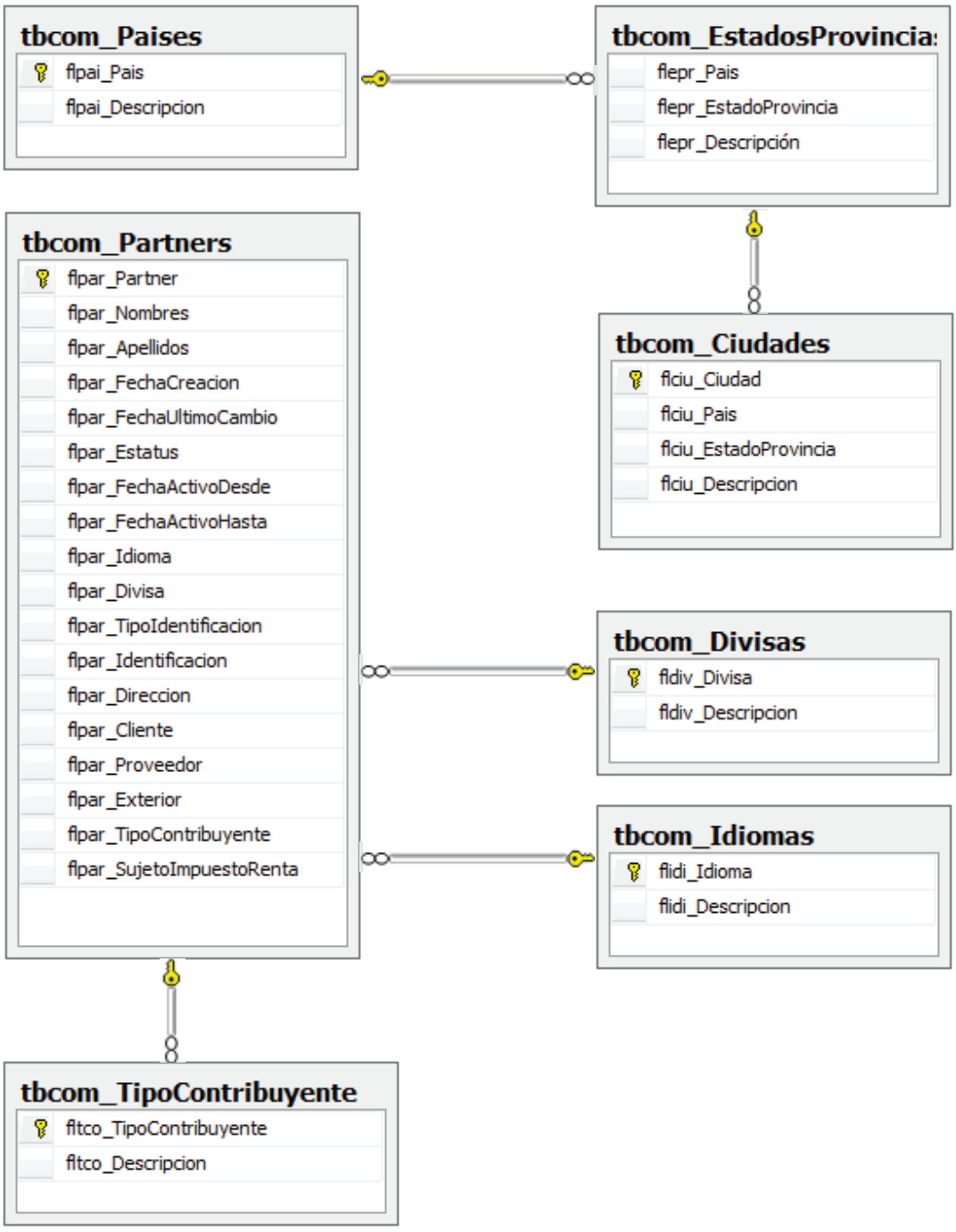
Para la creación de la base de datos y utilización de la misma, se ha definido la siguiente nomenclatura:

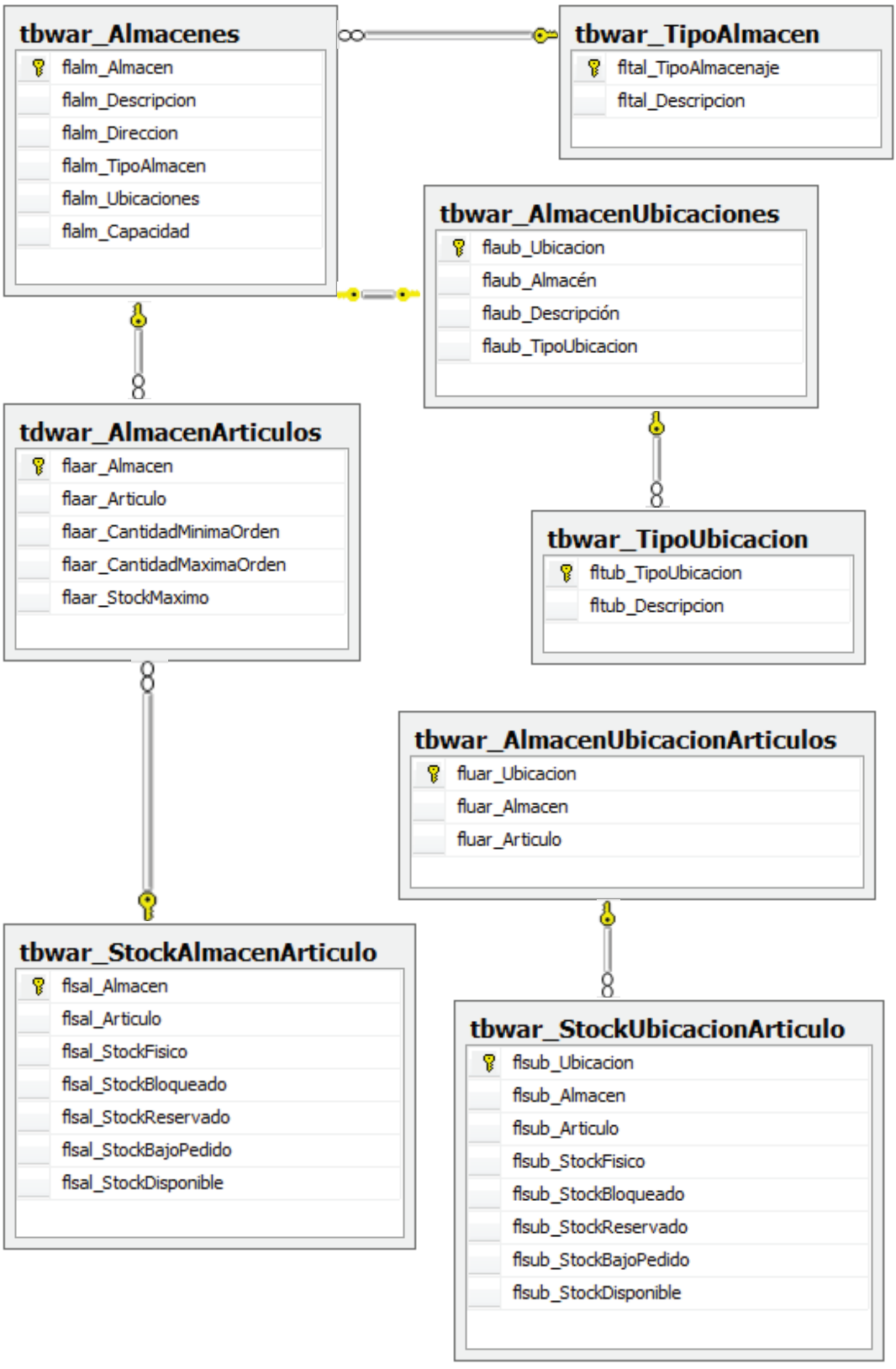
a. Nombre de la Base de Datos: db_LaPulpita, donde db significa Data Base, y subguión para separar el nombre de la Empresa.

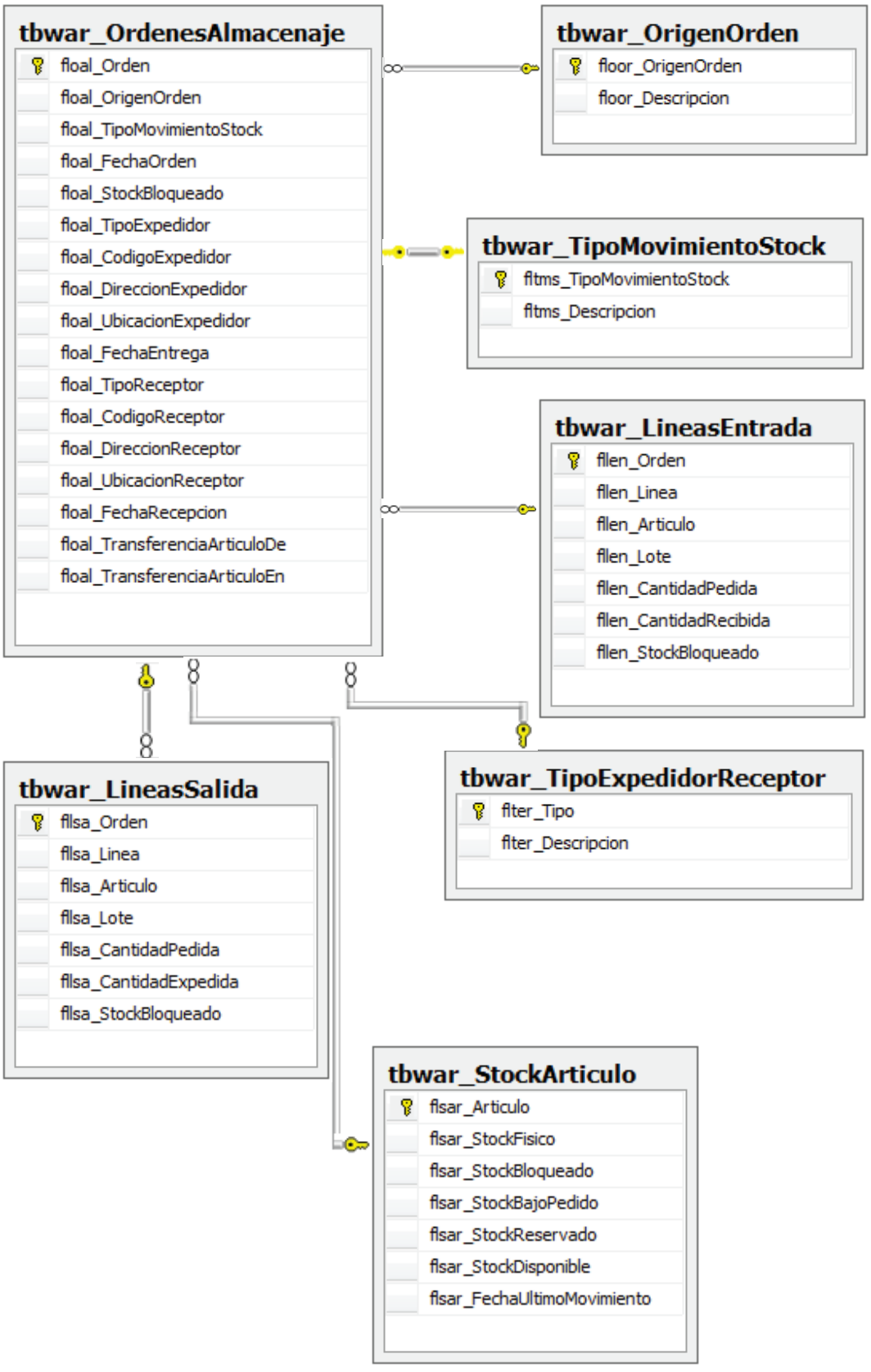
b. Nombre de las Tablas: tbsal_OrdenVenta, donde tb significa Table, seguido de las tres primeras letras del módulo en ingles sal (Sales - Ventas), subguión y el nombre de la tabla.

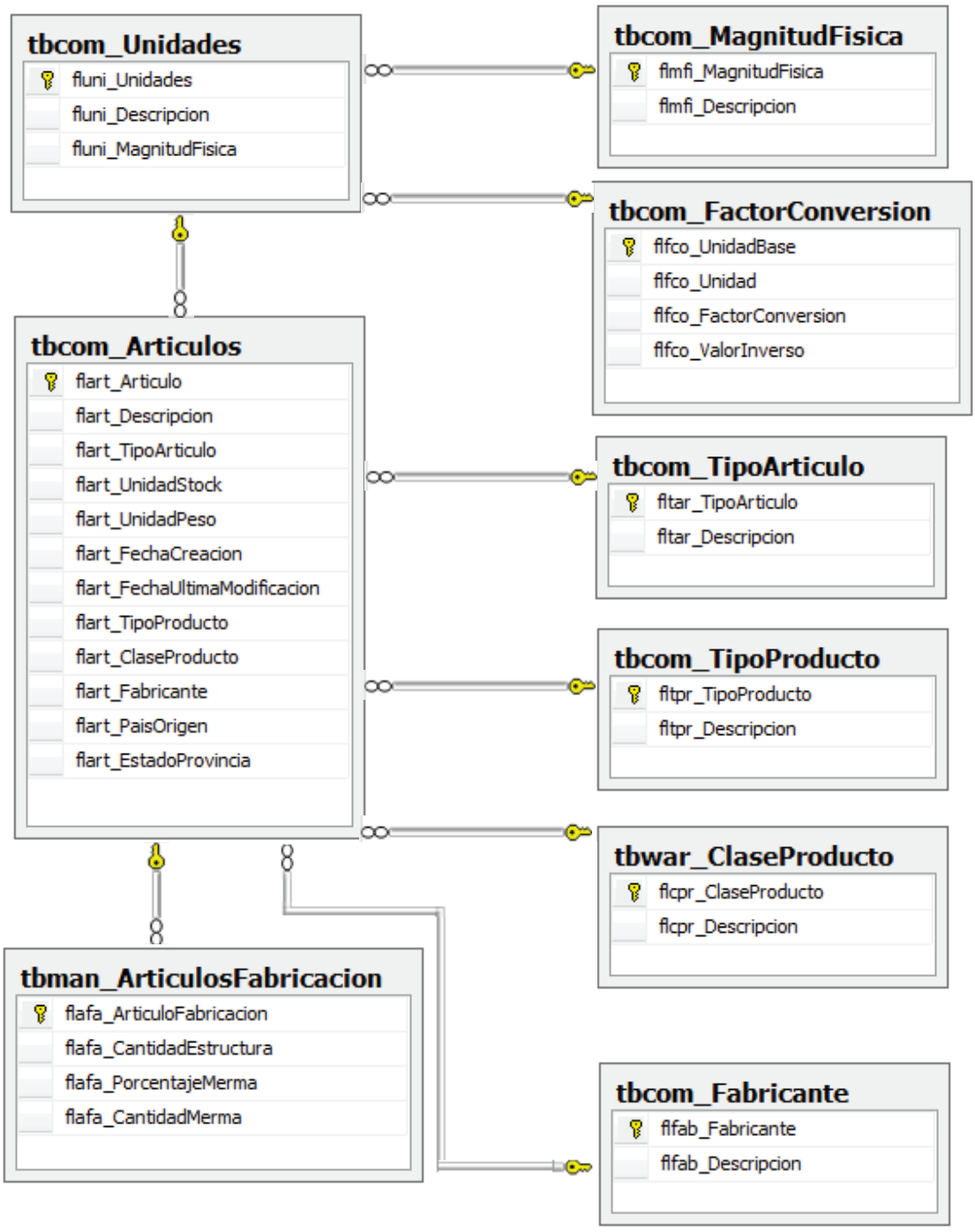
c. Nombre de los Campos: flors_NumeroOrden, donde fl significa File, seguido de las tres primeras letras del nombre de la tabla en inglés (Order Sale – Orden de Venta), subguión y el nombre del campo.

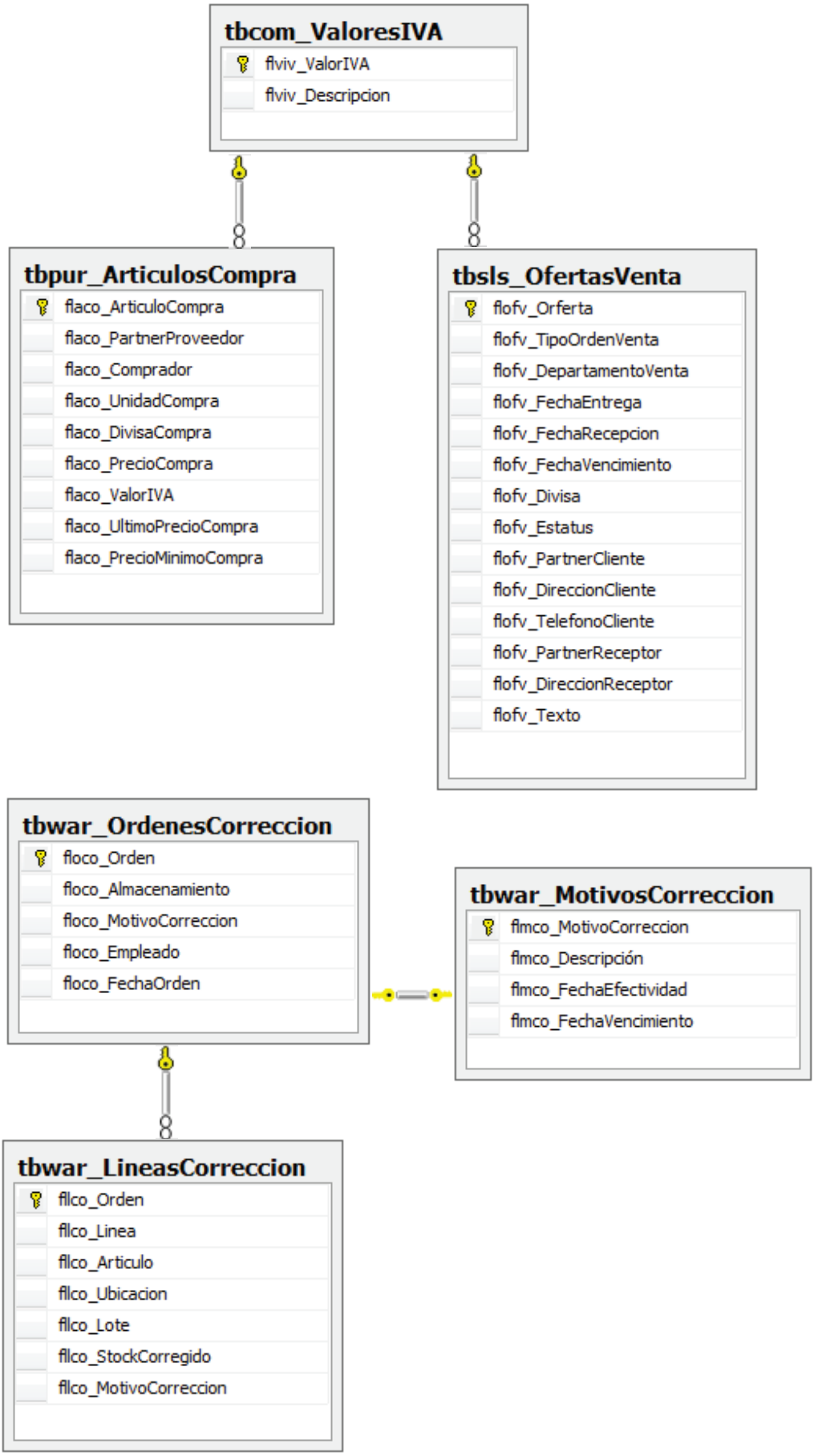

tbaud_Auditoria
flaud_Usuario
flaud_FechaIngreso
flaud_Sesion
flaud_Consulta
flaud_Nuevo
flaud_Actualizacion
flaud_Eliminacion
flaud_FechaSalida
tdaud_AuditoriaAuditoria
flaud_Usuario
flaud_FechaAuditoria
flaud_FechaIngreso
flaud_FechaSalida
tbcom_EstadoCivil
fleci_EstadoCivil
fleci_Descripcion
tbcom_Usuarios
flusu_Usuario
flusu_Administrador
flusu_Logistico
flusu_Produccion
flusu_SoloConsulta
tbcom_Empleados
flemp_Empleado
flemp_Nombres
flemp_Apellidos
flemp_TipoIdentificacion
flemp_Identificacion
flemp_Sexo
flemp_EstadoCivil
flemp_FechaNacimiento
flemp_Direccion
flemp_Salario
flemp_Texto
tbcom_Direcciones
fldic_Direccion
fldir_Descripcion1
fldir_Descripcion2
fldir_FechaCreacion
fldir_FechaUltimoCambio
fldir_Pais
fldir_EstadoProvincia
fldir_CodigoPostal
fldir_Ciudad
fldir_NumeroCasa
fldir_Telefono
fldir_Movil
fldir_TeleFax
fldir_CorreoElectronico
fldir_ZonaHoraria
fldir_Texto
tbcom_TipoIdentificacion
fltid_TipoIdentificacion
fltid_Descripcion
tbcom_ZonasHorarias
flzho_ZonaHoraria
flzho_Descripcion

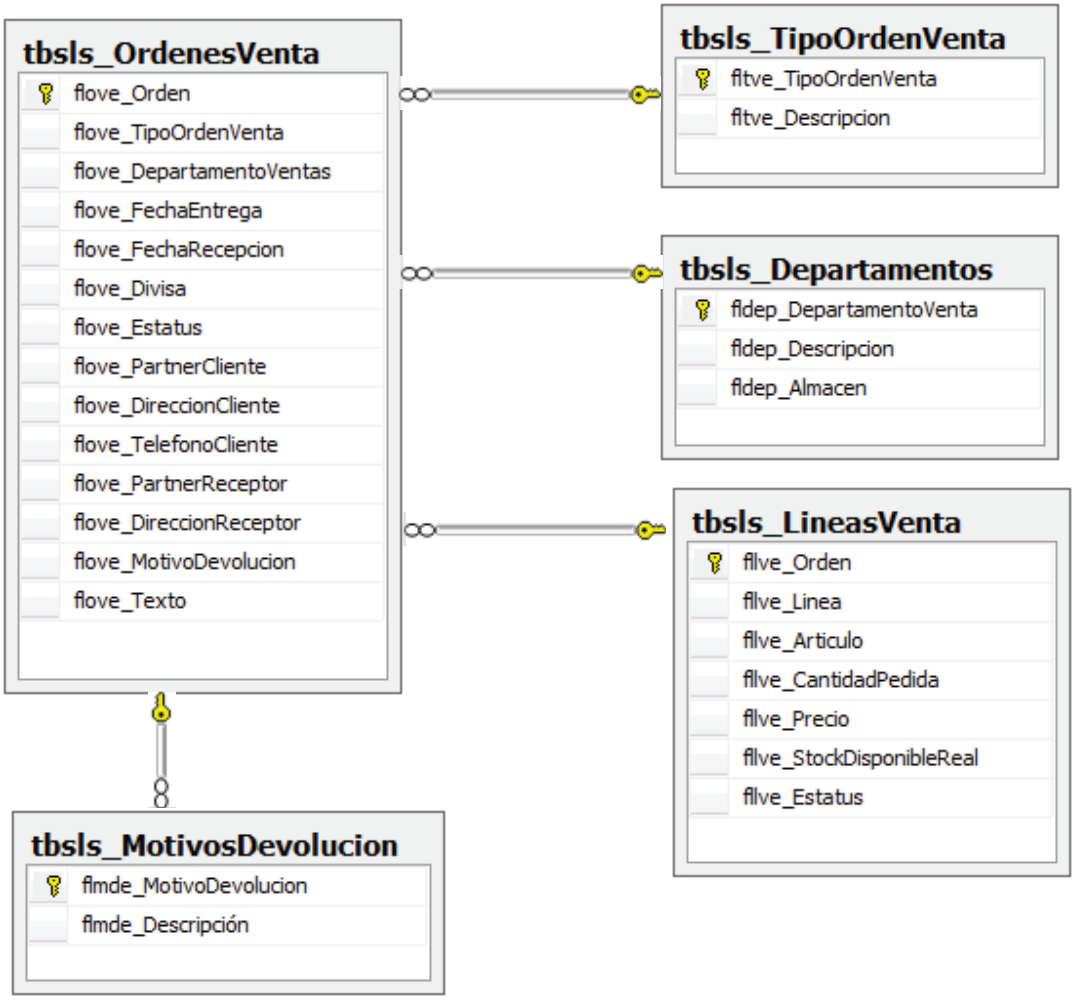

tbcom_Paises
flpai_Pais
flpai_Descripcion
tbcom_EstadosProvincia
flepr_Pais
flepr_EstadoProvincia
flepr_Descripción
tbcom_Partners
flpar_Partner
flpar_Nombres
flpar_Apellidos
flpar_FechaCreacion
flpar_FechaUltimoCambio
flpar_Estatus
flpar_FechaActivoDesde
flpar_FechaActivoHasta
flpar_Idioma
flpar_Divisa
flpar_TipoIdentificacion
flpar_Identificacion
flpar_Direccion
flpar_Cliente
flpar_Proveedor
flpar_Exterior
flpar_TipoContribuyente
flpar_SujetoImpuestoRenta
tbcom_Ciudades
flciu_Ciudad
flciu_Pais
flciu_EstadoProvincia
flciu_Descripcion
tbcom_Divisas
fldiv_Divisa
fldiv_Descripcion
tbcom_Idiomas
flidi_Idioma
flidi_Descripcion
tbcom_TipoContribuyente
fltco_TipoContribuyente
fltco_Descripcion

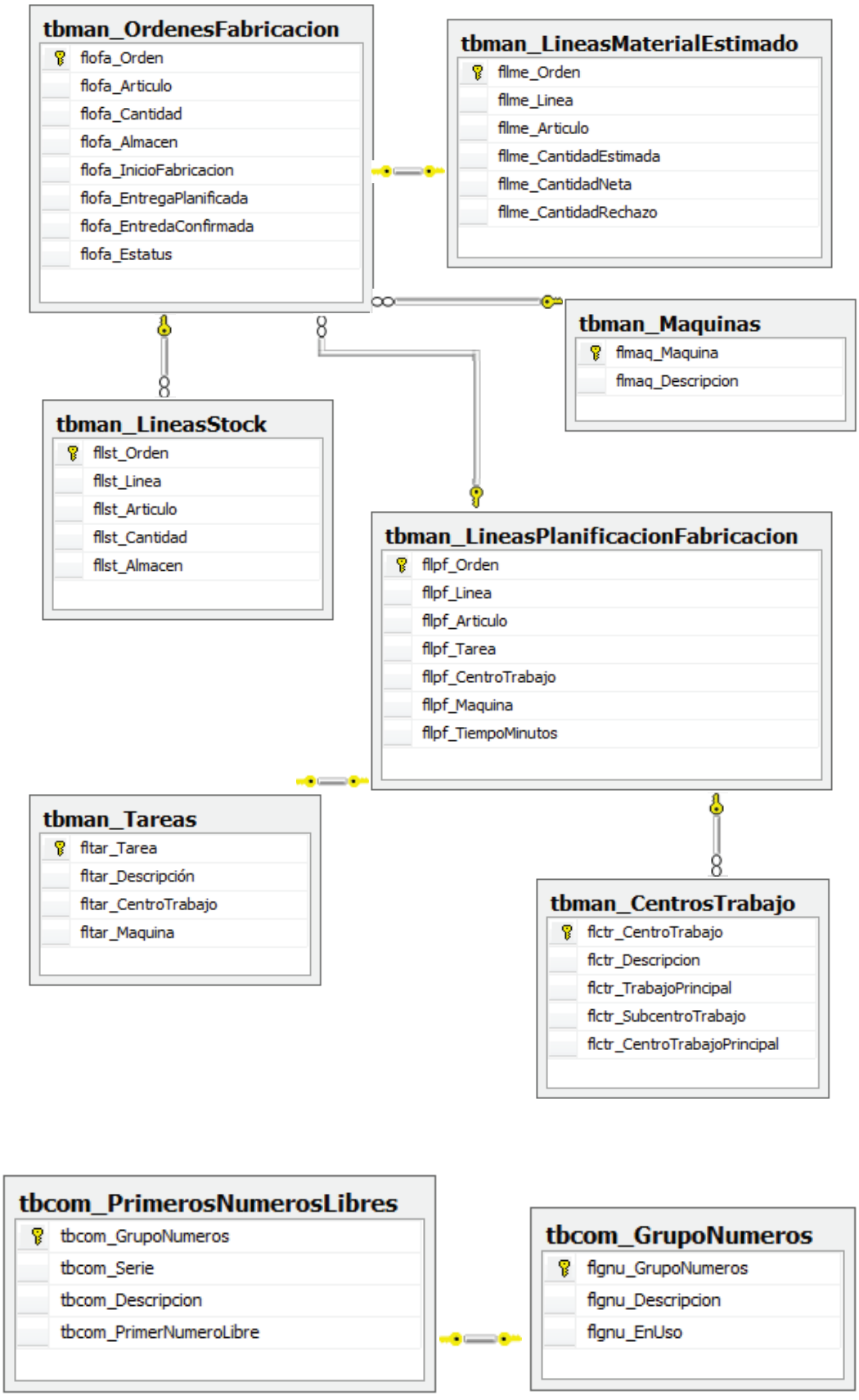

tbwar_Almacenes
flalm_Almacen
flalm_Descripcion
flalm_Direccion
flalm_TipoAlmacen
flalm_Ubicaciones
flalm_Capacidad
tbwar_TipoAlmacen
fltal_TipoAlmacenaje
fltal_Descripcion
tbwar_AlmacenUbicaciones
flaub_Ubicacion
flaub_Almacén
flaub_Descripción
flaub_TipoUbicacion
tdwar_AlmacenArticulos
flaar_Almacen
flaar_Articulo
flaar_CantidadMinimaOrden
flaar_CantidadMaximaOrden
flaar_StockMaximo
tbwar_TipoUbicacion
fltub_TipoUbicacion
fltub_Descripcion
tbwar_AlmacenUbicacionArticulos
fluar_Ubicacion
fluar_Almacen
fluar_Articulo
tbwar_StockAlmacenArticulo
flsal_Almacen
flsal_Articulo
flsal_StockFisico
flsal_StockBloqueado
flsal_StockReservado
flsal_StockBajoPedido
flsal_StockDisponible
tbwar_StockUbicacionArticulo
flsub_Ubicacion
flsub_Almacen
flsub_Articulo
flsub_StockFisico
flsub_StockBloqueado
flsub_StockReservado
flsub_StockBajoPedido
flsub_StockDisponible

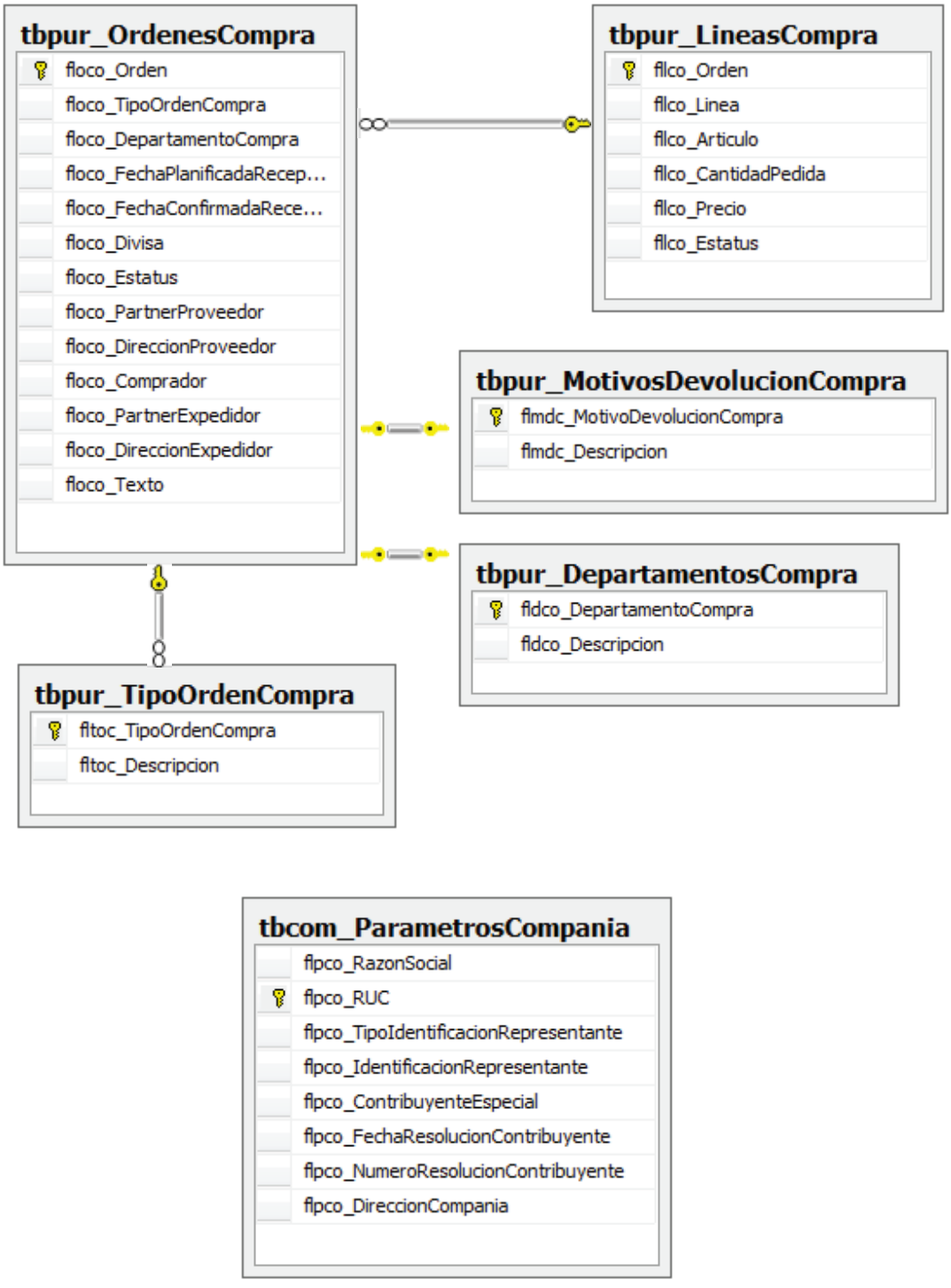

tbwar_OrdenesAlmacenaje
floal_Orden
floal_OrigenOrden
floal_TipoMovimientoStock
floal_FechaOrden
floal_StockBloqueado
floal_TipoExpedidor
floal_CodigoExpedidor
floal_DireccionExpedidor
floal_UbicacionExpedidor
floal_FechaEntrega
floal_TipoReceptor
floal_CodigoReceptor
floal_DireccionReceptor
floal_UbicacionReceptor
floal_FechaRecepcion
floal_TransferenciaArticuloDe
floal_TransferenciaArticuloEn
tbwar_OrigenOrden
floor_OrigenOrden
floor_Descripcion
tbwar_TipoMovimientoStock
fltms_TipoMovimientoStock
fltms_Descripcion
tbwar_LineasEntrada
fllen_Orden
fllen_Linea
fllen_Articulo
fllen_Lote
fllen_CantidadPedida
fllen_CantidadRecibida
fllen_StockBloqueado
tbwar_TipoExpedidorReceptor
flter_Tipo
flter_Descripcion
tbwar_LineasSalida
fllsa_Orden
fllsa_Linea
fllsa_Articulo
fllsa_Lote
fllsa_CantidadPedida
fllsa_CantidadExpedida
fllsa_StockBloqueado
tbwar_StockArticulo
flsar_Articulo
flsar_StockFisico
flsar_StockBloqueado
flsar_StockBajoPedido
flsar_StockReservado
flsar_StockDisponible
flsar_FechaUltimoMovimiento


tbcom_Unidades
fluni_Unidades
fluni_Descripcion
fluni_MagnitudFisica
tbcom_MagnitudFisica
flmfi_MagnitudFisica
flmfi_Descripcion
tbcom_FactorConversion
flfco_UnidadBase
flfco_Unidad
flfco_FactorConversion
flfco_ValorInverso
tbcom_Articulos
flart_Articulo
flart_Descripcion
flart_TipoArticulo
flart_UnidadStock
flart_UnidadPeso
flart_FechaCreacion
flart_FechaUltimaModificacion
flart_TipoProducto
flart_ClaseProducto
flart_Fabricante
flart_PaisOrigen
flart_EstadoProvincia
tbcom_TipoArticulo
fltar_TipoArticulo
fltar_Descripcion
tbcom_TipoProducto
fltpr_TipoProducto
fltpr_Descripcion
tbwar_ClaseProducto
flcpr_ClaseProducto
flcpr_Descripcion
tbman_ArticulosFabricacion
flafa_ArticuloFabricacion
flafa_CantidadEstructura
flafa_PorcentajeMerma
flafa_CantidadMerma
tbcom_Fabricante
flfab_Fabricante
flfab_Descripcion


tbcom_ValoresIVA
flviv_ValorIVA
flviv_Descripcion
tbpur_ArticulosCompra
flaco_ArticuloCompra
flaco_PartnerProveedor
flaco_Comprador
flaco_UnidadCompra
flaco_DivisaCompra
flaco_PrecioCompra
flaco_ValorIVA
flaco_UltimoPrecioCompra
flaco_PrecioMinimoCompra
tbsls_OfertasVenta
flofv_Orferta
flofv_TipoOrdenVenta
flofv_DepartamentoVenta
flofv_FechaEntrega
flofv_FechaRecepcion
flofv_FechaVencimiento
flofv_Divisa
flofv_Estatus
flofv_PartnerCliente
flofv_DireccionCliente
flofv_TelefonoCliente
flofv_PartnerReceptor
flofv_DireccionReceptor
flofv_Texto
tbwar_OrdenesCorreccion
floco_Orden
floco_Almacenamiento
floco_MotivoCorreccion
floco_Empleado
floco_FechaOrden
tbwar_MotivosCorreccion
flmco_MotivoCorreccion
flmco_Descripción
flmco_FechaEfectividad
flmco_FechaVencimiento
tbwar_LineasCorreccion
fllco_Orden
fllco_Linea
fllco_Articulo
fllco_Ubicacion
fllco_Lote
fllco_StockCorregido
fllco_MotivoCorreccion


tbsls_OrdenesVenta
flove_Orden
flove_TipoOrdenVenta
flove_DepartamentoVentas
flove_FechaEntrega
flove_FechaRecepcion
flove_Divisa
flove_Estatus
flove_PartnerCliente
flove_DireccionCliente
flove_TelefonoCliente
flove_PartnerReceptor
flove_DireccionReceptor
flove_MotivoDevolucion
flove_Texto
tbsls_TipoOrdenVenta
fltve_TipoOrdenVenta
fltve_Descripcion
tbsls_Departamentos
fldep_DepartamentoVenta
fldep_Descripcion
fldep_Almacen
tbsls_LineasVenta
fllve_Orden
fllve_Linea
fllve_Articulo
fllve_CantidadPedida
fllve_Precio
fllve_StockDisponibleReal
fllve_Estatus
tbsls_MotivosDevolucion
flmde_MotivoDevolucion
flmde_Descripción


tbman_OrdenesFabricacion
flofa_Orden
flofa_Articulo
flofa_Cantidad
flofa_Almacen
flofa_InicioFabricacion
flofa_EntregaPlanificada
flofa_EntredaConfirmada
flofa_Estatus
tbman_LineasMaterialEstimado
flme_Orden
flme_Linea
flme_Articulo
flme_CantidadEstimada
flme_CantidadNeta
flme_CantidadRechazo
tbman_Maquinas
flmaq_Maquina
flmaq_Descripcion
tbman_LineasStock
flst_Orden
flst_Linea
flst_Articulo
flst_Cantidad
flst_Almacen
tbman_LineasPlanificacionFabricacion
flpf_Orden
flpf_Linea
flpf_Articulo
flpf_Tarea
flpf_CentroTrabajo
flpf_Maquina
flpf_TiempoMinutos
tbman_Tareas
fltar_Tarea
fltar_Descripción
fltar_CentroTrabajo
fltar_Maquina
tbman_CentrosTrabajo
flctr_CentroTrabajo
flctr_Descripcion
flctr_TrabajoPrincipal
flctr_SubcentroTrabajo
flctr_CentroTrabajoPrincipal
tbcom_PrimerosNumerosLibres
tbcom_GrupoNumeros
tbcom_Serie
tbcom_Descripcion
tbcom_PrimerNumeroLibre
tbcom_GrupoNumeros
flgnu_GrupoNumeros
flgnu_Descripcion
flgnu_EnUso


tbpur_OrdenesCompra
floco_Orden
floco_TipoOrdenCompra
floco_DepartamentoCompra
floco_FechaPlanificadaRecep...
floco_FechaConfirmadaRece...
floco_Divisa
floco_Estatus
floco_PartnerProveedor
floco_DireccionProveedor
floco_Comprador
floco_PartnerExpedidor
floco_DireccionExpedidor
floco_Texto
tbpur_LineasCompra
fllco_Orden
fllco_Linea
fllco_Articulo
fllco_CantidadPedida
fllco_Precio
fllco_Estatus
tbpur_MotivosDevolucionCompra
flmdc_MotivoDevolucionCompra
flmdc_Descripcion
tbpur_DepartamentosCompra
fldco_DepartamentoCompra
fldco_Descripcion
tbpur_TipoOrdenCompra
fltoc_TipoOrdenCompra
fltoc_Descripcion

tbcom_ParametrosCompania
flpco_RazonSocial
flpco_RUC
flpco_TipoIdentificacionRepresentante
flpco_IdentificacionRepresentante
flpco_ContribuyenteEspecial
flpco_FechaResolucionContribuyente
flpco_NumeroResolucionContribuyente
flpco_DireccionCompania

CAPÍTULO IV
IMPLEMENTACIÓN

4.1. INTRODUCCIÓN

Se describe de forma general aquellos procesos que fueron utilizados en la base de datos para ser llamados desde el sistema con la finalidad de agilizar las consultas, ingresos, actualizaciones y eliminación de datos.

La programación desarrollada con Visual C# .Net para crear este sistema, contiene varias funciones, clases y llamadas a librerías estándar, con las cuales una vez integradas permiten que este software se convierta en un producto de calidad.

Se realizarán pruebas de software para verificar que éste no tiene errores en el momento de la ejecución y al ser manejado por el usuario final.

4.2. PROCESOS ALMACENADOS

Los procesos almacenados son módulos o rutinas que encapsulan código para su reutilización. Un procedimiento almacenado puede incluir parámetros de entrada, devolver resultados tabulares o escalares o mensajes para el cliente, invocar instrucciones de lenguaje de definición de datos (DDL) e instrucciones de lenguaje de manipulación de datos (DML), así como devolver parámetros de salida.[1]

Hay dos tipos de procesos almacenados en SQL Server 2008: Transact-SQL, y CLR. Con el proceso CLR (Common Language Runtime) de Microsoft.Net, el cual es utilizado para el desarrollo del sistema, se puede aceptar y devolver parámetros

[1] http://msdn.microsoft.com/es-es/library/ms187644.aspx

suministrados por el usuario. Son implementados como métodos públicos y estáticos en una clase de un ensamblado de .Net Framework.

Se describe de manera general los procesos almacenados que se crearon y están siendo utilizados para el sistema:

4.2.1. CONSULTAS

Las consultas fueron desarrolladas, con el fin de cargar los datos resultantes en las grillas de las sesiones de consulta, o en los campos de sesiones de nuevo, actualización, y copiado.

a) Este proceso es utilizado para seleccionar todos los datos de las tablas, el cual carga las sesiones que contienen la lista de datos:

```
ALTERPROCEDURE[dbo].[sp_Usuarios]
   -- Add the parameters for the stored procedure here
AS
BEGIN
   -- SET NOCOUNT ON added to prevent extra result sets from
   -- interfering with SELECT statements.
   SETNOCOUNTON;

-- Insert statements for procedure here
   SELECT      *
   FROM  tbcom_Usuarios
END
```

b) Conel siguiente proceso almacenado se selecciona todos los datos de la tabla, incluyendo aquellas descripciones de otras tablas, y se la carga de igual manera en las sesiones que contienen listados:

```
ALTERPROCEDURE[dbo].[sp_Ciudades]
   -- Add the parameters for the stored procedure here
AS
BEGIN
   -- SET NOCOUNT ON added to prevent extra result sets from
   -- interfering with SELECT statements.
   SETNOCOUNTON;
```

```
-- Insert statements for procedure here
   SELECT
   dbo.tbcom_Ciudades.*,dbo.tbcom_Paises.flpai_Descripcion,
               dbo.tbcom_EstadosProvincias.flepr_Descripcion
   FROM
   dbo.tbcom_Ciudades,dbo.tbcom_EstadosProvincias,dbo.tbcom_Paises
   WHERE flpai_Pais=flciu_Pais
   AND
   (flepr_Pais=flciu_PaisANDflepr_EstadoProvincia=flciu_EstadoProv
incia)
END
```

c) Para recuperar datos por medio del índice de las tablas, el cual es utilizado para las sesiones que hacen el trabajo de actualizar registros, se tiene el siguiente código:

```
ALTERPROCEDURE[dbo].[sp_SelectPais]
   -- Add the parameters for the stored procedure here
   @p_paisnchar(3)=''      -- Parámetro el índice de la tabla
AS
BEGIN
   -- SET NOCOUNT ON added to prevent extra result sets from
   -- interfering with SELECT statements.
   SETNOCOUNTON;

-- Insert statements for procedure here
   SELECT      flpai_Pais
   FROM  dbo.tbcom_Paises
   WHERE flpai_Pais=@p_pais
END
```

d) Para cargar las sesiones que cumplen la función de zoom, con todos los datos requeridos, sean estos filtrados o no, se lo hace de la siguiente manera:

```
ALTERPROCEDURE[dbo].[sp_SelectEstadoP]
   -- Add the parameters for the stored procedure here
   @p_paisnchar(3)=''            -- Parámetro para filtro
AS
BEGIN
   -- SET NOCOUNT ON added to prevent extra result sets from
   -- interfering with SELECT statements.
   SETNOCOUNTON;

-- Insert statements for procedure here
```

```
   SELECT
dbo.tbcom_EstadosProvincias.*,dbo.tbcom_Paises.flpai_Descripcion
   FROM  dbo.tbcom_EstadosProvincias,dbo.tbcom_Paises
   WHERE flepr_Pais=@p_pais
   AND   flpai_Pais=flepr_Pais
END
```

e) El proceso almacenado descrito a continuaciónes utilizado para obtener la fecha y hora actual de la base de datos en el servidor:

```
ALTERPROCEDURE[dbo].[sp_DateHours]
   -- Add the parameters for the stored procedure here
AS
BEGIN
   -- SET NOCOUNT ON added to prevent extra result sets from
   -- interfering with SELECT statements.
   SETNOCOUNTON;

-- Insert statements for procedure here
   SELECTGETDATE();
END
```

4.2.2. INSERCIÓN, ACTUALIZACIÓN

Los procesos almacenados para la inserción de registros nuevos, o actualizar datos ya existentes, fueron desarrollados pensando en evitar duplicidad de datos, y dar a este proceso una mayor rapidez en tiempo de ejecución.

a) Se guardan los registros en la tabla, recibiendo como parámetros los campos de la misma, es utilizada en las sesiones de guardar, copiar y actualizar registros:

```
ALTERPROCEDURE[dbo].[sp_InsertUsuarios]
   -- Add the parameters for the stored procedure here
   @p_usuanchar(10)='',    -- Usuario
   @p_passnchar(10)='',    -- Password
   @p_nombnchar(30)='',    -- Nombre
   @p_admibit= 0,          -- Administrador
   @p_logibit= 0,          -- Logistico
   @p_prodbit= 0,          -- Producción
```

```
    @p_dagebit= 0,          -- Datos Generales
    @p_socobit= 0           -- Solo Consulta

AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.
    SETNOCOUNTON;

-- Insert statements for procedure here
    MERGE dbo.tbcom_Usuarios
    USING (SELECT@p_usuaASID)
    ASSRCONSRC.ID=dbo.tbcom_Usuarios.flusu_Usuario
    WHENMATCHEDTHENUPDATESETflusu_Password=@p_pass,
                                flusu_Nombre=@p_nomb,
                                flusu_Administrador=@p_admi,
                                flusu_Logistico=@p_logi,
                                flusu_Produccion=@p_prod,
                                flusu_DatosGenerales=@p_dage,
                                flusu_SoloConsulta=@p_soco
    WHENNOTMATCHEDTHENINSERT (flusu_Usuario,flusu_Password,
                                flusu_Nombre,flusu_Administrador,
                                flusu_Logistico,flusu_Produccion,
                    flusu_DatosGenerales,flusu_SoloConsulta)
                                VALUES (@p_usua,@p_pass,@p_nomb,
                                      @p_admi,@p_logi,@p_prod,
                                            @p_dage,@p_soco)
    OUTPUT$action,inserted.*;
END
```

Se ha utilizado el comando MERGE, para realizar operaciones como: INSERT, UPDATE y DELETE, en una sola sentencia. Este comando es similar alUPSERT (fusión de UPDATE e INSERT), de Oracle, que inserta filas que no existen y actualiza filas ya existentes. El objetivo de manejar con mayor eficacia los escenarios más comunes de almacenamiento de datos, con la previa comprobación de si la fila existe, y luego ejecutar una inserción, actualización o eliminación.

4.2.3. ELIMINACIÓN

a) Para realizar la eliminación de registros, se utilizó el siguiente proceso, que permite borrar sólo datos existentes, sin devolver error al no encontrarlos:

```
ALTERPROCEDURE[dbo].[sp_DeleteUsuarios]
   -- Add the parameters for the stored procedure here
   @p_usuanchar(10)=''           -- Índice de la tabla
AS
BEGIN
   -- SET NOCOUNT ON added to prevent extra result sets from
   -- interfering with SELECT statements.
   SETNOCOUNTON;

-- Insert statements for procedure here
   MERGE dbo.tbcom_Usuarios
   USING (SELECT@p_usuaASID)
   ASSRCONSRC.ID=dbo.tbcom_Usuarios.flusu_Usuario
   WHENMATCHEDTHENDELETE;
END
```

4.3. CÓDIGO FUENTE DEL SISTEMA

Visual C# 4.0 Lenguaje y Compilador, proporciona soporte para las bibliotecascompartidas, mediante la introducción de un tipo diferente de runtime. Esto permite que muchos escenarios nuevos, incluyendo el acceso simplificado a la API COM, a la Oficina de Autorización API, a las API dinámicas como IronPython bibliotecas, y el modelo de objetos HTML de documento (DOM).

Se describe a continuación el código utilizado para el funcionamiento del sistema, en el lenguaje C#.

4.3.1. CLASES

a) **Conexion.cs**, clase para conectarse con la base de datos y con la ejecución de todos los procesos almacenados:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Data;
using System.Data.OleDb;
using System.Data.Sql;
```

```
using System.Data.SqlClient;

namespace Conectar_DB
{
classConexion
    {
// Objeto de conexión con la dirección de la base de datos
staticSqlConnection g_conn = newSqlConnection("Data Source=SANDINO-PC;Initial
Catalog=Datos_LaPulpita;Integrated Security=True");

publicstaticbool abrir_database(refstring p_mess)
// Mensaje de error devuelto por referencia
        {
// Función que abre la conexión a la base de datos
try
            {
                g_conn.Open();  // Abre la conexión con la base de datos
            }
catch (Exception e)
            {
                p_mess = "Error en Conectar con la Base de Datos" +
Convert.ToString(e);
return (false);
            }
return (true);
        }

publicstaticbool cerrar_database(refstring p_mess)
// Mensaje de error devuelto por referencia
        {
// Función que cierra la conexión a la base de datos
try
            {
                g_conn.Close(); // Cierra la conexión con la base de datos
            }
catch (Exception e)
            {
                p_mess = "Error en Desconectar con la Base de Datos" +
e.ToString();
return (false);
            }
return (true);
        }

        #region Backup DB
publicstaticstring backup_datbase(string p_nbac)
// Parámetro recibido de dirección y nombre del backup
        {
// Función que realiza el backup de la base de datos
string l_erro = string.Empty;

try
            {
string l_connStr = "Data Source=SANDINO-PC;Initial
Catalog=Datos_LaPulpita;Integrated Security=True";

using (SqlConnection l_conn = newSqlConnection(l_connStr))
```

```
                {
string sqlStmt = String.Format("BACKUP DATABASE Datos_LaPulpita TO DISK='{0}'",
p_nbac);

using (SqlCommand bu2 = newSqlCommand(sqlStmt, l_conn))
                    {
                        l_conn.Open();
                        bu2.ExecuteNonQuery();
                        l_conn.Close();
                    }
                }
            }
catch (Exception e)
{
                l_erro = "Error: " + e.ToString();
g_conn.Close();
            }

return (l_erro);
}
        #endregion

        #region Usuario Password
publicstaticSqlDataReader consulta_usuario_password(string p_usua)
// Parámetro recibido usuario
        {
/* Proceso almacenado para Usuarios
            / Input:   p_usua --> usuario
/ Output:  return --> datos consultados
            */
SqlCommand l_comd = newSqlCommand("dbo.pa_Usuarios", g_conn);
// Ejecuta el proceso almacenado
            l_comd.CommandType = CommandType.StoredProcedure;
// Comando del proceso almacenado
            l_comd.Parameters.Add("@p_Usuario", SqlDbType.NVarChar);
// Llamada al parámetro del sp
            l_comd.Parameters["@p_Usuario"].Value = p_usua;
// Valor del parámetro del sp
SqlDataReader l_dato = l_comd.ExecuteReader();
// Devuelve el objeto del resultado del select de la tabla
return (l_dato);
        }
        #endregion

        #region Usuarios
publicstaticSqlDataReader consulta_usuarios()
        {
/* Proceso almacenado para Usuarios
/ Input:
            / Output:  return --> datos consultados
            */
SqlCommand l_comd = newSqlCommand("dbo.sp_Usuarios", g_conn);
// Ejecuta el proceso almacenado
            l_comd.CommandType = CommandType.StoredProcedure;
// Comando del proceso almacenado
SqlDataReader l_dato = l_comd.ExecuteReader();
// Devuelve el objeto del resultado del select de la tabla
```

```
return (l_dato);
        }
        #endregion

        #region Consulta todos los campos de las tablas
publicstaticSqlDataReader consulta_datos_tabla(string p_nosp)
// Parámetro recibe el nompre del proceso almacenado
        {
/* Proceso almacenado para Usuarios
            / Input:   p_nosp --> Nombre del proceso almacenado
/ Output:  return --> datos consultados
            */
SqlCommand l_comd = newSqlCommand(p_nosp, g_conn);
// Ejecuta el proceso almacenado
            l_comd.CommandType = CommandType.StoredProcedure;
// Comando del proceso almacenado
SqlDataReader l_dato = l_comd.ExecuteReader();
// Devuelve el objeto del resultado del select de la tabla
return (l_dato);
        }
        #endregion

        #region Delete dato tabla
publicstaticstring delete_row_table(paramsobject[] p_parm)   // Datos de entrada
{
/* Proceso almacenado para Usuarios
/ Input:   p_parm --> Datos
/ Output:  return --> Error al eliminar dato
            */
string l_erro = string.Empty;
int l_coun = 0;
int l_npar = 0;
int l_dato = 0;

            l_coun = p_parm.Count();

try
            {
                g_conn.Open();
SqlCommand l_comd = newSqlCommand(p_parm[0].ToString(), g_conn);
// Ejecuta el proceso almacenado
                l_comd.CommandType = CommandType.StoredProcedure;
// Comando del proceso almacenado
                l_coun = (l_coun - 1) / 3;

while (l_npar < l_coun)
                {
                    l_dato += 1;
switch (l_dato)
                    {
case 1:                     // Campo con valor String
                            l_comd.Parameters.Add(p_parm[l_dato + 1].ToString(),
SqlDbType.NChar).Value = p_parm[l_dato + 2];     // Llamada al parámetro del sp
break;
case 2:                     // Campo con valor
//l_quer += " USING (SELECT " + p_parm[3] + " AS ID)";
break;
```

```
default:
break;
                    }
                    l_npar += 1;
}

                l_comd.ExecuteNonQuery();
// Ejecución de la sentencia proceso almacenado
g_conn.Close();
            }
catch (Exception e)
{
                l_erro = "Error: " + e.ToString();
g_conn.Close();
            }
return (l_erro);
}
        #endregion

        #region Insertar o Actualizar Usuarios
publicstaticstring insert_update_table(paramsobject[] p_parm)
// Parámetro Datos
{
/* Proceso almacenado para Usuarios
/ Input:   p_parm --> Datos
/ Output:  return --> Error
            */
string l_erro = string.Empty;

try
            {
                g_conn.Open();
SqlCommand l_comd = newSqlCommand(p_parm[0].ToString(), g_conn);
// Ejecuta el proceso almacenado
l_comd.CommandType = CommandType.StoredProcedure;
// Comando del proceso almacenado
                l_comd.Parameters.Add("@p_usua", SqlDbType.NChar, 10).Value =
p_parm[1];     // Llamada al parámetro del sp
                l_comd.Parameters.Add("@p_pass", SqlDbType.NChar, 10).Value =
p_parm[2];     // Llamada al parámetro del sp
                l_comd.Parameters.Add("@p_nomb", SqlDbType.NChar, 30).Value =
p_parm[3];     // Llamada al parámetro del sp
                l_comd.Parameters.Add("@p_admi", SqlDbType.Bit).Value =
p_parm[4];     // Llamada al parámetro del sp
                l_comd.Parameters.Add("@p_logi", SqlDbType.Bit).Value =
p_parm[5];     // Llamada al parámetro del sp
                l_comd.Parameters.Add("@p_prod", SqlDbType.Bit).Value =
p_parm[6];     // Llamada al parámetro del sp
                l_comd.Parameters.Add("@p_dage", SqlDbType.Bit).Value =
p_parm[7];     // Llamada al parámetro del sp
                l_comd.Parameters.Add("@p_soco", SqlDbType.Bit).Value =
p_parm[8];     // Llamada al parámetro del sp
                l_comd.ExecuteNonQuery();
// Ejecución de la sentencia proceso almacenado
g_conn.Close();
            }
catch (Exception e)
```

```
{
                l_erro = "Error: " + e.ToString();
g_conn.Close();
            }
return (l_erro);
}
        #endregion

        #region Insertar o Actualizar Auditoría
publicstaticstring insert_update_auditoria(paramsobject[] p_parm)
// Parámetro Dato
{
/* Proceso almacenado para Usuarios
/ Input:   p_parm --> Datos
/ Output:  return --> Error
            */
string l_erro = string.Empty;

try
            {
                g_conn.Open();
SqlCommand l_comd = newSqlCommand(p_parm[0].ToString(), g_conn);
// Ejecuta el proceso almacenado
l_comd.CommandType = CommandType.StoredProcedure;
// Comando del proceso almacenado
                l_comd.Parameters.Add("@p_usua", SqlDbType.NChar, 10).Value =
p_parm[1];     // Llamada al parámetro del sp
                l_comd.Parameters.Add("@p_fing", SqlDbType.DateTime2).Value =
p_parm[2];     // Llamada al parámetro del sp
                l_comd.Parameters.Add("@p_sesi", SqlDbType.NChar, 30).Value =
p_parm[3];     // Llamada al parámetro del sp
                l_comd.Parameters.Add("@p_cons", SqlDbType.Bit).Value =
p_parm[4];     // Llamada al parámetro del sp
                l_comd.Parameters.Add("@p_impr", SqlDbType.Bit).Value =
p_parm[5];     // Llamada al parámetro del sp
                l_comd.Parameters.Add("@p_nuev", SqlDbType.Bit).Value =
p_parm[6];     // Llamada al parámetro del sp
                l_comd.Parameters.Add("@p_actu", SqlDbType.Bit).Value =
p_parm[7];     // Llamada al parámetro del sp
                l_comd.Parameters.Add("@p_elim", SqlDbType.Bit).Value =
p_parm[8];     // Llamada al parámetro del sp
                l_comd.ExecuteNonQuery();
// Ejecución de la sentencia proceso almacenado
g_conn.Close();
            }
catch (Exception e)
{
                l_erro = "Error: " + e.ToString();
g_conn.Close();
            }
return (l_erro);
}
        #endregion

        #region Obtener Fecha y Hora del DB
publicstaticDateTime obtener_fecha_hora_database()
        {
```

```
// Se obtiene la fecha actual de la base de datos
DateTime l_fech = newDateTime();

            g_conn.Open();
SqlCommand l_comd = newSqlCommand("dbo.sp_DateHours", g_conn);
// Ejecuta el proceso almacenado
l_comd.CommandType = CommandType.StoredProcedure;
// Comando del proceso almacenado
SqlDataReader l_dato = l_comd.ExecuteReader();
// Devuelve el objeto del resultado del select de la tabla
while (l_dato.Read())
            {
                l_fech = Convert.ToDateTime(l_dato[0]);
            }
            g_conn.Close();

return(l_fech);
}
        #endregion

        #region Insertar o Actualizar Tipo Identificacion
publicstaticstring insert_update_tipo_identificacion(paramsobject[] p_parm)
// Parámetros datos
{
/* Proceso almacenado para Usuarios
/ Input:   p_parm --> Datos
/ Output:  return --> Error
            */
string l_erro = string.Empty;

try
            {
                g_conn.Open();
SqlCommand l_comd = newSqlCommand(p_parm[0].ToString(), g_conn);
// Ejecuta el proceso almacenado
l_comd.CommandType = CommandType.StoredProcedure;
// Comando del proceso almacenado
                l_comd.Parameters.Add("@p_tide", SqlDbType.NChar, 1).Value =
p_parm[1];     // Llamada al parámetro del sp
                l_comd.Parameters.Add("@p_desc", SqlDbType.NChar, 30).Value =
p_parm[2];     // Llamada al parámetro del sp
                l_comd.ExecuteNonQuery();
// Ejecución de la sentencia proceso almacenado
g_conn.Close();
            }
catch (Exception e)
{
                l_erro = "Error: " + e.ToString();
g_conn.Close();
            }
return (l_erro);
}
        #endregion

        #region Insertar o Actualizar Paises
publicstaticstring insert_update_paises(paramsobject[] p_parm)
// Parámetro Datos
```

```
{
/* Proceso almacenado para Usuarios
/ Input:   p_parm --> Datos
/ Output:  return --> Error
            */
string l_erro = string.Empty;

try
            {
                g_conn.Open();
SqlCommand l_comd = newSqlCommand(p_parm[0].ToString(), g_conn);
// Ejecuta el proceso almacenado
l_comd.CommandType = CommandType.StoredProcedure;
// Comando del proceso almacenado
                l_comd.Parameters.Add("@p_pais", SqlDbType.NChar, 3).Value =
p_parm[1];     // Llamada al parámetro del sp
                l_comd.Parameters.Add("@p_desc", SqlDbType.NChar, 30).Value =
p_parm[2];     // Llamada al parámetro del sp
                l_comd.ExecuteNonQuery();
// Ejecución de la sentencia proceso almacenado
g_conn.Close();
            }
catch (Exception e)
{
                l_erro = "Error: " + e.ToString();
g_conn.Close();
            }
return (l_erro);
}
        #endregion

        #region Insertar o Actualizar Estados Provincias
publicstaticstring insert_update_estados_provincias(paramsobject[] p_parm)
// Parámetros Datos
        {
/* Proceso almacenado para Usuarios
/ Input:   p_parm --> Datos
/ Output:  return --> Error
            */
string l_erro = string.Empty;

try
            {
                g_conn.Open();
SqlCommand l_comd = newSqlCommand(p_parm[0].ToString(), g_conn);
// Ejecuta el proceso almacenado
l_comd.CommandType = CommandType.StoredProcedure;
// Comando del proceso almacenado
                l_comd.Parameters.Add("@p_pais", SqlDbType.NChar, 3).Value =
p_parm[1];     // Llamada al parámetro del sp
                l_comd.Parameters.Add("@p_epro", SqlDbType.NChar, 3).Value =
p_parm[2];     // Llamada al parámetro del sp
                l_comd.Parameters.Add("@p_desc", SqlDbType.NChar, 30).Value =
p_parm[3];     // Llamada al parámetro del sp
                l_comd.ExecuteNonQuery();
// Ejecución de la sentencia proceso almacenado
g_conn.Close();
```

```
            }
catch (Exception e)
{
                l_erro = "Error: " + e.ToString();
g_conn.Close();
            }
return (l_erro);
}
        #endregion

        #region Insertar o Actualizar Ciudades
publicstaticstring insert_update_ciudades(paramsobject[] p_parm) // Datos
{
/* Proceso almacenado para Usuarios
/ Input:   p_parm --> Datos
/ Output:  return --> Error
            */
string l_erro = string.Empty;

try
            {
                g_conn.Open();
SqlCommand l_comd = newSqlCommand(p_parm[0].ToString(), g_conn);
// Ejecuta el proceso almacenado
l_comd.CommandType = CommandType.StoredProcedure;
// Comando del proceso almacenado
                l_comd.Parameters.Add("@p_pais", SqlDbType.NChar, 3).Value =
p_parm[1];     // Llamada al parámetro del sp
                l_comd.Parameters.Add("@p_epro", SqlDbType.NChar, 3).Value =
p_parm[2];     // Llamada al parámetro del sp
                l_comd.Parameters.Add("@p_ciud", SqlDbType.NChar, 3).Value =
p_parm[3];     // Llamada al parámetro del sp
                l_comd.Parameters.Add("@p_desc", SqlDbType.NChar, 30).Value =
p_parm[4];     // Llamada al parámetro del sp
                l_comd.ExecuteNonQuery();
// Ejecución de la sentencia proceso almacenado
g_conn.Close();
            }
catch (Exception e)
{
                l_erro = "Error: " + e.ToString();
g_conn.Close();
            }
return (l_erro);
}
        #endregion

        #region Busca existencia dato
publicstaticSqlDataReader valida_existencia_pais(string p_pais)
// Parámetro país
        {
/* Proceso almacenado para Usuarios
/ Input:   p_pais --> País
/ Output:  return --> datos consultados
            */
string l_erro = string.Empty;
```

```
            g_conn.Open();
SqlCommand l_comd = newSqlCommand("dbo.sp_SelectPais", g_conn);
// Ejecuta el proceso almacenado
l_comd.CommandType = CommandType.StoredProcedure;
// Comando del proceso almacenado
            l_comd.Parameters.Add("@p_pais", SqlDbType.NChar, 3).Value = p_pais;
// Llamada al parámetro del sp
SqlDataReader l_dato = l_comd.ExecuteReader();
// Devuelve el objeto del resultado del select de la tabla
return (l_dato);
        }
        #endregion
    }
}
```

b) **Funciones.cs**, contiene las funciones que sirven para realizar validaciones generales para el sistema:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using CRM___WAP;
using System.Data.Sql;
using System.Data.SqlClient;
using Conectar_DB;

namespace Funciones_Generales
{
classFunciones
    {
        #region Carga los datos de la auditoría
publicstaticvoid cargar_auditoria(DateTime p_dtim, string p_form, bool p_cons,
bool p_impr, bool p_nuev, bool p_actu, bool p_elim)
{
// Se actualiza la tabla de auditoría
string l_usuario = string.Empty;

frm_MPrincipal l_mprin = newfrm_MPrincipal();
// Instancia del from Login
            l_usuario = l_mprin.Codigo_Usuario;
       // Cargar usuario logeado


Conectar_DB.Conexion.insert_update_auditoria("dbo.sp_InsertAuditoria",
l_usuario, p_dtim, p_form,
                                                        p_cons, p_impr, p_nuev,
p_actu, p_elim);
        }
        #endregion
```

```
    }
}
```

4.3.2. LOGIN

a) Sesión que permite el ingreso al sistema, de aquellos usuarios que se encuentran registrados en la base de datos:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Data.Sql;
using System.Data.SqlClient;
using Conectar_DB;

namespace CRM___WAP
{
publicpartialclassfrm_Login : Form
    {
// Variables Globales
string g_usuario;

public frm_Login()
        {
            InitializeComponent();
        }

        #region Boton Cancelar
privatevoid _btnCancelar_Click(object sender, EventArgs e)
{
// Cancelación del inicio del sistema
Application.Exit();
        }
        #endregion

        #region Boton Ingresar
privatevoid _btnIngresar_Click(object sender, EventArgs e)
        {
// Ingreso al sistema
string l_mess = string.Empty;
SqlDataReader l_dato;

this._errorLogin.Clear();           // Limpia los mensajes de error

if (Conectar_DB.Conexion.abrir_database(ref l_mess) == true)
```

```
// Conectar base datos
            {
                l_dato =
Conectar_DB.Conexion.consulta_usuario_password(this._txtUsuario.Text);
// Consulta del usuario y password

while (l_dato.Read())            // Lectura de los datos consultados
{
if (this._txtUsuario.Text == l_dato[0].ToString().Trim()
&&this._txtPassword.Text == l_dato[1].ToString().Trim())
{
// Usuario y Password correctos
                        g_usuario = this._txtUsuario.Text;
this.Close();
                    }
else
                    {
if (this._txtUsuario.Text == l_dato[0].ToString().Trim())
                        {
// Password incorrecto
this._errorLogin.SetError(this._txtPassword, "El password ingresado es
incorrecto!.");
                        }
                    }
                }
if (!l_dato.HasRows)
{
// Usuario y Password incorrectos
this._errorLogin.SetError(this._txtUsuario, "El usuario ingresado no existe!.");
this._errorLogin.SetError(this._txtPassword, "El password ingresado es
incorrecto!.");
                }
            }
else
            {
MessageBox.Show(l_mess);
}

if (Conectar_DB.Conexion.cerrar_database(ref l_mess) == false)
// Cerrar conexión base datos
            {
MessageBox.Show(l_mess);
            }
        }
        #endregion

        #region Finciones Varias
publicstring Codigo_Usuario
        {
// Cargar usuario logeado
get
            {
return (g_usuario);
            }
        }
        #endregion
    }
```

```
}
```

4.3.3. MENÚ PRINCIPAL

a) Esta es la sesión principal del sistema, la cual contiene todos los accesos a los distintos módulos, que tenga permisos el usuario logeado:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Data.Sql;
using System.Data.SqlClient;
using Conectar_DB;

namespace CRM___WAP
{
publicpartialclassfrm_MPrincipal : Form
{
// Variables Globales
publicstaticstring g_usuario;                // Variable general del sistema
código de Usuario
privatestaticobject[] g_cons;             // Variable general del sistema
usuario consulta todo

public frm_MPrincipal()
        {
            InitializeComponent();
g_cons[0] =false;     // Datos Generales solo consultar
g_cons[1] =false;     // Producción solo consultar
g_cons[2] =false;     // Logístico solo consultar
g_cons[3] =false;     // Sistema Solo Consultar
}

        #region Form Menu Principal
privatevoid frm_MPrincipal_Load(object sender, EventArgs e)
{
frm_Login l_Login = newfrm_Login();                // Instancia del from Login
l_Login.ShowDialog();
            g_usuario = l_Login.Codigo_Usuario;        // Cargar usuario logeado
            Bloquea_Opciones_Menu();
        }
        #endregion

        #region Usuarios - Auditoría
privatevoid _btnUsuarios_Click(object sender, EventArgs e)
{
```

```
// Llamada a la sesión de usuarios y auditoría
frm_UsuarioAuditoria l_uadi = newfrm_UsuarioAuditoria();
l_uadi.Show();
        }
        #endregion

        #region Parámetros
privatevoid _btnParametros_Click(object sender, EventArgs e)
        {
// Llamada a la sesión de todos los parámetros
frm_Parametros l_para = newfrm_Parametros();
            l_para.Show();
        }
        #endregion

        #region Datos Generales
privatevoid _btnDatosGenerales_Click(object sender, EventArgs e)
{
// Llamada a la sesión de todos los datos generales
frm_DatosGenerales l_dgen = newfrm_DatosGenerales();
            l_dgen.Show();
        }
        #endregion

        #region Fabricación
privatevoid _btnFabricacion_Click(object sender, EventArgs e)
{
frm_Fabricacion l_fabr = newfrm_Fabricacion();
            l_fabr.Show();
}
        #endregion

        #region Almacenamiento
privatevoid _btnAlmacenamiento_Click(object sender, EventArgs e)
        {
frm_Almacenamiento l_alma = newfrm_Almacenamiento();
l_alma.Show();
        }
        #endregion

        #region
privatevoid _btnVentas_Click(object sender, EventArgs e)
{
frm_Ventas l_vent = newfrm_Ventas();
            l_vent.Show();
        }
        #endregion

        #region Compras
privatevoid _btnCompras_Click(object sender, EventArgs e)
        {
frm_Compras l_comp = newfrm_Compras();
l_comp.Show();
        }
        #endregion

        #region Funciones Varios
```

```
privatevoid Bloquea_Opciones_Menu()
        {
// Función para deshabilitar las opciones del menú
string l_mess = string.Empty;
SqlDataReader l_dato;
            g_cons = newobject[4];

if (Conectar_DB.Conexion.abrir_database(ref l_mess) == true)
// Conectar base datos
            {
                l_dato =
Conectar_DB.Conexion.consulta_usuario_password(g_usuario);
// Consulta del usuario y password

while (l_dato.Read())            // Lectura de los datos consultados
{
if (!Convert.ToBoolean(l_dato[3]))
                     // Usuario Administrador
                    {
this._btnParametros.Enabled = false;
this._btnUsuarios.Enabled = false;
                    }

if (!Convert.ToBoolean(l_dato[4]) && !Convert.ToBoolean(l_dato[5]) &&
Convert.ToBoolean(l_dato[6]) && !Convert.ToBoolean(l_dato[7]))
// Usuario para Datos Generales
{
                        g_cons[1] = true;
                        g_cons[2] = true;
                    }

if (!Convert.ToBoolean(l_dato[4]) &&Convert.ToBoolean(l_dato[5]) &&
!Convert.ToBoolean(l_dato[6]) && !Convert.ToBoolean(l_dato[7]))
// Usuario Producción y Almacenaje
{
                        g_cons[0] = true;
                        g_cons[2] = true;
                    }

if (Convert.ToBoolean(l_dato[4]) && !Convert.ToBoolean(l_dato[5]) &&
!Convert.ToBoolean(l_dato[6]) && !Convert.ToBoolean(l_dato[7]))
// Usuario Logístico
{
                        g_cons[0] = true;
                        g_cons[1] = true;
}

if (Convert.ToBoolean(l_dato[7]))
                     // Usuario Solo Consulta de todo el sistema
{
                        g_cons[3] = true;
                    }

if (!Convert.ToBoolean(l_dato[4]) &&Convert.ToBoolean(l_dato[5]) &&
Convert.ToBoolean(l_dato[6]) && !Convert.ToBoolean(l_dato[7]))
// Usuario para Datos Generales y Producción
{
```

```
                        g_cons[2] = true;
                    }

if (Convert.ToBoolean(l_dato[4]) && !Convert.ToBoolean(l_dato[5]) &&
Convert.ToBoolean(l_dato[6]) && !Convert.ToBoolean(l_dato[7]))
// Usuario para Datos Generales y Logístico
{
                        g_cons[1] = true;
                    }

if (!Convert.ToBoolean(l_dato[4]) && !Convert.ToBoolean(l_dato[5]) &&
Convert.ToBoolean(l_dato[6]) && !Convert.ToBoolean(l_dato[7]))
// Usuario Producción y Logístico
{
                        g_cons[0] = true;
                    }
                }
            }
else
            {
MessageBox.Show(l_mess);
}

if (Conectar_DB.Conexion.cerrar_database(ref l_mess) == false)
// Cerrar conexión base datos
{
MessageBox.Show(l_mess);
            }
        }

publicstring Codigo_Usuario
{
// Cargar usuario logeado
get
            {
return (g_usuario);
}
        }

publicobject[] Usuarios_Consulta
        {
get
            {
return (g_cons);
            }
        }
        #endregion
    }
}
```

4.3.4. MENÚS SECUNDARIOS

a) **frm_UsuarioAuditoria**, menú de usuarios, auditoría y backup de la base de datos:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace CRM___WAP
{
publicpartialclassfrm_UsuarioAuditoria : Form
{
public frm_UsuarioAuditoria()
        {
            InitializeComponent();
        }

privatevoid _btnUsuarios_Click(object sender, EventArgs e)
{
// Llamada a la sesión de usuarios
frm_Usuarios l_usua = newfrm_Usuarios();
l_usua.Show();
        }

privatevoid _btnBackup_Click(object sender, EventArgs e)
{
// Llamada a la sesión de backup
frm_Backup l_back = newfrm_Backup();
            l_back.Show();
        }

privatevoid _btnAuditoria_Click(object sender, EventArgs e)
{
// Llamada a la sesión de auditoría
frm_Auditoria l_audi = new frm_Auditoria();
            l_audi.Show();
        }

privatevoid _btnAuditoriaAuditoria_Click(object sender, EventArgs e)
{
// Llamada a la sesión de auditoría auditoría
frm_AuditoriaAuditoria l_aaud = new frm_AuditoriaAuditoria();
l_aud.Show();
        }
    }
   }
```

b) **frm_Parametros**, menú donde se encuentra todos los parámetros del sistema:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace CRM___WAP
{
publicpartialclassfrm_Parametros : Form
{
public frm_Parametros()
        {
            InitializeComponent();
        }

        #region Botón Parámetros Empresa
privatevoid _btnPEmpresa_Click(object sender, EventArgs e)
{
// Llamada a la sesión de datos empresa
frm_DatosEmpresa l_demp = newfrm_DatosEmpresa();
            l_demp.Show();
        }
        #endregion

        #region Botón Tipos Identificación
privatevoid _btnTiposIdentidicacion_Click(object sender, EventArgs e)
        {
// Llamada a la sesión de tipos de identificación
frm_TipoIdentificacion l_tide = newfrm_TipoIdentificacion();
            l_tide.Show();
        }
        #endregion

        #region Botón Estados Civiles
privatevoid _btnEstadosCiviles_Click(object sender, EventArgs e)
{
// Llamada a la sesión de estado civil
frm_EstadoCivil l_eciv = newfrm_EstadoCivil();
l_eciv.Show();
        }
        #endregion

        #region Botón Divisas
privatevoid _btnDivisas_Click(object sender, EventArgs e)
{
// Llamada a la sesión de divisas
frm_Divisas l_divi = newfrm_Divisas();
l_divi.Show();
```

```
}
        #endregion

        #region Botón Magnitudes Físicas
privatevoid _btnMagnitudesF_Click(object sender, EventArgs e)
{
// Llamada a la sesión de magnitudes físicas
frm_MagnitudesF l_mfis = newfrm_MagnitudesF();
l_mfis.Show();
        }
        #endregion

        #region Botón Grupos Números
privatevoid _btnGruposNumeros_Click(object sender, EventArgs e)
        {
// Llamada a la sesión de grupos de números
frm_GruposNumeros l_gnum = newfrm_GruposNumeros();
            l_gnum.Show();
        }
        #endregion

        #region Botón Primeros Números Libres
privatevoid _btnPrimerosNL_Click(object sender, EventArgs e)
{
// Llamada a la sesión de primeros números libres
frm_PrimerosNumerosLibres l_pnli = newfrm_PrimerosNumerosLibres();
            l_pnli.Show();
        }
        #endregion

        #region Botón Valores IVA
privatevoid _btnValoresIVA_Click(object sender, EventArgs e)
        {
// Llamada a la sesión de valores iva
frm_ValoresIVA l_viva = newfrm_ValoresIVA();
            l_viva.Show();
        }
        #endregion
    }
}
```

c) **frm_DatosGenerales**, contiene los datos que son generales para el sistema, como: Países, Ciudades, Direcciones, Empleados, Artículos, etc:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
```

```
namespace CRM___WAP
{
publicpartialclassfrm_DatosGenerales : Form
    {
public frm_DatosGenerales()
        {
            InitializeComponent();
        }

        #region Botón Paises
privatevoid _btnPaises_Click(object sender, EventArgs e)
{
// Llamada a la sesión de países
frm_Paises l_pais = newfrm_Paises();
l_pais.Show();
        }
        #endregion

        #region Botón Estados Provincias
privatevoid _btnEstadosProvincias_Click(object sender, EventArgs e)
        {
// Llamada a la sesión de estados y provincias
frm_EstadosProvincias l_epro = newfrm_EstadosProvincias();
            l_epro.Show();
        }
        #endregion

        #region Botón Ciudades
privatevoid _btnCiudades_Click(object sender, EventArgs e)
{
// Llamada a la sesión de ciudades
frm_Ciudades l_ciud = newfrm_Ciudades();
            l_ciud.Show();
        }
        #endregion

        #region Botón Direcciones
privatevoid _btnDirecciones_Click(object sender, EventArgs e)
        {
// Llamada a la sesión de direcciones
frm_Direcciones l_dire = newfrm_Direcciones();
            l_dire.Show();
        }
        #endregion

        #region Botón Empleados
privatevoid _btnEmpleados_Click(object sender, EventArgs e)
{
// Llamada a la sesión de empleados
frm_Empleados l_empl = newfrm_Empleados();
            l_empl.Show();
        }
        #endregion

        #region Botón Artículos
privatevoid _btnArtículos_Click(object sender, EventArgs e)
        {
```

```
// Llamada a la sesión de artículos
frm_Articulos l_arti = newfrm_Articulos();
l_arti.Show();
        }
        #endregion

        #region Botón Fabricantes
privatevoid _Fabricantes_Click(object sender, EventArgs e)
        {
// Llamada a la sesión de fabricantes
frm_Fabricantes l_fabi = newfrm_Fabricantes();
l_fabi.Show();
        }
        #endregion

        #region Botón Idiomas
privatevoid _btnIdiomas_Click(object sender, EventArgs e)
        {
// Llamada a la sesión de idiomas
frm_Idiomas l_idio = newfrm_Idiomas();
l_idio.Show();
        }
        #endregion

        #region Botón Partners
privatevoid _btnPartners_Click(object sender, EventArgs e)
{
// Llamada a la sesión de partners
frm_Partners l_part = newfrm_Partners();
l_part.Show();
        }
        #endregion

        #region Botón Unidades
privatevoid _btnUnidades_Click(object sender, EventArgs e)
        {
// Llamada a la sesión de unidades
frm_Unidades l_unid = newfrm_Unidades();
            l_unid.Show();
        }
        #endregion
    }
}
```

d) **frm_Compras**, contiene llamadas a las opciones de compras:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
```

```
using System.Windows.Forms;

namespace CRM___WAP
{
publicpartialclassfrm_Compras : Form
{
public frm_Compras()
        {
            InitializeComponent();
        }

        #region Botón Departamentos
privatevoid _btnDepartamentos_Click(object sender, EventArgs e)
        {
// Llamada a la sesión de departamentos
frm_Departamentos l_depa = newfrm_Departamentos();
            l_depa.Show();
        }
        #endregion

        #region Botón Artículos de Compra
privatevoid _btnArticulosCompra_Click(object sender, EventArgs e)
        {
// Llamada a la sesión de artículos de compra
frm_ArticulosCompra l_acom = newfrm_ArticulosCompra();
            l_acom.Show();
        }
        #endregion

        #region Botón Tipos de Orden de Compra
privatevoid _btnTOrdenC_Click(object sender, EventArgs e)
{
// Llamada a la sesión de tipos de orden de compra
frm_TiposOrdenCompra l_toco = newfrm_TiposOrdenCompra();
            l_toco.Show();
        }
        #endregion

        #region Botón Motivos de Devolución
privatevoid _btnMotivosD_Click(object sender, EventArgs e)
{
// Llamada a la sesión de motivos de devolución
frm_MotivosDevolucion l_mdev = newfrm_MotivosDevolucion();
            l_mdev.Show();
        }
        #endregion

        #region Botón Órdenes de Compra
privatevoid _btnOrdenesCompra_Click(object sender, EventArgs e)
{
// Llamada a la sesión de órdenes de compra
frm_OrdenesCompra l_ocom = newfrm_OrdenesCompra();
l_ocom.Show();
        }
        #endregion
    }
}
```

e) **frm_Ventas**, menú que contiene todo lo referente a ventas, el realizar una venta, y crear también cotizaciones, para en un proceso posterior pasar a convertirse en una orden de venta:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace CRM___WAP
{
publicpartialclassfrm_Ventas : Form
{
public frm_Ventas()
        {
            InitializeComponent();
        }

        #region Botón Departamentos
privatevoid _btnDepartamentos_Click(object sender, EventArgs e)
        {
// Llamada a la sesión de departamentos
frm_Departamentos l_depa = newfrm_Departamentos();
            l_depa.Show();
        }
        #endregion

        #region Botón Artículos de Venta
privatevoid _btnArticulosVenta_Click(object sender, EventArgs e)
{
// Llamada a la sesión de artículos de venta
frm_ArticulosVenta l_aven = newfrm_ArticulosVenta();
            l_aven.Show();
}
        #endregion

        #region Botón Tipos de Orden de Venta
privatevoid _btnTOrdenV_Click(object sender, EventArgs e)
{
// Llamada a la sesión de tipos de orden de venta
frm_TiposOrdenVenta l_tove = newfrm_TiposOrdenVenta();
            l_tove.Show();
        }
        #endregion

        #region Botón Motivos de Devolución
privatevoid _btnMotivosD_Click(object sender, EventArgs e)
{
// Llamada a la sesión de motivos de devolución
```

```
frm_MotivosDevolucion l_mdev = newfrm_MotivosDevolucion();
            l_mdev.Show();
        }
        #endregion

        #region Botón Ofertas de Venta
privatevoid _btnOfertasVenta_Click(object sender, EventArgs e)
{
// Llamada a la sesión de ofertas de venta
frm_OfertasVenta l_ofve = newfrm_OfertasVenta();
            l_ofve.Show();
        }
        #endregion

        #region Botón Órdenes de Venta
privatevoid _btnOrdenesVenta_Click(object sender, EventArgs e)
{
// Llamada a la sesión de órdenes de venta
frm_OrdenesVenta l_orve = newfrm_OrdenesVenta();
l_orve.Show();
        }
        #endregion
    }
}
```

f) **frm_Almacenamiento**, contiene las funciones necesarias para que se administre el almacenamiento y stock:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace CRM___WAP
{
publicpartialclassfrm_Almacenamiento : Form
{
public frm_Almacenamiento()
        {
            InitializeComponent();
        }

privatevoid _btnTiposAlmacen_Click(object sender, EventArgs e)
        {
// Llamada a la sesión de tipos de almacén
frm_TipoAlmacen l_talm = newfrm_TipoAlmacen();
            l_talm.Show();
```

```
        }

privatevoid _btnTiposUbicacion_Click(object sender, EventArgs e)
        {
// Llamada a la sesión de tipos de ubicación
frm_TiposUbicacion l_tubi = newfrm_TiposUbicacion();
            l_tubi.Show();
        }

privatevoid _btnTiposMovimiento_Click(object sender, EventArgs e)
        {
// Llamada a la sesión de tipos de movimiento
frm_TiposMovimiento l_tmov = newfrm_TiposMovimiento();
            l_tmov.Show();
        }

privatevoid _TiposExpedidorR_Click(object sender, EventArgs e)
        {
// Llamada a la sesión de tipos de expedidor y receptor
frm_TiposExpedidorR l_tere = newfrm_TiposExpedidorR();
l_tere.Show();
        }

privatevoid _btnAlmacenes_Click(object sender, EventArgs e)
{
// Llamada a la sesión de almacenes
frm_Almacenes l_alma = newfrm_Almacenes();
            l_alma.Show();
        }

privatevoid _btnAlmacenesUbicaciones_Click(object sender, EventArgs e)
        {
// Llamada a la sesión de almacenes y ubicaciones
frm_AlmacenesUbicaciones l_aubi = newfrm_AlmacenesUbicaciones();
            l_aubi.Show();
        }

privatevoid _btnAlmacenesUArticulos_Click(object sender, EventArgs e)
        {
// Llamada a la sesión de almacenes ubicaciones artículos
            frm_AUbicacionesArticulos l_auar = new frm_AUbicacionesArticulos();
            l_auar.Show();
        }

privatevoid _btnAlmacenesArticulos_Click(object sender, EventArgs e)
        {
// Llamada a la sesión de almacén y artículos
            frm_AlmacenArticulos l_aart = new frm_AlmacenArticulos();
l_aart.Show();
        }

privatevoid _btnClaseProducto_Click(object sender, EventArgs e)
{
// Llamada a la sesión de Clases de producto
frm_ClasesProductos l_cpro = newfrm_ClasesProductos();
l_cpro.Show();
        }
```

```
privatevoid _btnOrigenesOrdenes_Click(object sender, EventArgs e)
{
// Llamada a la sesión de orígenes de orden
frm_OrigenesOrden l_oord = new frm_OrigenesOrden();
l_oord.Show();
        }

privatevoid _btnMotivosCorreccion_Click(object sender, EventArgs e)
{
// Llamada a la sesión de motivos de corrección
frm_MotivosCorreccion l_mcor = newfrm_MotivosCorreccion();
            l_mcor.Show();
        }

privatevoid _btnOrdenesAlmacenamiento_Click(object sender, EventArgs e)
        {
// Llamada a la sesión de órdenes de almacenamiento
frm_OrdenesAlmacenaje l_oalm = newfrm_OrdenesAlmacenaje();
l_oalm.Show();
        }

privatevoid _btnOrdenesCorreccion_Click(object sender, EventArgs e)
{
// Llamada a la sesión de órdenes de corrección
frm_OrdenesCorreccion l_ocor = newfrm_OrdenesCorreccion();
            l_ocor.Show();
        }

privatevoid _btnStockArticulo_Click(object sender, EventArgs e)
{
// Llamada a la sesión de stock del artículo
frm_StockArticulo l_sart = newfrm_StockArticulo();
l_sart.Show();
        }

privatevoid _btnStockAlmacenA_Click(object sender, EventArgs e)
{
// Llamada a la sesión de stock almacén artículo
frm_StockAlmacenArticulo l_saar = newfrm_StockAlmacenArticulo();
l_saar.Show();
        }

privatevoid _btnStockUbicacionA_Click(object sender, EventArgs e)
{
// Llamada a la sesión de stock ubicación artículo
            frm_StockUbicacionArticulo l_suar = new
frm_StockUbicacionArticulo();
            l_suar.Show();
        }
    }
}
```

g) **frm_Fabricacion**, contiene el llamado a: Centros de Trabajo, Artículos de Fabricación, Máquinas, etc., para crear una orden de fabricación:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace CRM___WAP
{
publicpartialclassfrm_Fabricacion : Form
{
public frm_Fabricacion()
        {
            InitializeComponent();
        }

privatevoid _btnCentroTrabajo_Click(object sender, EventArgs e)
        {
// Llamada a la sesión de centros de trabajo
frm_CentrosTrabajo l_ctra = newfrm_CentrosTrabajo();
l_ctra.Show();
        }

privatevoid _btnArticulosFabricacion_Click(object sender, EventArgs e)
{
// Llamada a la sesión de artículos de fabricación
frm_ArticulosFabricacion l_afab = newfrm_ArticulosFabricacion();
l_afab.Show();
        }

privatevoid _btnMaquinas_Click(object sender, EventArgs e)
{
// Llamada a la sesión de máquinas
frm_Maquinas l_maqu = newfrm_Maquinas();
l_maqu.Show();
        }

privatevoid _btnTareas_Click(object sender, EventArgs e)
{
// Llamada a la sesión de Tareas
frm_Tareas l_tare = newfrm_Tareas();
l_tare.Show();
        }

privatevoid _btnOrdenesFabricacion_Click(object sender, EventArgs e)
{
// Llamada a la sesión de órdenes de fabricación
frm_OrdenesFabricacion l_ofab = newfrm_OrdenesFabricacion();
            l_ofab.Show();
```

```
}
    }
}
```

4.3.5. SESIONES DE CARGA DE LOS DATAGRIDS

a) Se presentarán dos ejemplos de sesiones que contienen datagrid's, que en sus filasse cargan registros de las tablas. Al seleccionar una o varias de estas filas, se pueden eliminar registros de la base de datos, así como también se muestra el funcionamiento general en el llamado de funciones y demás procesos:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Data.Sql;
using System.Data.SqlClient;
using Conectar_DB;
using Funciones_Generales;

namespace CRM___WAP
{
publicpartialclassfrm_Usuarios : Form
{
// Variables Locales
privatestaticfrm_SaveUsuarios g_susu;    // Objeto del Form
privatestaticDateTime g_dtim = newDateTime();
       // Fecha para auditoría

public frm_Usuarios()
        {
            InitializeComponent();

            g_dtim = Conectar_DB.Conexion.obtener_fecha_hora_database();
Funciones.cargar_auditoria(g_dtim, "frm_Usuarios", true, false, false, false,
false);

            cargar_datagrid_con_database();     // Cargar datos en el datagrid
}

        #region Form
privatevoid frm_Usuarios_Load(object sender, EventArgs e)
        {
```

```
        }
        #endregion

        #region DataGrid Usuarios
privatevoid _dgvUsuarios_CellFormatting(object sender,
DataGridViewCellFormattingEventArgs e)
        {
//color a cada fila
if (this._dgvUsuarios.Rows.Count > 2)
            {
for (int i = 0; i < (this._dgvUsuarios.Rows.Count) - 1; i++)
                {
if ((i % 2) == 0)
                    {
this._dgvUsuarios.Rows[i].DefaultCellStyle.BackColor = Color.AliceBlue;
                    }
                }
            }
        }

privatevoid _dgvUsuarios_DoubleClick(object sender, EventArgs e)
        {
int l_lfie = 0;
int l_rows = 0;
SqlDataReader l_dato;
string l_pass = string.Empty;
string l_mess = string.Empty;

l_lfie = this._dgvUsuarios.Rows.Count - 1;
              // Obtiene el número de filas
g_susu = newfrm_SaveUsuarios();

// Al seleccionar una fila se envían los valores de columna
l_rows = this._dgvUsuarios.CurrentRow.Index;

if (l_lfie == this._dgvUsuarios.CurrentRow.Index)
              // Comparación última fila y fila posicionada
            {
// Modo de nuevo registro
                g_susu.g_efor = true;
g_susu.Show();
                new_update_row_datagrid(true, 0);
            }
else
            {
if (Conectar_DB.Conexion.abrir_database(ref l_mess) == true)
// Conectar base datos
                {
                    l_dato =
Conectar_DB.Conexion.consulta_usuario_password(this._dgvUsuarios.Rows[l_rows].Ce
lls[0].Value.ToString());         // Consulta del usuario y password
while (l_dato.Read())   // Lectura de los datos consultados
{
                        l_pass = l_dato[1].ToString();
                    }
                }
```

```
else
                {
MessageBox.Show(l_mess);
}

if (Conectar_DB.Conexion.cerrar_database(ref l_mess) == false)
// Cerrar conexión base datos
                {
MessageBox.Show(l_mess);
                }
// Modo de actualización de registro
g_susu.CargarVariables(this._dgvUsuarios.Rows[l_rows].Cells[0].Value.ToString(),
                                    l_pass,
this._dgvUsuarios.Rows[l_rows].Cells[1].Value.ToString(),
Convert.ToBoolean(this._dgvUsuarios.Rows[l_rows].Cells[2].Value),
Convert.ToBoolean(this._dgvUsuarios.Rows[l_rows].Cells[3].Value),
Convert.ToBoolean(this._dgvUsuarios.Rows[l_rows].Cells[4].Value),
Convert.ToBoolean(this._dgvUsuarios.Rows[l_rows].Cells[5].Value),
Convert.ToBoolean(this._dgvUsuarios.Rows[l_rows].Cells[6].Value));
                g_susu.g_efor = false;
                g_susu.Show();
                new_update_row_datagrid(false, l_rows);
}
            g_susu = null;
        }
        #endregion

        #region Botón Nuevo
privatevoid _btnNuevo_Click(object sender, EventArgs e)
{
if (g_susu == null || g_susu.IsDisposed)
              // Verifica si el Form ya fue mostrado
            {
                g_susu = newfrm_SaveUsuarios();  // Instancia del Form
g_susu.g_efor = true;
                g_susu.Show();
                new_update_row_datagrid(true, 0);
}
            g_susu = null;
        }
        #endregion

        #region Botón Actualizar
privatevoid _btnActualizar_Click(object sender, EventArgs e)
        {
int l_rows = 0;
string l_mess = string.Empty;
SqlDataReader l_dato;
string l_pass = string.Empty;

if (g_susu == null || g_susu.IsDisposed)
              // Verifica si el Form ya fue mostrado
{
                g_susu = newfrm_SaveUsuarios();
l_rows = this._dgvUsuarios.CurrentRow.Index;
              // Obtiene el índice de la fila seleccionada
```

```
if (Conectar_DB.Conexion.abrir_database(ref l_mess) == true)
// Conectar base datos
                {
                    l_dato =
Conectar_DB.Conexion.consulta_usuario_password(this._dgvUsuarios.Rows[l_rows].Ce
lls[0].Value.ToString());         // Consulta del usuario y password
while (l_dato.Read())    // Lectura de los datos consultados
{
                        l_pass = l_dato[1].ToString();
                    }
                }
else
                {
MessageBox.Show(l_mess);
}

if (Conectar_DB.Conexion.cerrar_database(ref l_mess) == false)
// Cerrar conexión base datos
                {
MessageBox.Show(l_mess);
                }

// Modo de actualización de registro
g_susu.CargarVariables(this._dgvUsuarios.Rows[l_rows].Cells[0].Value.ToString(),
                                    l_pass,
this._dgvUsuarios.Rows[l_rows].Cells[1].Value.ToString(),
Convert.ToBoolean(this._dgvUsuarios.Rows[l_rows].Cells[2].Value),
Convert.ToBoolean(this._dgvUsuarios.Rows[l_rows].Cells[3].Value),
Convert.ToBoolean(this._dgvUsuarios.Rows[l_rows].Cells[4].Value),
Convert.ToBoolean(this._dgvUsuarios.Rows[l_rows].Cells[5].Value),
Convert.ToBoolean(this._dgvUsuarios.Rows[l_rows].Cells[6].Value));
                g_susu.g_efor = false;
                g_susu.Show();
                new_update_row_datagrid(false, l_rows);
}
            g_susu = null;
        }
        #endregion

        #region Borrar
privatevoid _btnBorrar_Click(object sender, EventArgs e)
        {
string l_erro = string.Empty;
DateTime l_dtim = newDateTime();

foreach (DataGridViewRow l_dgv_DI inthis._dgvUsuarios.SelectedRows)
// Pasa por las filas seleccionadas obteniendo los datos de las mismas
            {
                l_erro =
Conectar_DB.Conexion.delete_row_table("dbo.sp_DeleteUsuarios", 1, "@p_usua",
this._dgvUsuarios.Rows[l_dgv_DI.Index].Cells[0].Value.ToString());
// Llamada DLL para borrar registro de la base de datos
}

if (l_erro != string.Empty)
            {
MessageBox.Show("Error: " + l_erro);
```

```
}
else
            {
                l_dtim = Conectar_DB.Conexion.obtener_fecha_hora_database();
Funciones.cargar_auditoria(l_dtim, "frm_Usuarios", false, false, false, false,
true);
                cargar_datagrid_con_database();
Funciones.cargar_auditoria(l_dtim, "frm_Usuarios", false, false, false, false,
true);
            }
        }
        #endregion

        #region Botón Imprimir
privatevoid _btnImprimir_Click(object sender, EventArgs e)
        {

        }
        #endregion

        #region Botón Salir
privatevoid _btnSalir_Click(object sender, EventArgs e)
        {
Funciones.cargar_auditoria(g_dtim, "frm_Usuarios", true, false, false, false,
false);
this.Close();
        }
        #endregion

        #region Funciones Varias
publicvoid cargar_datagrid_con_database()
        {
// Carga el datagrid con los valores de la tabla correspondiente
string l_mess = string.Empty;
SqlDataReader l_dato;

if (Conectar_DB.Conexion.abrir_database(ref l_mess) == true)
{
this._dgvUsuarios.Rows.Clear();       // Limpia el DataGrid
                l_dato =
Conectar_DB.Conexion.consulta_datos_tabla("sp_Usuarios");
while (l_dato.Read())
                {
this._dgvUsuarios.Rows.Add(l_dato[0].ToString(), l_dato[2].ToString(),
l_dato[3].GetHashCode(),
                                                    l_dato[4].GetHashCode(),
l_dato[5].GetHashCode(), l_dato[6].GetHashCode(),
                                                    l_dato[7].GetHashCode());
                }
            }
else
            {
MessageBox.Show(l_mess);
            }
if (Conectar_DB.Conexion.cerrar_database(ref l_mess) == false)
            {
MessageBox.Show(l_mess);
```

```
            }
        }

privatevoid new_update_row_datagrid(bool p_neup, int p_fila)
{
// Ingresar y actualizar líneas del datagrid
g_susu.Visible = false;
if (p_neup)
{
// Ingreso de nueva fila
if (g_susu.ShowDialog() == DialogResult.OK)
                {
this._dgvUsuarios.Rows.Add(g_susu.LineasDatagrid[0], g_susu.LineasDatagrid[1],
g_susu.LineasDatagrid[2],
                                                    g_susu.LineasDatagrid[3],
g_susu.LineasDatagrid[4], g_susu.LineasDatagrid[5],
                                                    g_susu.LineasDatagrid[6]);
                }
            }
else
            {
// Actualización de la fila
if (g_susu.ShowDialog() == DialogResult.OK)
                {
this._dgvUsuarios.Rows[p_fila].SetValues(g_susu.LineasDatagrid[0],
g_susu.LineasDatagrid[1], g_susu.LineasDatagrid[2],

g_susu.LineasDatagrid[3], g_susu.LineasDatagrid[4], g_susu.LineasDatagrid[5],
g_susu.LineasDatagrid[6]);
                }
            }
        }
        #endregion
    }
}



using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Data.Sql;
using System.Data.SqlClient;
using Conectar_DB;
using Funciones_Generales;

namespace CRM___WAP
{
publicpartialclassfrm_Direcciones : Form
    {
privatestaticbool g_cons;
privatestaticfrm_SaveDirecciones g_sdir;    // Objeto del Form
```

```
privatestaticDateTime g_dtim = newDateTime();
       // Fecha para auditoría

public frm_Direcciones()
        {
            InitializeComponent();

            g_dtim = Conectar_DB.Conexion.obtener_fecha_hora_database();
Funciones.cargar_auditoria(g_dtim, "frm_Direcciones", true, false, false, false,
false);

            cargar_datagrid_con_database();
        }

        #region Form Direcciones
privatevoid frm_Direcciones_Load(object sender, EventArgs e)
        {
frm_MPrincipal l_mpri = newfrm_MPrincipal();

            g_cons = false;

if (Convert.ToBoolean(l_mpri.Usuarios_Consulta[0]) ||
Convert.ToBoolean(l_mpri.Usuarios_Consulta[3]))
            {
this._btnNuevo.Enabled = false;
this._btnActualizar.Enabled = false;
this._btnBorrar.Enabled = false;
                g_cons = true;
            }
        }
        #endregion

        #region DataGrid Direcciones
privatevoid _dgvDirecciones_CellFormatting(object sender,
DataGridViewCellFormattingEventArgs e)
        {
//color a cada fila
if (this._dgvDirecciones.Rows.Count > 2)
            {
for (int i = 0; i < (this._dgvDirecciones.Rows.Count) - 1; i++)
                {
if ((i % 2) == 0)
                    {
this._dgvDirecciones.Rows[i].DefaultCellStyle.BackColor = Color.AliceBlue;
                    }
                }
            }
        }

privatevoid _dgvDirecciones_DoubleClick(object sender, EventArgs e)
        {
if (!g_cons)
            {
int l_lfie = 0;
int l_rows = 0;
string l_mess = string.Empty;
```

```
l_lfie = this._dgvDirecciones.Rows.Count - 1;
              // Obtiene el número de filas
                g_sdir = newfrm_SaveDirecciones();

// Al seleccionar una fila se envían los valores de columna
l_rows = this._dgvDirecciones.CurrentRow.Index;

if (l_lfie == this._dgvDirecciones.CurrentRow.Index)
                     // Comparación última fila y fila posicionada
                {
// Modo de nuevo registro
                    g_sdir.g_efor = true;
g_sdir.Show();
                    new_update_row_datagrid(true, 0);
}
else
                {
// Modo de actualización de registro
g_sdir.CargarVariables(this._dgvDirecciones.Rows[l_rows].Cells[0].Value.ToString
());
                    g_sdir.g_efor = false;
                    g_sdir.Show();
                    new_update_row_datagrid(false, l_rows);
}
                g_sdir = null;
            }
        }
        #endregion

        #region Botón Nuevo
privatevoid _btnNuevo_Click(object sender, EventArgs e)
{
if (g_sdir == null || g_sdir.IsDisposed)
              // Verifica si el Form ya fue mostrado
            {
                g_sdir = newfrm_SaveDirecciones();  // Instancia del Form
g_sdir.g_efor = true;
                g_sdir.Show();
                new_update_row_datagrid(true, 0);
}
            g_sdir = null;
        }
        #endregion

        #region Botón Actualizar
privatevoid _btnActualizar_Click(object sender, EventArgs e)
        {
int l_rows = 0;
string l_mess = string.Empty;

if (g_sdir == null || g_sdir.IsDisposed)
              // Verifica si el Form ya fue mostrado
            {
                g_sdir = newfrm_SaveDirecciones();
                l_rows = this._dgvDirecciones.CurrentRow.Index;
              // Obtiene el índice de la fila seleccionada
```

```
// Modo de actualización de registro

g_sdir.CargarVariables(this._dgvDirecciones.Rows[l_rows].Cells[0].Value.ToString
());
g_sdir.g_efor = false;
                g_sdir.Show();
                new_update_row_datagrid(false, l_rows);
}
            g_sdir = null;
        }
        #endregion

        #region Botón Borrar
privatevoid _btnBorrar_Click(object sender, EventArgs e)
        {
string l_erro = string.Empty;
DateTime l_dtim = newDateTime();

foreach (DataGridViewRow l_dgv_DI inthis._dgvDirecciones.SelectedRows)
// Pasa por las filas seleccionadas obtiendo los datos de las mismas
            {
                l_erro =
Conectar_DB.Conexion.delete_row_table("dbo.sp_DeleteDirecciones", 1, "@p_dire",
this._dgvDirecciones.Rows[l_dgv_DI.Index].Cells[0].Value.ToString());
// Llamada DLL para borrar registro de la base de datos
}

if (l_erro != string.Empty)
            {
MessageBox.Show("Error: " + l_erro);
}
else
            {
                l_dtim = Conectar_DB.Conexion.obtener_fecha_hora_database();
Funciones.cargar_auditoria(l_dtim, "frm_Direcciones", false, false, false,
false, true);
                cargar_datagrid_con_database();
Funciones.cargar_auditoria(l_dtim, "frm_Direcciones", false, false, false,
false, true);
            }
        }
        #endregion

        #region Botón Salir
privatevoid _btnSalir_Click(object sender, EventArgs e)
        {
Funciones.cargar_auditoria(g_dtim, "frm_Direcciones", true, false, false, false,
false);
this.Close();
        }
        #endregion

        #region Funciones Varias
publicvoid cargar_datagrid_con_database()
        {
// Carga el datagrid con los valores de la tabla correspondiente
string l_mess = string.Empty;
```

```
SqlDataReader l_dato;

if (Conectar_DB.Conexion.abrir_database(ref l_mess) == true)
{
this._dgvDirecciones.Rows.Clear();       // Limpia el DataGrid
                l_dato =
Conectar_DB.Conexion.consulta_datos_tabla("sp_Direcciones");
while (l_dato.Read())
                {
this._dgvDirecciones.Rows.Add(l_dato[0].ToString(), l_dato[1].ToString(),
l_dato[5].ToString(),
                                                        l_dato[8].ToString(),
l_dato[10].ToString());
                }
            }
else
            {
MessageBox.Show(l_mess);
            }
if (Conectar_DB.Conexion.cerrar_database(ref l_mess) == false)
{
MessageBox.Show(l_mess);
            }
        }

privatevoid new_update_row_datagrid(bool p_neup, int p_fila)
{
// Ingresar y actualizar líneas del datagrid
g_sdir.Visible = false;
if (p_neup)
{
// Ingreso de nueva fila
if (g_sdir.ShowDialog() == DialogResult.OK)
                {
this._dgvDirecciones.Rows.Add(g_sdir.LineasDatagrid[0],
g_sdir.LineasDatagrid[1]);
                }
            }
else
            {
// Actualización de la fila
if (g_sdir.ShowDialog() == DialogResult.OK)
                {
this._dgvDirecciones.Rows[p_fila].SetValues(g_sdir.LineasDatagrid[0],
g_sdir.LineasDatagrid[1]);
                }
            }
        }
        #endregion
    }
}
```

4.3.6. SESIONES DE INGRESO Y ACTUALIZACIÓN

a) Se presenta tres ejemplos diferentes de sesiones que contienen la parte de cargar y actualizar los registros en las tablas, y sus funciones independientes:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using Conectar_DB;
using Funciones_Generales;

namespace CRM___WAP
{
publicpartialclassfrm_SaveUsuarios : Form
{
// Variables Globales
publicbool g_efor;  // Estátus para Form (true nuevo, false actualizar)
privatestaticbool g_flag;     // Bandera para verificar si se guardo el valor al
hacer un cambio
privatestaticstring g_usua;
privatestaticstring g_pass;
privatestaticstring g_nomb;
privatestaticBoolean g_admi;
privatestaticBoolean g_logi;
privatestaticBoolean g_prod;
privatestaticBoolean g_cons;
privatestaticBoolean g_dage;
privateobject[] g_columnas;  // Objeto para envío de datos al DataGrid
privatestaticDateTime g_dtim = newDateTime();

public frm_SaveUsuarios()
{
            InitializeComponent();

            g_flag = false;
g_dtim = Conectar_DB.Conexion.obtener_fecha_hora_database();
}

        #region Form
privatevoid frm_SaveUsuarios_Load(object sender, EventArgs e)
{
if (g_efor == false)
            {
this._txtUsuario.Text = g_usua.Trim();
              // Cargar variable sin espacios en blanco al final
this._txtPassword.Text = g_pass.Trim();
this._txtNombre.Text = g_nomb.Trim();
```

```
this._chbAdministrador.Checked = g_admi;
this._chbLogistico.Checked = g_logi;
this._chbProduccion.Checked = g_prod;
this._chbConsulta.Checked = g_cons;
this._chbDatosGenerales.Checked = g_dage;
this._txtUsuario.Enabled = false;
            }
            enabled_disabled_checkboxs();
}
        #endregion

        #region Botón Guardar
privatevoid _btnGuardar_Click(object sender, EventArgs e)
        {
// Guardar registros de inserción o actualización
g_columnas = newobject[8];
string l_erro = string.Empty;

// Valida que se ingresen todos los datos
if (this._txtUsuario.Text != string.Empty &&this._txtPassword.Text !=
string.Empty &&this._txtNombre.Text != string.Empty &&
                (this._chbAdministrador.Checked == true ||
this._chbLogistico.Checked == true
                || this._chbProduccion.Checked == true ||
this._chbConsulta.Checked == true || this._chbDatosGenerales.Checked == true))
{
                l_erro =
Conectar_DB.Conexion.insert_update_table("dbo.sp_InsertUsuarios",

_txtUsuario.Text.Trim(),

_txtPassword.Text.Trim(),

_txtNombre.Text.Trim(),

_chbAdministrador.Checked,

_chbLogistico.Checked,

_chbProduccion.Checked,

_chbDatosGenerales.Checked,

_chbConsulta.Checked);
// Verificación de error al guardar o actualizar
if (l_erro == string.Empty)
                {
if (g_efor)
                    {
Funciones.cargar_auditoria(g_dtim, "frm_SaveUsuarios", false, false, true,
false, false);
                    }
else
                    {
Funciones.cargar_auditoria(g_dtim, "frm_SaveUsuarios", false, false, false,
true, false);
}
```

```
                    g_columnas[0] = this._txtUsuario.Text;
                    g_columnas[1] = this._txtNombre.Text;
                    g_columnas[2] = this._chbAdministrador.Checked;
                    g_columnas[3] = this._chbLogistico.Checked;
                    g_columnas[4] = this._chbProduccion.Checked;
                    g_columnas[5] = this._chbDatosGenerales.Checked;
                    g_columnas[6] = this._chbConsulta.Checked;
                }
else
                {
MessageBox.Show(l_erro);
}
            }
else
            {
MessageBox.Show("Se deben completar todos los campos.");
                l_erro = "Error";
            }

if (l_erro == string.Empty)
            {
Funciones.cargar_auditoria(g_dtim, "frm_SaveUsuarios", false, false, true,
false, false);
}
        }
        #endregion

        #region Botón Deshacer
privatevoid _btnDeshacer_Click(object sender, EventArgs e)
        {
if (g_efor)
            {
this._txtUsuario.Text = "";
this._txtPassword.Text = "";
this._txtNombre.Text = "";
this._chbAdministrador.Checked = false;
this._chbLogistico.Checked = false;
this._chbProduccion.Checked = false;
this._chbConsulta.Checked = false;
this._chbDatosGenerales.Checked = false;
            }
else
            {
this._txtPassword.Text = g_pass;
this._txtNombre.Text = g_nomb;
this._chbAdministrador.Checked = g_admi;
this._chbLogistico.Checked = g_logi;
this._chbProduccion.Checked = g_prod;
this._chbConsulta.Checked = g_cons;
this._chbDatosGenerales.Checked = g_dage;
            }
        }
        #endregion

        #region Botón Salir
privatevoid _btnSalir_Click(object sender, EventArgs e)
```

```
        {
if ((this._txtUsuario.Text != string.Empty || this._txtPassword.Text !=
string.Empty || this._txtNombre.Text != string.Empty ||
                (this._chbAdministrador.Checked == true ||
this._chbLogistico.Checked == true || this._chbProduccion.Checked == true ||
this._chbConsulta.Checked == true || this._chbDatosGenerales.Checked == true))
&& g_flag)
{
if (DialogResult.Yes == MessageBox.Show("Desea salir sin guardar?", "Salir de
Usuarios", MessageBoxButtons.YesNo))
{
this.Close();
                }
            }
else
            {
this.Close();
            }
        }
        #endregion

        #region Text Usuario
privatevoid _txtUsuario_KeyPress(object sender, KeyPressEventArgs e)
{
// Valida al presionar al tecla que solo ingrese caractéres y backspace
if (Char.IsLetter(e.KeyChar) || (e.KeyChar == (char)Keys.Back))
{
                e.Handled = false;      // Si permitido
            }
else
            {
                e.Handled = true;       // No permitido
            }
        }
        #endregion

        #region Text Password
privatevoid _txtPassword_KeyPress(object sender, KeyPressEventArgs e)
{
// Valida al presionar el tecla que solo ingrese varios caractéres
if (Char.IsLetter(e.KeyChar) || (Char.IsDigit(e.KeyChar)))
{
                e.Handled = false;      // Si permitido
            }
else
            {
                e.Handled = true;       // No permitido
            }
        }
        #endregion

        #region Text Nombre
privatevoid _txtNombre_KeyPress(object sender, KeyPressEventArgs e)
{
// Valida al presionar al tecla que solo ingrese letras y backspace
if (Char.IsLetter(e.KeyChar) || (e.KeyChar == (char)Keys.Back) ||
!(Char.IsDigit(e.KeyChar)))
```

```
            {
                e.Handled = false;      // Si permitido
            }
else
            {
                e.Handled = true;       // No permitido
            }
        }
        #endregion

        #region CheckBox Administrador
privatevoid _chbAdministrador_CheckedChanged(object sender, EventArgs e)
{
// Al cambiar el checkbox se habilita la bandera
if (g_admi != this._chbAdministrador.Checked)
            {
                g_flag = true;
            }
            enabled_disabled_checkboxs();
        }
        #endregion

        #region CheckBox Logístico
privatevoid _chbLogistico_CheckedChanged(object sender, EventArgs e)
{
// Al cambiar el checkbox se habilita la bandera
if (g_logi != this._chbLogistico.Checked)
            {
                g_flag = true;
            }
            enabled_disabled_checkboxs();
        }
        #endregion

        #region CheckBox Producción
privatevoid _chbProduccion_CheckedChanged(object sender, EventArgs e)
{
// Al cambiar el checkbox se habilita la bandera
if (g_prod != this._chbProduccion.Checked)
            {
                g_flag = true;
            }
            enabled_disabled_checkboxs();
        }
        #endregion

        #region CheckBox Consulta
privatevoid _chbConsulta_CheckedChanged(object sender, EventArgs e)
{
// Al cambiar el checkbox se habilita la bandera
if (g_cons != this._chbConsulta.Checked)
            {
                g_flag = true;
            }
            enabled_disabled_checkboxs();
        }
        #endregion
```

```
        #region CheckBox Datos Generales
privatevoid _chbDatosGenerales_CheckedChanged(object sender, EventArgs e)
{
// Al cambiar el checkbox se habilita la bandera
if (g_dage != this._chbConsulta.Checked)
            {
                g_flag = true;
            }
            enabled_disabled_checkboxs();
}
        #endregion

        #region Funciones Varias
publicvoid CargarVariables(paramsobject[] p_parm)
       // Función con n cantidad de parámetros
        {
// Función para cargar las variables
            g_usua = p_parm[0].ToString();
g_pass = p_parm[1].ToString();
            g_nomb = p_parm[2].ToString();
            g_admi = Convert.ToBoolean(p_parm[3]);
            g_logi = Convert.ToBoolean(p_parm[4]);
            g_prod = Convert.ToBoolean(p_parm[5]);
            g_dage = Convert.ToBoolean(p_parm[6]);
            g_cons = Convert.ToBoolean(p_parm[7]);
        }

publicvoid enabled_disabled_checkboxs()
        {
// Habilita o deshabilita los checkbox
if (this._chbAdministrador.Checked == true)
            {
this._chbLogistico.Enabled = false;
this._chbProduccion.Enabled = false;
this._chbDatosGenerales.Enabled = false;
this._chbConsulta.Enabled = false;
            }
else
            {
this._chbLogistico.Enabled = true;
this._chbProduccion.Enabled = true;
this._chbDatosGenerales.Enabled = true;
this._chbConsulta.Enabled = true;
if (this._chbLogistico.Checked == true || this._chbProduccion.Checked == true
                    || this._chbDatosGenerales.Checked == true)
                {
this._chbAdministrador.Enabled = false;
this._chbConsulta.Enabled = false;
                }
else
                {
this._chbAdministrador.Enabled = true;
this._chbConsulta.Enabled = true;

if (this._chbConsulta.Checked == true)
                    {
```

```
this._chbAdministrador.Enabled = false;
this._chbLogistico.Enabled = false;
this._chbProduccion.Enabled = false;
this._chbDatosGenerales.Enabled = false;
                    }
else
                    {
this._chbAdministrador.Enabled = true;
this._chbLogistico.Enabled = true;
this._chbProduccion.Enabled = true;
this._chbDatosGenerales.Enabled = true;
                    }
                }
            }
this.Refresh();
        }

publicobject[] LineasDatagrid
        {
get
            {
return (g_columnas);
}
        }
        #endregion
    }
}


using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Data.Sql;
using System.Data.SqlClient;
using Conectar_DB;
using Funciones_Generales;

namespace CRM___WAP
{
publicpartialclassfrm_SaveCiudad : Form
{
publicbool g_efor;  // Estátus para Form (true nuevo, false actualizar)
privatestaticbool g_flag;     // Bandera para verificar si se guarda el valor al
hacer un cambio
privatestaticstring g_pais;
privatestaticstring g_epro;
privatestaticstring g_ciud;
privatestaticstring g_desc;
privateobject[] g_columnas;   // Objeto para envío de datos al DataGrid
privatestaticDateTime g_dtim = newDateTime();
frm_ZoomPais g_zpai;
frm_ZoomEstadosP g_zepr;
```

```
public frm_SaveCiudad()
        {
            InitializeComponent();

            g_flag = false;
g_dtim = Conectar_DB.Conexion.obtener_fecha_hora_database();
}

        #region From Save Ciudad
privatevoid frm_SaveCiudad_Load(object sender, EventArgs e)
{
if (g_efor == false)
            {
this._txtPais.Text = g_pais.Trim();
              // Cargar variable sin espacios en blanco al final
this._txtEstadoP.Text = g_epro.Trim();
this._txtCiudad.Text = g_ciud.Trim();
this._txtDescripcion.Text = g_desc.Trim();
}
        }
        #endregion

        #region Botón Guardar
privatevoid _btnGuardar_Click(object sender, EventArgs e)
        {
// Guardar registros de inserción o actualización
g_columnas = newobject[6];
string l_erro = string.Empty;

// Valida que se ingresen todos los datos
if (this._txtPais.Text != string.Empty &&this._txtDescripcion.Text !=
string.Empty &&
this._txtEstadoP.Text != string.Empty)
{
                l_erro =
Conectar_DB.Conexion.insert_update_estados_provincias("dbo.sp_InsertCiudades",

_txtPais.Text.Trim(),

_txtEstadoP.Text.Trim(),

_txtCiudad.Text.Trim(),

_txtDescripcion.Text.Trim());
// Verificación de error al guardar o actualizar
if (l_erro == string.Empty)
                {
if (g_efor)
                    {
Funciones.cargar_auditoria(g_dtim, "frm_SaveCiudad", false, false, true, false,
false);
                    }
else
                    {
Funciones.cargar_auditoria(g_dtim, "frm_SaveCiudad", false, false, false, true,
false);
```

```
                    }

                    g_columnas[0] = this._txtPais.Text;
                    g_columnas[1] = this._lblPais.Text;
                    g_columnas[2] = this._txtEstadoP.Text;
                    g_columnas[3] = this._lblDEProvincia.Text;
                    g_columnas[4] = this._txtCiudad.Text;
                    g_columnas[5] = this._txtDescripcion.Text;
                }
else
                {
MessageBox.Show(l_erro);
                }
            }
else
            {
MessageBox.Show("Se deben completar todos los campos.");
                l_erro = "Error";
            }

if (l_erro == string.Empty)
            {
Funciones.cargar_auditoria(g_dtim, "frm_SaveCiudad", false, false, true, false,
false);
}
        }
        #endregion

        #region Botón Deshacer
privatevoid _btnDeshacer_Click(object sender, EventArgs e)
        {
if (g_efor)
            {
this._txtPais.Text = "";
this._txtEstadoP.Text = "";
this._txtCiudad.Text = "";
this._txtDescripcion.Text = "";
            }
else
            {
this._txtPais.Text = g_pais;
this._txtEstadoP.Text = g_epro;
this._txtCiudad.Text = g_ciud;
this._txtDescripcion.Text = g_desc;
            }
        }
        #endregion

        #region Botón Zoom País
privatevoid _btnZoomPais_Click(object sender, EventArgs e)
{
if (g_zpai == null || g_zpai.IsDisposed)
              // Verifica si el Form ya fue mostrado
{
                g_zpai = newfrm_ZoomPais();
g_zpai.Show();
g_zpai.Visible = false;
```

```
                    // Necesario antes de ser llamado como díalogo
if (g_zpai.ShowDialog() == DialogResult.OK)
                {
this._txtPais.Text = g_zpai.RetornoPais[0].ToString();
this._lblDPais.Text = g_zpai.RetornoPais[1].ToString();
}
            }
            g_zpai = null;
        }
        #endregion

        #region Botón Estados Provincias
privatevoid _btnZoomEstadosP_Click(object sender, EventArgs e)
{
if (g_zepr == null || g_zepr.IsDisposed)
       // Verifica si el Form ya fue mostrado
{
                g_zepr = newfrm_ZoomEstadosP();
g_zepr.Show();
                g_zepr.Visible = false;
              // Necesario antes de ser llamado como diálogo
if (g_zepr.ShowDialog() == DialogResult.OK)
                {
this._txtEstadoP.Text = g_zepr.RetornoPais[0].ToString();
this._lblDEProvincia.Text = g_zepr.RetornoPais[1].ToString();
}
            }
            g_zpai = null;
        }
        #endregion

        #region Botón Salir
privatevoid _btnSalir_Click(object sender, EventArgs e)
        {
if ((this._txtPais.Text != string.Empty || this._txtEstadoP.Text != string.Empty
||
this._txtCiudad.Text != string.Empty || this._txtDescripcion.Text !=
string.Empty) && g_flag)
{
if (DialogResult.Yes == MessageBox.Show("Desea salir sin guardar?", "Salir de
Usuarios", MessageBoxButtons.YesNo))
{
this.Close();
                }
            }
else
            {
this.Close();
            }
        }
        #endregion

        #region Text País
privatevoid _txtPais_KeyPress(object sender, KeyPressEventArgs e)
{
// Valida al presionar al tecla que solo ingrese caractéres y backspace
if (Char.IsLetter(e.KeyChar) || (e.KeyChar == (char)Keys.Back))
```

```
{
                e.Handled = false;      // Si permitido
            }
else
            {
                e.Handled = true;       // No permitido
            }
        }

privatevoid _txtPais_Validated(object sender, EventArgs e)
{
// Se ejecuta después de ingresar el dato
this._txtPais.Text = this._txtPais.Text.ToUpper();
        }

privatevoid _txtPais_Validating(object sender, CancelEventArgs e)
{
// Se ejecuta después de ingresar el dato
if (this._txtPais.Text != "")
            {
SqlDataReader l_dato;
string l_mess = string.Empty;

this._erroValidar.Clear();       // Límpia los mensajes de error

                l_dato =
Conectar_DB.Conexion.valida_existencia_pais(this._txtPais.Text);

if (!l_dato.HasRows)
                {
// Usuario y Password incorrectos
this._erroValidar.SetError(this._txtPais, "El País ingresado no existe!.");
                }

if (Conectar_DB.Conexion.cerrar_database(ref l_mess) == false)
// Cerrar conexión base datos
{
MessageBox.Show(l_mess);
                }
            }
        }
        #endregion

        #region Text Estado o Provincia
privatevoid _txtEstadoP_KeyPress(object sender, KeyPressEventArgs e)
{
// Valida al presionar al tecla que solo ingrese caractéres y backspace
if (Char.IsLetter(e.KeyChar) || (e.KeyChar == (char)Keys.Back))
{
                e.Handled = false;      // Si permitido
            }
else
            {
                e.Handled = true;       // No permitido
            }
        }
```

```
privatevoid _txtEstadoP_Validated(object sender, EventArgs e)
{
// Se ejecuta después de ingresar el dato
this._txtEstadoP.Text = this._txtEstadoP.Text.ToUpper();
        }

privatevoid _txtEstadoP_Validating(object sender, CancelEventArgs e)
{
// Se ejecuta después de ingresar el dato
if (this._txtEstadoP.Text != "")
            {
SqlDataReader l_dato;
string l_mess = string.Empty;

this._erroValidar.Clear();       // Límpia los mensajes de error

                l_dato =
Conectar_DB.Conexion.valida_existencia_pais(this._txtEstadoP.Text);

if (!l_dato.HasRows)
                {
// Usuario y Password incorrectos
this._erroValidar.SetError(this._txtPais, "El Estado o Provincia ingresado no
existe!.");
                }

if (Conectar_DB.Conexion.cerrar_database(ref l_mess) == false)
// Cerrar conexión base datos
{
MessageBox.Show(l_mess);
                }
            }
        }
        #endregion

        #region Text Ciudad
privatevoid _txtCiudad_KeyPress(object sender, KeyPressEventArgs e)
{
// Valida al presionar al tecla que solo ingrese caractéres y backspace
if (Char.IsLetter(e.KeyChar) || (e.KeyChar == (char)Keys.Back))
{
                e.Handled = false;      // Si permitido
            }
else
            {
                e.Handled = true;       // No permitido
            }
        }

privatevoid _txtCiudad_Validated(object sender, EventArgs e)
{
// Se ejecuta después de ingresar el dato
this._txtCiudad.Text = this._txtCiudad.Text.ToUpper();
}
        #endregion

        #region Text Descripción
```

```
privatevoid _txtDescripcion_KeyPress(object sender, KeyPressEventArgs e)
        {
if (Char.IsDigit(e.KeyChar))
{
                e.Handled = true;       // No permito
            }
else
            {
                e.Handled = false;      // Si permitido
            }
        }

privatevoid _txtDescripcion_TextChanged(object sender, EventArgs e)
{
// Al cambiar el texto se habilita la bandera
if (g_desc != this._txtDescripcion.Text)
{
                g_flag = true;
            }
        }

privatevoid _txtDescripcion_Validated(object sender, EventArgs e)
{
// Se ejecuta después de ingresar el dato
if (this._txtDescripcion.Text != "")
            {
this._txtDescripcion.Text = this._txtDescripcion.Text.Substring(0, 1).ToUpper()
+
this._txtDescripcion.Text.Substring(1);
            }
        }
        #endregion

        #region Funciones Varias
publicvoid CargarVariables(paramsobject[] p_parm)
       // Función con n cantidad de parámetros
        {
// Función para cargar las variables
            g_pais = p_parm[0].ToString();
g_epro = p_parm[1].ToString();
            g_ciud = p_parm[2].ToString();
            g_desc = p_parm[3].ToString();
        }

publicobject[] LineasDatagrid
        {
get
            {
return (g_columnas);
}
        }
        #endregion
    }
}

using System;
using System.Collections.Generic;
```

```
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Data.Sql;
using System.Data.SqlClient;
using Conectar_DB;
using Funciones_Generales;

namespace CRM___WAP
{
publicpartialclassfrm_SaveEstadosP : Form
{
publicbool g_efor;  // Estatus para Form (true nuevo, false actualizar)
privatestaticbool g_flag;     // Bandera para verificar si se guardó el valor al
hacer un cambio
privatestaticstring g_pais;
privatestaticstring g_epro;
privatestaticstring g_desc;
privateobject[] g_columnas;   // Objeto para envío de datos al DataGrid
privatestaticDateTime g_dtim = newDateTime();
frm_ZoomPais g_zpai;

public frm_SaveEstadosP()
{
            InitializeComponent();

            g_flag = false;
g_dtim = Conectar_DB.Conexion.obtener_fecha_hora_database();
}

        #region Form Estado Provincia
privatevoid frm_SaveEstadosP_Load(object sender, EventArgs e)
{
if (g_efor == false)
            {
this._txtPais.Text = g_pais.Trim();
              // Cargar variable sin espacios en blanco al final
this._txtEstadoP.Text = g_epro.Trim();
this._txtDescripcion.Text = g_desc.Trim();
}
        }
        #endregion

        #region Botón Guardar
privatevoid _btnGuardar_Click(object sender, EventArgs e)
        {
// Guardar registros de inserción o actualización
g_columnas = newobject[4];
string l_erro = string.Empty;

// Valida que se ingresen todos los datos
if (this._txtPais.Text != string.Empty &&this._txtDescripcion.Text !=
string.Empty &&
this._txtEstadoP.Text != string.Empty)
```

```
{
                l_erro =
Conectar_DB.Conexion.insert_update_estados_provincias("dbo.sp_InsertEProvincias"
,

_txtPais.Text.Trim(),

_txtEstadoP.Text.Trim(),

_txtDescripcion.Text.Trim());
// Verificación de error al guardar o actualizar
if (l_erro == string.Empty)
                {
if (g_efor)
                    {
Funciones.cargar_auditoria(g_dtim, "frm_SaveEstadosP", false, false, true,
false, false);
                    }
else
                    {
Funciones.cargar_auditoria(g_dtim, "frm_SaveEstadosP", false, false, false,
true, false);
                    }

                    g_columnas[0] = this._txtPais.Text;
                    g_columnas[1] = this._lblPais.Text;
                    g_columnas[2] = this._txtEstadoP.Text;
                    g_columnas[3] = this._txtDescripcion.Text;
                }
else
                {
MessageBox.Show(l_erro);
                }
            }
else
            {
MessageBox.Show("Se deben completar todos los campos.");
                l_erro = "Error";
            }

if (l_erro == string.Empty)
            {
Funciones.cargar_auditoria(g_dtim, "frm_SaveEstadoSP", false, false, true,
false, false);
}
        }
        #endregion

        #region Botón Deshacer
privatevoid _btnDeshacer_Click(object sender, EventArgs e)
        {
if (g_efor)
            {
this._txtPais.Text = "";
this._txtEstadoP.Text = "";
this._txtDescripcion.Text = "";
            }
```

```
else
            {
this._txtPais.Text = g_pais;
this._txtEstadoP.Text = g_epro;
this._txtDescripcion.Text = g_desc;
            }
        }
        #endregion

        #region Botón Zoom País
privatevoid _btnZoomPais_Click(object sender, EventArgs e)
{
if (g_zpai == null || g_zpai.IsDisposed)
             // Verifica si el Form ya fue mostrado
{
                g_zpai = newfrm_ZoomPais();
g_zpai.Show();
g_zpai.Visible = false;
             // Necesario antes de ser llamado como diálogo
if (g_zpai.ShowDialog() == DialogResult.OK)
                {
this._txtPais.Text = g_zpai.RetornoPais[0].ToString();
this._lblDPais.Text = g_zpai.RetornoPais[1].ToString();
}
            }
            g_zpai = null;
        }
        #endregion

        #region Botón Salir
privatevoid _btnSalir_Click(object sender, EventArgs e)
        {
if ((this._txtPais.Text != string.Empty || this._txtEstadoP.Text != string.Empty
||
this._txtDescripcion.Text != string.Empty) && g_flag)
{
if (DialogResult.Yes == MessageBox.Show("Desea salir sin guardar?", "Salir de
Usuarios", MessageBoxButtons.YesNo))
{
this.Close();
                }
            }
else
            {
this.Close();
            }
        }
        #endregion

        #region Text Pais
privatevoid _txtPais_KeyPress(object sender, KeyPressEventArgs e)
{
// Valida al presionar al tecla que solo ingrese caractéres y backspace
if (Char.IsLetter(e.KeyChar) || (e.KeyChar == (char)Keys.Back))
{
                e.Handled = false;      // Si permitido
            }
```

```
else
            {
                e.Handled = true;       // No permitido
            }
        }

privatevoid _txtPais_Validated(object sender, EventArgs e)
{
// Se ejecuta después de ingresar el dato
this._txtPais.Text = this._txtPais.Text.ToUpper();
        }

privatevoid _txtPais_Validating(object sender, CancelEventArgs e)
{
// Se ejecuta después de ingresar el dato
if (this._txtPais.Text != "")
            {
SqlDataReader l_dato;
string l_mess = string.Empty;

this._erroPais.Clear();          // Limpia los mensajes de error

                l_dato =
Conectar_DB.Conexion.valida_existencia_pais(this._txtPais.Text);

if (!l_dato.HasRows)
                {
// Usuario y Password incorrectos
this._erroPais.SetError(this._txtPais, "El País ingresado no existe!.");
                }

if (Conectar_DB.Conexion.cerrar_database(ref l_mess) == false)
// Cerrar conexión base datos
{
MessageBox.Show(l_mess);
                }
            }
        }
        #endregion

        #region Text Estado Provincia
privatevoid _txtEstadoP_KeyPress(object sender, KeyPressEventArgs e)
{
// Valida al presionar al tecla que solo ingrese caractéres y backspace
if (Char.IsLetter(e.KeyChar) || (e.KeyChar == (char)Keys.Back))
{
                e.Handled = false;      // Si permitido
            }
else
            {
                e.Handled = true;       // No permitido
            }
        }

privatevoid _txtEstadoP_Validated(object sender, EventArgs e)
{
// Se ejecuta después de ingresar el dato
```

```
this._txtEstadoP.Text = this._txtEstadoP.Text.ToUpper();
        }
        #endregion

        #region Text Descripción
privatevoid _txtDescripcion_KeyPress(object sender, KeyPressEventArgs e)
        {
if (Char.IsDigit(e.KeyChar))
            {
                e.Handled = true;       // No permito
            }
else
            {
                e.Handled = false;      // Si permitido
            }
        }

privatevoid _txtDescripcion_TextChanged(object sender, EventArgs e)
{
// Al cambiar el texto se habilita la bandera
if (g_desc != this._txtDescripcion.Text)
            {
                g_flag = true;
            }
        }

privatevoid _txtDescripcion_Validated(object sender, EventArgs e)
{
// Se ejecuta después de ingresar el dato
if (this._txtDescripcion.Text != "")
            {
this._txtDescripcion.Text = this._txtDescripcion.Text.Substring(0, 1).ToUpper()
+
this._txtDescripcion.Text.Substring(1);
            }
        }
        #endregion

        #region Funciones Varias
publicvoid CargarVariables(paramsobject[] p_parm)
       // Función con n cantidad de parámetros
        {
// Función para cargar las variables
            g_pais = p_parm[0].ToString();
g_epro = p_parm[1].ToString();
            g_desc = p_parm[2].ToString();
        }

publicobject[] LineasDatagrid
        {
get
            {
return (g_columnas);
}
        }
        #endregion
    }
```

```
}
```

4.3.7. SESIONES DE ZOOM

a) Código de las sesiones que funcionan como zoom, estas son cargadas con los registros de la tabla, para una vez seleccionada una de las filas requeridas, devuelva por referencia el campo solicitado:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Data.Sql;
using System.Data.SqlClient;
using Conectar_DB;

namespace CRM___WAP
{
publicpartialclassfrm_ZoomEstadosP : Form
    {
privateobject[] g_columnas;

public frm_ZoomEstadosP()
{
            InitializeComponent();

            cargar_datagrid_con_database();
        }

        #region DataGrid Estados Provincias
privatevoid _dgvEstadosP_CellFormatting(object sender,
DataGridViewCellFormattingEventArgs e)
        {
//color a cada fila
if (this._dgvEstadosP.Rows.Count > 2)
            {
for (int i = 0; i < (this._dgvEstadosP.Rows.Count) - 1; i++)
                {
if ((i % 2) == 0)
                    {
this._dgvEstadosP.Rows[i].DefaultCellStyle.BackColor = Color.AliceBlue;
                    }
                }
            }
        }
```

```
privatevoid _dgvEstadosP_Click(object sender, EventArgs e)
        {
int l_rows = 0;
            g_columnas = newobject[2];

// Al seleccionar una fila se envían los valores de columna
l_rows = this._dgvEstadosP.CurrentRow.Index;

            g_columnas[0] =
this._dgvEstadosP.Rows[l_rows].Cells[2].Value.ToString();
            g_columnas[1] =
this._dgvEstadosP.Rows[l_rows].Cells[3].Value.ToString();
        }
        #endregion

        #region Botón Selección
privatevoid _btnSeleccion_Click(object sender, EventArgs e)
        {
this.Close();
        }
        #endregion

        #region Funciones Varias
publicvoid cargar_datagrid_con_database()
{
// Carga el datagrid con los valores de la tabla correspondiente
string l_mess = string.Empty;
SqlDataReader l_dato;

if (Conectar_DB.Conexion.abrir_database(ref l_mess) == true)
{
this._dgvEstadosP.Rows.Clear();       // Limpia el DataGrid
                l_dato =
Conectar_DB.Conexion.consulta_datos_tabla("sp_EstadosProvincias");
while (l_dato.Read())
                {
this._dgvEstadosP.Rows.Add(l_dato[0].ToString(), l_dato[1].ToString(),
                                                    l_dato[2].ToString(),
l_dato[3].ToString());
                }
            }
else
            {
MessageBox.Show(l_mess);
            }
if (Conectar_DB.Conexion.cerrar_database(ref l_mess) == false)
            {
MessageBox.Show(l_mess);
            }
        }

publicobject[] RetornoPais
        {
get
            {
return (g_columnas);
            }
```

```
        }
        #endregion
    }
}
```

4.4. CÓDIGO FUENTE DEL MÓDULO WAP

A continuación se explica el código utilizado, para el manejo del módulo de ventas, al que se accederá por medio de celulares que posean salida a internet. Este contiene dos partes, una que es en HTML, y la otra .cs de ASP C#.

4.4.1. PANTALLA DE BIENVENIDA

a) Para el ingreso al sistema por medio del celular, se tiene la verificación de que el usuario, se encuentre registrado en el sistema:

a.1) **Código HTML**

```
<%@PageLanguage="C#"AutoEventWireup="true"Inherits="Prueba.Default1"Codebehind="
Bienvenida.aspx.cs"%>

<%@RegisterAssembly="System.Web.Mobile, Version=4.0.0.0, Culture=neutral,
PublicKeyToken=b03f5f7f11d50a3a"
Namespace="System.Web.UI.MobileControls"TagPrefix="cc1"%>

<htmlxmlns="http://www.w3.org/1999/xhtml">
<HEAD>
<metaname="GENERATOR"content="Microsoft Visual Studio .NET 7.1">
<metaname="CODE_LANGUAGE"content="Visual Basic .NET 7.1">
<metaname="vs_targetSchema"content="http:"//schemas.microsoft.com/Mobile/Page">
</HEAD>
<bodyXmlns:mobile="http:"//schemas.microsoft.com/Mobile/WebForm">
<mobile:Formid="Form1"runat="server"Font-Name="Arial Black"Font-Size=" Normal
"Alignment="Center"><BR><BR>LA PULPITA<BR><BR>
<cc1:LabelID="lblLogin"runat="server">Usuario</cc1:Label><br/>
<cc1:TextBoxID="txtLogin"runat="server"></cc1:TextBox>
<cc1:LabelID="lblPassword"runat="server">Password</cc1:Label>
<cc1:TextBoxID="txtPass"runat="server"Password="True"></cc1:TextBox><br/>
<cc1:CommandID="cmdEnviar"runat="server">Ingresar</cc1:Command>
</mobile:Form>
</body>
```

```
</html>
```

a.2) **Código .cs**

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Web;
using System.Web.Mobile;
using System.Web.SessionState;
using System.Web.UI;
using System.Web.UI.MobileControls;
using System.Web.UI.WebControls;
using System.Web.UI.HtmlControls;
using System.Diagnostics;
using System.Data.OleDb;
using System.Data.SqlClient;

namespace Prueba
{
publicpartialclassDefault1 : System.Web.UI.MobileControls.MobilePage
    {
        #region " Web Form Designer Generated Code "

//This call is required by the Web Form Designer.
        [System.Diagnostics.DebuggerStepThrough()]
privatevoid InitializeComponent()
        {
            cmdEnviar.Click += newEventHandler(cmdEnviar_Click);
        }

void cmdEnviar_Click(object sender, EventArgs e)
        {
            Response.Redirect("Resultado.aspx?login=" + txtLogin.Text + "&pass="
+ txtPass.Text);
        }
protected System.Web.UI.MobileControls.Link Ingresar;
protected System.Web.UI.MobileControls.List List1;
protected System.Web.UI.MobileControls.Form Form1;

privatevoid Page_Init(System.Object sender, System.EventArgs e)
        {
//CODEGEN: This method call is required by the Web Form Designer
//Do not modify it using the code editor.
            InitializeComponent();
        }

        #endregion
protectedvoid Page_Load(object sender, EventArgs e)
        {
```

```
        }
public Default1()
    {
            Load += Page_Load;
        Init += Page_Init;
    }
    }
}
```

4.4.2. PANTALLA DE ERROR

a) Esta es una pantalla que muestra error si el usuario que intenta logearse al sistema, no existe, o ingreso un password erroneo:

a.1) **Código HTML**

```
<%@PageLanguage="C#"AutoEventWireup="true"Inherits="Prueba.Error"Codebehind="Err
or.aspx.cs"%>

<%@RegisterAssembly="System.Web.Mobile, Version=4.0.0.0, Culture=neutral,
PublicKeyToken=b03f5f7f11d50a3a"
Namespace="System.Web.UI.MobileControls"TagPrefix="cc1"%>

<htmlxmlns="http://www.w3.org/1999/xhtml">
<body>
<mobile:Formid="Form1"runat="server"Font-Name="Arial Black"Font-Size=" Normal
"Alignment="Center">
<mobile:LabelID="lblBienvenido"Runat="server"></mobile:Label>
<cc1:LinkID="lnkMenu"runat=serverVisible="false"NavigateUrl="Menu.aspx">Menu
principal</cc1:Link>
</mobile:Form>
</body>
</html>
```

a.2) **Código .cs**

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Web;
using System.Web.Mobile;
using System.Web.SessionState;
```

```
using System.Web.UI;
using System.Web.UI.MobileControls;
using System.Web.UI.WebControls;
using System.Web.UI.HtmlControls;

namespace Prueba
{
publicpartialclassError : System.Web.UI.MobileControls.MobilePage
    {
protectedvoid Page_Load(object sender, EventArgs e)
        {
            lnkMenu.Visible = false;
String[] strQuery = QueryStringText.Split('=');
            lblBienvenido.Text = strQuery[1].Replace('+',' ');
if (lblBienvenido.Text == "Pedido realizado con exito")
lnkMenu.Visible = true;
        }
    }
}
```

4.4.3. PANTALLA DEL MENÚ

a) Contiene las opciones a las que se pueden acceder, como la consulta de stock, y la toma del pedido:

a.1) **Código HTML**

```
<%@PageLanguage="C#"AutoEventWireup="true"Inherits="Prueba.Menu"Codebehind="Menu
.aspx.cs"%>

<%@RegisterAssembly="System.Web.Mobile, Version=4.0.0.0, Culture=neutral,
PublicKeyToken=b03f5f7f11d50a3a"
Namespace="System.Web.UI.MobileControls"TagPrefix="cc1"%>

<htmlxmlns="http://www.w3.org/1999/xhtml">
<body>
<mobile:Formid="Form1"runat="server"Font-Name="Arial Black"Font-Size=" Normal
"Alignment="Center">
<mobile:LabelID="lblBienvenido"Runat="server">Bienvenido al
sistema</mobile:Label><BR><BR>
<mobile:Linkid="lnkConsultarStock"Font-
Size="Small"runat="server"NavigateUrl="Stock.aspx">Consultar
stock</mobile:Link><BR>
<mobile:Linkid="lnkIngresarPedido"Font-
Size="Small"runat="server"NavigateUrl="Pedido.aspx">Ingresar
pedido</mobile:Link><BR>
</mobile:Form>
</body>
</html>
```

4.4.4. PANTALLA DE CONSULTA DE STOCK

a) Aquí se consulta el stock disponible del artículo, independientemente de la ubicación y almacén en el que se encuentren:

a.1) **Código HTML**

```
<%@PageLanguage="C#"AutoEventWireup="true"Inherits="Prueba.Stock"Codebehind="Sto
ck.aspx.cs"%>

<htmlxmlns="http://www.w3.org/1999/xhtml">
<bodyXmlns:mobile="http:"//schemas.microsoft.com/Mobile/WebForm">
<mobile:Formid="Form1"runat="server"Font-Name="Arial Black"Font-Size=" Normal
"Alignment="Center">
<mobile:ListID="lstStock"Runat="server"></mobile:List>
</mobile:Form>
</body>
</html>
```

a.2) **Código .cs**

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Web;
using System.Web.Mobile;
using System.Web.SessionState;
using System.Web.UI;
using System.Web.UI.MobileControls;
using System.Web.UI.WebControls;
using System.Web.UI.HtmlControls;
using System.Data.OleDb;

namespace Prueba
{
publicpartialclassStock : System.Web.UI.MobileControls.MobilePage
    {
        #region " Web Form Designer Generated Code "

//This call is required by the Web Form Designer.
        [System.Diagnostics.DebuggerStepThrough()]
privatevoid InitializeComponent()
        {
```

```
        }

privatevoid Page_Init(System.Object sender, System.EventArgs e)
        {
//CODEGEN: This method call is required by the Web Form Designer
//Do not modify it using the code editor.
            InitializeComponent();
        }

        #endregion
protectedvoid Page_Load(object sender, EventArgs e)
        {
OleDbConnection cn = null;
            System.Data.DataSet ds = null;
OleDbDataAdapter da = null;
string strConexion = null;
string Ssql = null;

            strConexion = @"Provider=SQLOLEDB;Data Source=BORIS-
PC\SQLEXPRESS;Integrated Security=SSPI;Initial Catalog=Cines";

//Abrimos la conexión
            cn = newOleDbConnection(strConexion);
cn.Open();

            ds = new System.Data.DataSet();
            Ssql = "SELECT * FROM producto";
            da = newOleDbDataAdapter(Ssql, cn);

OleDbCommandBuilder cmdBuilder = newOleDbCommandBuilder(da);
//Cargamos el DataSet
            da.Fill(ds, "producto");

DataRow fila = null;
            lstStock.Items.Clear();
foreach (DataRow fila_loopVariable in ds.Tables["producto"].Rows)
            {
                fila = fila_loopVariable;
                lstStock.Items.Add(fila["producto"] + " - " + fila["stock"]);
            }
        }
public Stock()
        {
            Load += Page_Load;
Init += Page_Init;
        }
    }
}
```

4.4.5. PANTALLA DE INGRESO DE PEDIDO

a) Se presenta una sesión de ingreso de datos para que el usuario registre un pedido ordenado por el cliente:

a.1) **Código HTML**

```
<%@PageLanguage="C#"AutoEventWireup="true"Inherits="Prueba.Pedido"Codebehind="Pe
dido.aspx.cs"%>

<%@RegisterAssembly="System.Web.Mobile, Version=4.0.0.0, Culture=neutral,
PublicKeyToken=b03f5f7f11d50a3a"
Namespace="System.Web.UI.MobileControls"TagPrefix="cc1"%>

<htmlxmlns="http://www.w3.org/1999/xhtml">
<body>
<mobile:Formid="Form1"runat="server">
<table>
<tr>
<td>
<cc1:LabelID="lblIdentificacion"runat="server">Identificacion</cc1:Label>
</td>
<td>
<cc1:TextBoxID="txtIdentificacion"runat="server"></cc1:TextBox>
</td>
</tr>
<tr>
<td>
<cc1:LabelID="lblProducto"runat="server">Producto</cc1:Label>
</td>
<td>
<cc1:SelectionListID="sltProducto"runat="server"></cc1:SelectionList>
</td>
</tr>
<tr>
<td>
<cc1:LabelID="lblCantidad"runat="server">Cantidad</cc1:Label>
</td>
<td>
<cc1:TextBoxID="txtCantidad"runat="server"></cc1:TextBox>
</td>
</tr>
<tr>
<td>
<cc1:CommandID="cmdPedido"runat="server">Aceptar</cc1:Command>
</td>
</tr>
</table>


</mobile:Form>
```

```
</body>
       </html>
```

a.2) **Código .cs**

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Web;
using System.Web.Mobile;
using System.Web.SessionState;
using System.Web.UI;
using System.Web.UI.MobileControls;
using System.Web.UI.WebControls;
using System.Web.UI.HtmlControls;
using System.Data.OleDb;
using System.Data.SqlClient;


namespace Prueba
{
publicpartialclassPedido : System.Web.UI.MobileControls.MobilePage
    {

        #region " Web Form Designer Generated Code "

//This call is required by the Web Form Designer.
        [System.Diagnostics.DebuggerStepThrough()]
privatevoid InitializeComponent()
        {
            cmdPedido.Click += newEventHandler(cmdPedido_Click);
        }

void cmdPedido_Click(object sender, EventArgs e)
        {
OleDbConnection cn = null;
            System.Data.DataSet ds = null;
OleDbDataAdapter da = null;
string strConexion = null;
string Ssql = null;

            strConexion = @"Provider=SQLOLEDB;Data Source=BORIS-
PC\SQLEXPRESS;Integrated Security=SSPI;Initial Catalog=Cines";

//Abrimos la conexión
            cn = newOleDbConnection(strConexion);
cn.Open();

            ds = new System.Data.DataSet();
```

```
            Ssql = "SELECT * FROM cliente where identificacion = '" +
txtIdentificacion.Text + "'";
            da = newOleDbDataAdapter(Ssql, cn);

OleDbCommandBuilder cmdBuilder = newOleDbCommandBuilder(da);
//Cargamos el DataSet
            da.Fill(ds, "cliente");
if (ds.Tables["cliente"].Rows.Count <= 0)
Response.Redirect("Error.aspx?mensaje=No existe cliente con esa
identificacion");
int intIdCliente = 0;
            intIdCliente =
Convert.ToInt32(ds.Tables["cliente"].Rows[0]["id_cliente"].ToString());

            ds = new System.Data.DataSet();
            Ssql = "SELECT * FROM producto where producto = '" +
sltProducto.Selection.ToString() + "'";
da = newOleDbDataAdapter(Ssql, cn);

//Cargamos el DataSet
            da.Fill(ds, "producto");
int intIdProducto = 0;
int intDisponible = 0;
int intCantidad = 0;
int intTotal = 0;
            intDisponible =
Convert.ToInt32(ds.Tables["producto"].Rows[0]["stock"].ToString());
            intIdProducto =
Convert.ToInt32(ds.Tables["producto"].Rows[0]["id_producto"].ToString());
            intCantidad = Convert.ToInt32(txtCantidad.Text);
if (intCantidad > intDisponible)
                Response.Redirect("Error.aspx?mensaje=Stock no disponible");
intTotal = intDisponible - intCantidad;


            Ssql = "insert into pedido values(" + intIdProducto.ToString() + ","
+ intIdCliente.ToString() + "," + intCantidad.ToString() + ")";
OleDbCommand objInsertar = newOleDbCommand();
objInsertar.Connection = cn;
            objInsertar.CommandType = CommandType.Text;
            objInsertar.CommandText = Ssql;
            objInsertar.ExecuteNonQuery();
//da = new OleDbDataAdapter(Ssql, cn);
            Ssql = "update producto set stock = " + intTotal.ToString() + "
where id_producto = " + intIdProducto.ToString();
OleDbCommand objActualizar = newOleDbCommand(Ssql, cn);
            objActualizar.Connection = cn;
            objActualizar.CommandType = CommandType.Text;
objActualizar.CommandText = Ssql;
            objActualizar.ExecuteNonQuery();

            Response.Redirect("Error.aspx?mensaje=Pedido realizado con exito");


//            da.InsertCommand = "insert into pedido values(" +
sltProducto.Selection.ToString() + "," + intIdCliente.ToString() + "," +
intCantidad.ToString();
```

```
        }

privatevoid Page_Init(System.Object sender, System.EventArgs e)
        {
//CODEGEN: This method call is required by the Web Form Designer
//Do not modify it using the code editor.
            InitializeComponent();
        }

        #endregion

protectedvoid Page_Load(object sender, EventArgs e)
        {
if (!IsPostBack)
            {
OleDbConnection cn = null;
                System.Data.DataSet ds = null;
OleDbDataAdapter da = null;
string strConexion = null;
string Ssql = null;

                strConexion = @"Provider=SQLOLEDB;Data Source=BORIS-
PC\SQLEXPRESS;Integrated Security=SSPI;Initial Catalog=Cines";

//Abrimos la conexión
                cn = newOleDbConnection(strConexion);
cn.Open();

                ds = new System.Data.DataSet();
                Ssql = "SELECT * FROM producto";
                da = newOleDbDataAdapter(Ssql, cn);

OleDbCommandBuilder cmdBuilder = newOleDbCommandBuilder(da);
//Cargamos el DataSet
                da.Fill(ds, "producto");

sltProducto.DataSource = ds.Tables["producto"];
                sltProducto.DataTextField = "producto";
                sltProducto.DataValueField = "id_producto";
                sltProducto.DataBind();
//lstStock.Items.Clear();
//foreach (DataRow fila_loopVariable in ds.Tables["producto"].Rows)
//{
//    fila = fila_loopVariable;
//    lstStock.Items.Add(fila["producto"] + " - " + fila["stock"]);
//}
            }
        }
public Pedido()
        {
            Load += Page_Load;
Init += Page_Init;
        }
    }
}
```

4.5. PRUEBAS DEL SISTEMA

La prueba del software es el proceso que permitrá verificar y revelar la calidad del producto que se ha desarrollado. Esto se utilizará para la identificación de posibles fallos que puedan presentarse en el momento de la ejecución. Todo esto es un conjunto de herramientas, técnicas y métodos que hacen llegar a la excelencia del programa. Involucrará las operaciones del sistema bajo condiciones controladas y evaluando los resultados.

Forzará intencionalmente al programa para producir un error en las respuestas, determinando si los sucesos ocurren cuando no tiene que hacerlo o cuando los hechos no suceden cuando deberían suceder.

Los tipos de pruebas a realizarse son: Pruebas unitarias, Caja blanca, y Caja negra.

4.5.1. PRUEBAS UNITARIAS

Las pruebas unitarias permitirán probar el correcto funcionamiento de los módulos, asegurandouna funcionalidad exitosa del sistema. Para esto se usarán las herramientas de prueba de Visual Studio Application Lifecycle Management.

Crear el test unitario, para realizar pruebas:

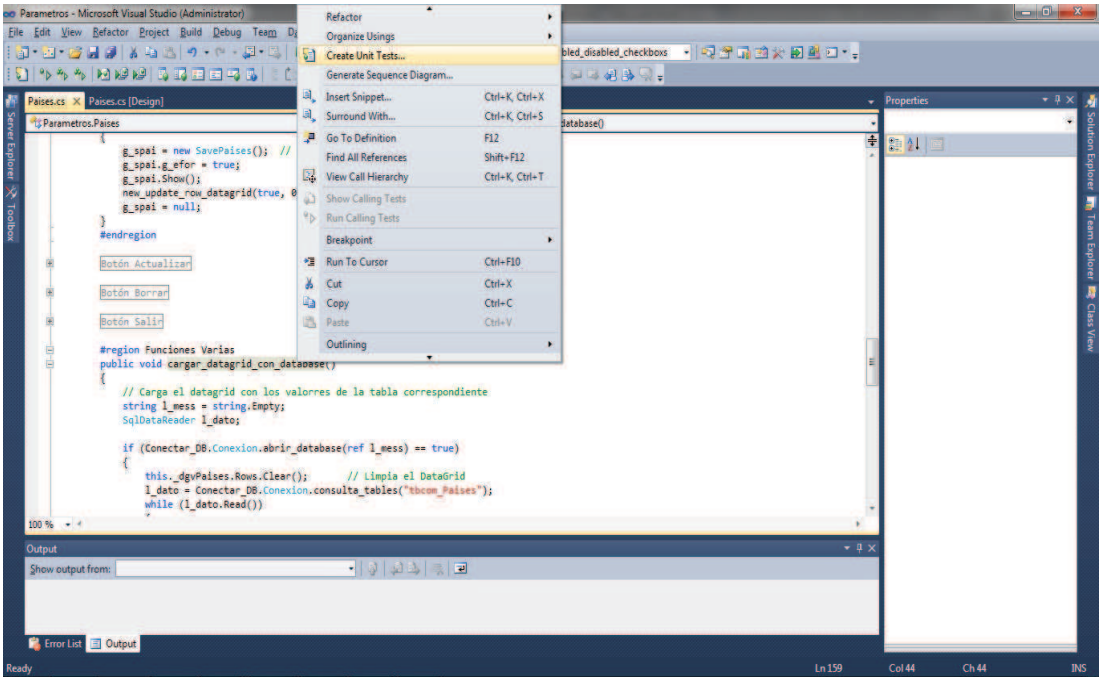


Fig. 4.1Creación del Proyecto Test Único

Fuente: Tesistas

Se selecciona cada una de las sesiones para que pasen por el test, así como las funciones y clases creadas para el sistema. Por ejemplo se realizará de una a una las sesiones y su contenido, seleccionando en primera instancia la sesión de Usuarios, y así sucesivamente con las demás.

El test también se lo puede realizarseleccionando varias sesiones al mismo tiempo, pero eso no permitirá dar un mejor seguimiento al sistema. En el siguiente gráfico, se muestra la manera de selección.

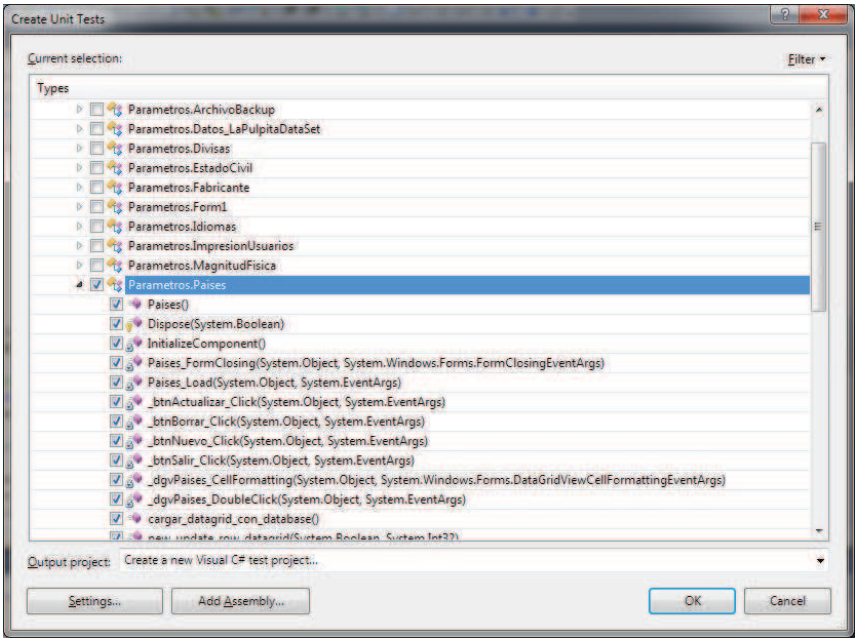


Fig. 4.2Selección de las Sesiones para el Test

Fuente: Tesistas

Creación del proyecto para el test:

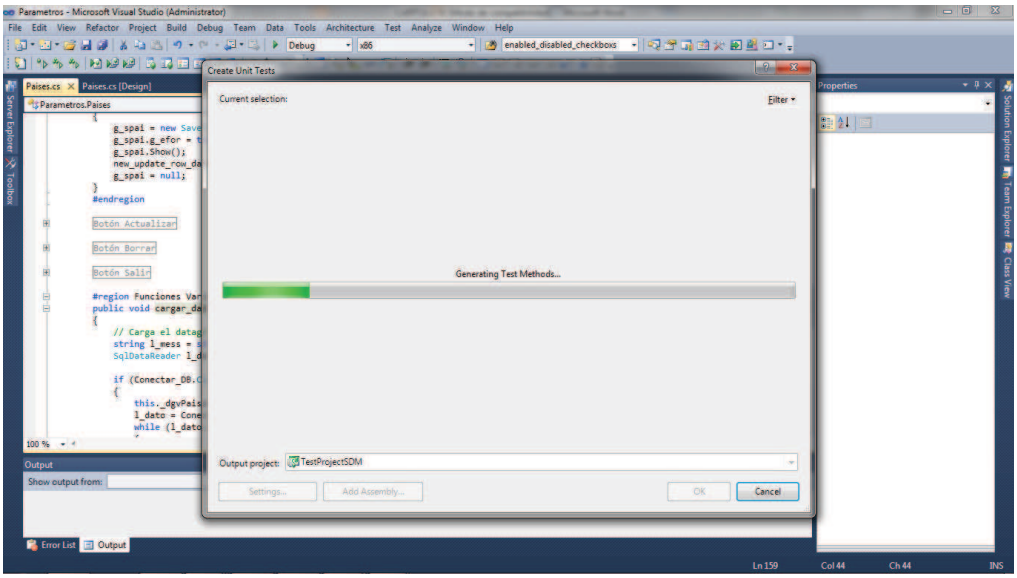


Fig. 4.3Carga del Proyecto de Test Creado

Fuente: Tesistas

4.5.2. PRUEBAS CAJA BLANCA

La caja blanca es un tipo de pruebas de software que se realizará sobre las funciones internas de un módulo. Las técnicas que se usarán son: La cobertura de caminos (pruebas que se harán recorriendo todos los posibles caminos de ejecución), y comprobación de bucles (se verificarán los bucles para 0, 1 y n iteracciones, y luego las iteracciones máximas, máximas menos uno y mas uno).

Las pruebas de caja blanca se llevarán en primer lugar, sobre un módulo concreto, como por ejemplo: Módulo de Usuarios – Auditoría - Backup. En estas se encontrarán también los llamados a coberturas (sentencia, decisión, condición y múltiple).

4.5.2.1. DETALLE DE PRUEBAS CAJA BLANCA

Los pasos generales realizados por este tipo de pruebas, en cada uno de los módulos del sistema son los siguientes:

- **Cobertura de sentencias.-** cada sentencia se ejecuta al menos una vez.

- **Cobertura de decisiones.-** cada decisión toma todos los resultados posibles al menos una vez.

- **Cobertura de condiciones.-** cada condición en la decisión toma todos los posibles resultados al menos una vez.

- **Cobertura de condiciones múltiples.-** todas las posibles combinaciones de los resultados de la condición en cada decisión ocurre al menos una vez.

A continuación se describe lo realizado en los módulos, para analizar los resultados, determinar el tipo de solución que se aplicará al problema encontrado:

- **Módulo Usuario – Auditoría - Backup.-** se crearon usuarios para el sistema, con todas las combinaciones de parámetros que pueden tener. Se imprimió el reporte de la auditoría. Y se exportó un backup de la base de datos del sistema.

El inconveniente que se presentó fue en el instante que cada usuario, ingresaba al sistema, el comportamiento que tenía en el mismo según sus parámetros, no se abrían ciertas sesiones en modo sólo de consulta según el parámetro. Para corregir el errorse incluyó la validación de modo de consulta a las sesiones que así las debía visualizar el usuario.

- **Módulo Parámetros.-** se crearon nuevos datos, y actualización de los mismos en cada uno de los parámetros, comprobando también la eliminación al seleccionar uno o varios registros al mismo tiempo.

El único error en común que se presentó, fue la sincronización de las sesiones de consulta con las de ingreso de datos, al cargar los campos seleccionados para ser actualizados. En la solución se procedió a colocar correctamente los parámetros recibidos por la sesión de actualización para que ésta los lea, posterior a esto los presente en pantalla, en el campo y orden que les fue asignado.

- **Módulo Datos Generales.-** se realizaron los ejercicios comunes de ingreso, actualizar, y borrado de los datos generales, en los cuales

al no tener procesos complicados, ni una cargaexcesiva de datos, no se presentaron errores en las pruebas realizadas.

– **Menú Fabricación.-** a parte de realizar las pruebas comunes de los anteriores módulos, se verificó la carga de los de registros en las cuadrículas, y el proceso de crear las órdenes de fabricación.

Uno de los errores encontrados y que fue común en los siguientes menús,fue el proceso almacenado ejecutado para la consulta de varios campos, pertenecientes a distintas tablas de la base de datos, que en el momento de ser cargados al datagrid de la sesión, lo hacía de manera desordenada, e inconsistente con lo que debía ser presentado. Para reparar este error se cambió el proceso almacenado que estaba consultando todos los campos de las tablas relacionadas, por buscar sólo aquellos que son necesarios, en su mayoría los campos que contienen las descripciones de los códigos.

Otro problema fue encontrado al momento de ingresar las líneas en las órdenes de fabricación, las cuales se guardaban al finalizar el ingresodel último campo de la línea, y al querer ingresar otra de manera consecutiva, el registro se sobreescribía en el anterior. La corrección de esto requirió cambiar la forma de cargar las líneas en la base de datos, ya no haciéndolo línea a línea, si no al finalizar el ingreso de todas las líneas deseadas, y presionando el botón de guardar.

– **Módulo de Almacenamiento.-** con la ejecución de las pruebas generales, se ejecutaron adicionalmente las consultas de stock, como estaban siendo cargadas las tablas, y los valores presentados a pantalla sean los correctos. Se validó el proceso de almacenamiento de

producto, sea éste de entrada (compra o proceso de fabricación) o salida (venta o resultado de lo fabricado).

El inconveniente encontrado en este módulo se concentró en el movimiento de stock de los artículos, el cual al tener un flujo grande, entre salida, entrada, transferencia, producida por falta de variables globales el mal cálculo del stock al realizar cualquiera de los procesos. Por tal razón se añadierón para estos movimientos, variables que se puedan consultar en todo el sistema, las cuales almacenan la cantidad del artículo, para luego realizar las conversiones de unidad, si así lo amerita, y cargar el stock realizando una operación con cantidades reales.

- **Módulo de Ventas.-** se realizaron pruebas de carga de datos y creación de órdenes de venta, la validación de cada proceso. El paso de convertir cotizaciones en órdenes de venta.

Al momento de pasar de una cotización a una orden de venta, se encontró inconsistencia entre las dos órdenes, no se copiaban las líneas en su totalidad, o cambiado los valores, y campos vacíos. La solución a esta novedad, fue leer línea a línea la cotización que esta guardada, y copiar de una en una las filas, y en ese instante copiar en la otra tabla los datos.

- **Módulo de Compras.-** se comprueba que la compra realizada, llegue a ser creada de manera correcta, con las líneas de esta órden completando los campos necesarios para la misma.

El inconveniente presentado se tuvo al permitir guardar la orden de compra sin que las líneas estén correctamente ingresadas, y esta

validación es muy importante para evitar datos inconsistentes al momento de pasar los artículos comprados, al módulo de almacenamiento, distribuyendo a las ubicaciones. Se colocó banderas que servirían para identificar una vez se cargaran las líneas correctamente, caso contrario no permite la creación de una nueva línea de compra.

4.5.3. PRUEBAS CAJA NEGRA

La prueba de caja negra es el estudio desde el punto de vista de las entradas que se recibirán y las salidas o respuestas que producirá el sistema, sin tomar en cuenta el funcionamiento interno. Probará la forma en la que interactuará con el medio que le rodea, entendiendo que es lo que realiza, pero sin dar importancia a cómo lo hace. Para esto se tiene bien definido cuáles son sus entradas y salidas, es decir la interfaz.

Al estar dividido en módulos el sistema permitirá dar una visión más clara, dando al sistema una forma más robusta y fácil de mantener, en caso de ocurrir un fallo.

4.5.3.1. DETALLE DE PRUEBAS CAJA NEGRA

El primer error encontrado después de validar el ingreso del usuario al sistema, se presentó al momento de acceder a los menús de cada uno de los módulos, que una vez visibleslas sesiones en pantalla, se detectó que se podía volver a ejecutar la misma sesión varias veces, teniendola n cantidad de veces abierta, lo cual es incorrecto, por lo que se procedio a validar que una vez presente en pantalla, ya no se vuelva abrir otra sesión igual, dejandola visible una sola sesión por menú.

En cada uno de los módulos a nivel de funcionamiento, se encontraron los siguientes errores, descritos a continuación, con sus respectivas soluciones:

- **Módulo Usuario – Auditoría – Backup.-** la única observación encontrada, se presentó en la sesión generación del bakcup de la base de datos del sistema, en la que se tenía deshabilitado el campo “Base de Datos” para el ingreso del nombre. Para esto se bloqueo el campo impidiendo su edición, y asi evitar posibles errores innecesarios.

- **Módulo Parámetros.-** a partirde este módulo en adelante se recomendó que se identifique de alguna manera los campos que son de tipo obligatorio, para que el usuario del sistema tenga presente que es lo que debe llenar antes de guardar los datos, y evitar asi los mensajes de que falta completar campos. Se procedió para dar una mejor presentación a todo el sistema, colocar un asterisco de color rojo al lado izquiero del campo que es obligatorio ser ingresado para guadar los datos.

- **Módulo Datos Generales.-** se tomó como error que en las sesiones de consulta, especificamente en aquellas que tienen muchos campos, como por ejemplo: Direcciones, Empleados, Artículos, y Partners, no se coloquen en la cuadrícula, todas las columnas, si no solamente aquellos que son necesesarios para su identificación, los más importantes, haciendo la sesión más pequeña, amigable y entendible para el usuario.

- **Módulo Fabricación.-** en este módulo se encontró, que la sesión de creación de las órdenes de fabricación debía cambiarse en su mayor parte, para que tenga una distribución de los campos, de manera que asemejen a la hoja que en el pasado eran llenadas por los

empleados encargados de este proceso. Con esto se le dará al sistema una presentación más amigable, con lo que se venía haciendo en la empresa, y el impacto de la introducción de este sistema sea más facil de asimilar.

- **Módulo Almacenamiento.-** se encontró un mal proceso en las órdenes de almacenaje y órdenes de corrección de stock, las cuales no estaban actualizando correctamente las cantidades existentes en unidad de stock en los almacenes, al momento de describir en las líneas de las órdenes una ubicación de las mismas, dando como resultado que la suma de stock de las ubicaciones con el total del almacén no eran similares. Lo que fue solucionado con una correcta actualización de los datos considerando que la ubicación pertenece a un almacén, y por ende el stock en este también aumenta.

- **Módulo Ventas.-** en la revisión de este módulo se presentó como novedad que al instante de convertir una oferta de venta, en una orden de venta, lo hacía totalmente transparente al usuario. Se aceptó como recomendación la creación de una nueva sesión al momento de querer realizar este proceso, en esta sesión se presenta el código de la oferta de venta, y la selección de la serie para la orden de venta.

- **Módulo Compras.-** éste es el único módulo en que no se encontró inconvenientes a nivel funcional.

CAPÍTULO V
MANUALES DE USUARIO E INSTALACIÓN

5.1. INTRODUCCIÓN

El documento de Manual de Usuario describe el manejo de cada uno de los módulos del sistema, con la configuración dada por el desarrollo del sistema. Se debe tener en cuenta que la funcionalidad del sistema puede variar cuando se desee.

El manual de usuario es creado para que pueda ser entendido por cualquier usuario principiante como por ejemplo: usuario encargado de la toma de pedidos, usuario de ventas y compras, y usuario de la fabricación del producto,e incluso llegará a ser útil a usuarios avanzados, es decir que ya conozcan el sistema y olviden algún punto específico del funcionamiento.

El Manual de Instalación, explica la manera de cómo se llevará a cabo la instalación, en cualquier servidor, y máquina del cliente, detallando paso a paso, desde las especificaciones requeridas en las computadoras, hasta la descripción de cada una de las pantallas y las opciones que se tiene que seguir.

5.2.NAVEGACIÓN

A continuación se describirá el funcionamiento de las grillas que contienen los datos consultados, y cada uno de los botones que se encuentran en todo el sistema:

Grilla→ En todas las grillas se tienen las siguientes opciones:

1) Al seleccionar una fila que contenga datos, se puedecopiar o eliminar. También se puede seleccionar varios registros y borrarlos todos al instante.

2) Cuando se haga doble clicken una de las filas con datos, éste automáticamente llamará a las sesiones donde se Actualizán los registros.

3) Si se da doble click en la fila que no tiene datos, es decir la última que se encuentra en blanco, llamará a la sesión de crear un nuevo registro.

Botón Ingresar→Con este se permite el acceso al sistema. Acceso rápido “Alt + i”.

Botón Cancelar→Se rechaza el ingreso al sistema. Acceso rápido “Alt + c”.

Botón Usuarios→Se accede al menú de usuarios, auditoría, y backup de la base de datos. Acceso rápido “Alt + u”.

Botón Parámetros→Ingreso al menú de parámetros del sistema, donde se encuentran los Datos de la Empresa, y parámetros generales del sistema. Acceso rápido “Alt + p”.

Botón Datos Generales→Se puede acceder al menú de Datos Generales donde se encontrará por ejemplo: Países, Ciudades, Direcciones, Empleados, Partners, etc. Acceso rápido “Alt + d”.

Botón Fabricación→Accede al menú de Fabricación en el sistema, en el que se encuentra, la producción de la pulpa. Acceso rápido “Alt + f”.

Botón Almacenamiento→Ingreso al menú de Almacenamiento en el sistema, en el que se encuentran las órdenes de almacenaje, y consultas de stock del producto. Acceso rápido “Alt + a”.

Botón Ventas→Se encuentra el menú de Ventas del sistema, así como también las cotizaciones que se pueden realizar. Acceso rápido “Alt + v”.

Botón Compras→Muestra el menú de Compras, con todo lo necesario para la adquisición de materia prima e insumos. Acceso rápido “Alt + c”.

Botón Nuevo→Llama a la sesión que sirve para el ingreso deun nuevo registro en el sistema. Acceso rápido “Alt + n”.

Botón Actualizar→Actualización de registros, desde el llamado de la sesión en la que se realiza esta acción. Acceso rápido “Alt + a”.

Botón Copiar→Realiza la copia de un registro seleccionado previamente, para ingresarlo en el sistema desde la sesión llamada para este paso. Acceso rápido "Alt + c".

Botón Borrar→Borra el registro o los registros que se seleccionaron. Acceso rápido "Alt + b".

Botón Imprimir→Llama a la sesión que imprime los registros. Acceso rápido "Alt + i".

Botón Salir→Sale de la sesión en la que se encuentra actualmente. Acceso rápido "Alt + s".

Botón Guardar→Guardael registro, nuevo o modificado, según sea el caso. Acceso rápido “Alt + g”.

Botón Deshacer→Deshace los cambios realizados en los campos de la sesión, sea en el estado de actualización, copiado, o nuevo. Acceso rápido “Alt + d”.

Botón Zoom→Llama a la sesión de zoom, la misma que contiene el listado del dato requerido.

Botón Descargar→ Al seleccionar un registro de la sesión de zoom, se descarga el campo solicitado desde donde fue llamado.

5.3. SEGURIDAD EN EL SISTEMA

Para la creación de este sistema informático aplicado a la empresa La Pulpita, es muy importante considerar como controlar la seguridad del mismo.

La información constituirá el activo más importante de la empresa, frecuentemente se olvida ciertas condiciones fundamentalesque hay que recordar:

- La información se encuentra almacenada y se procesa a través de computadoras.

- Puede ser confidencial para ciertos empleados, dependiendo su nivel jerárquico.

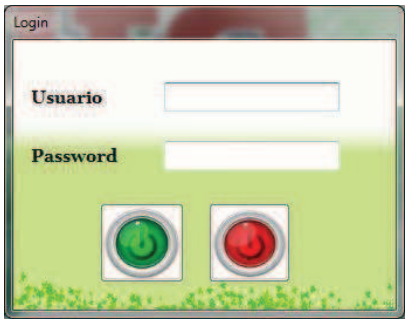
- Puede ser mal utilizada o divulgada.

- Está sujeta a riesgos como: robos, sabotajes, fraudes, incendios, daños en los equipos, virus u otros.

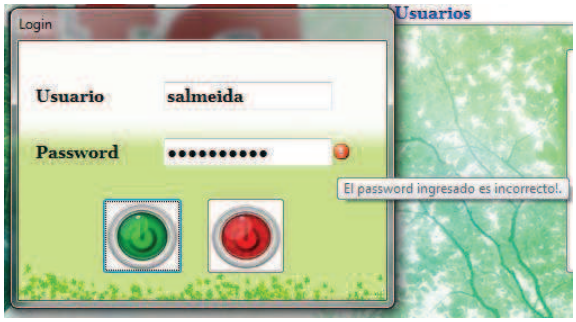
Siendo este un punto crítico en la empresa, es algo que debe resolver. Para asegurar la información y el sistema como tal en La Pulpita se han controlado los siguientes aspectos:

5.3.1.INICIO DEL SISTEMA

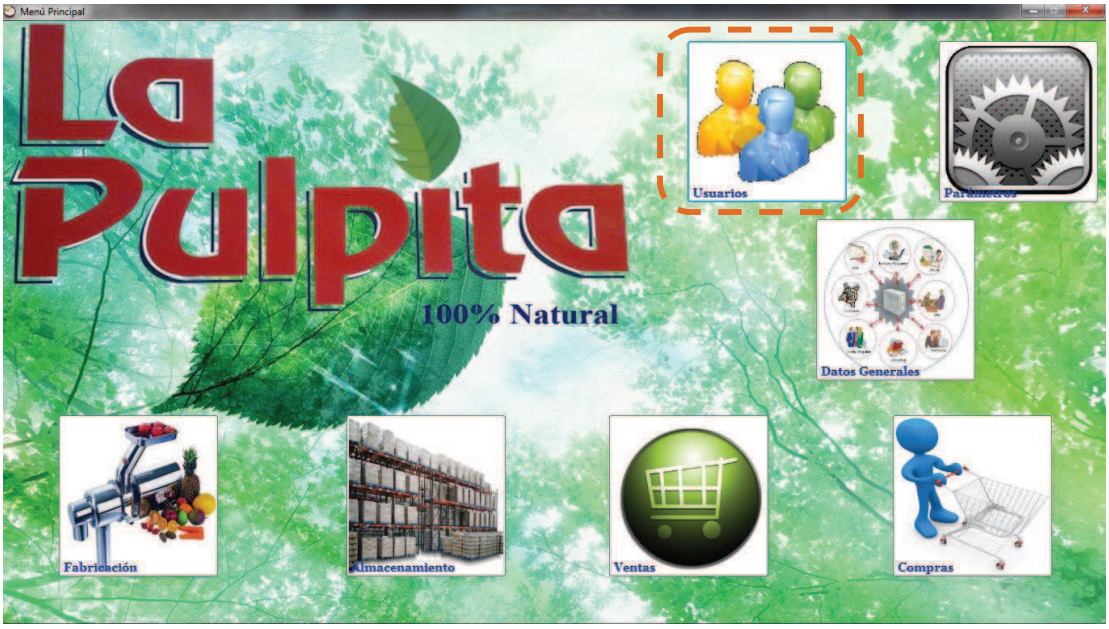
El ingreso al sistema se realiza mediante una sesión de autentificación de usuario y password donde se valida que los mismos hayan sido creados previamente. La sesión cuenta con el botón de ingreso (botón verde) y el botón de cancelar (botón rojo).



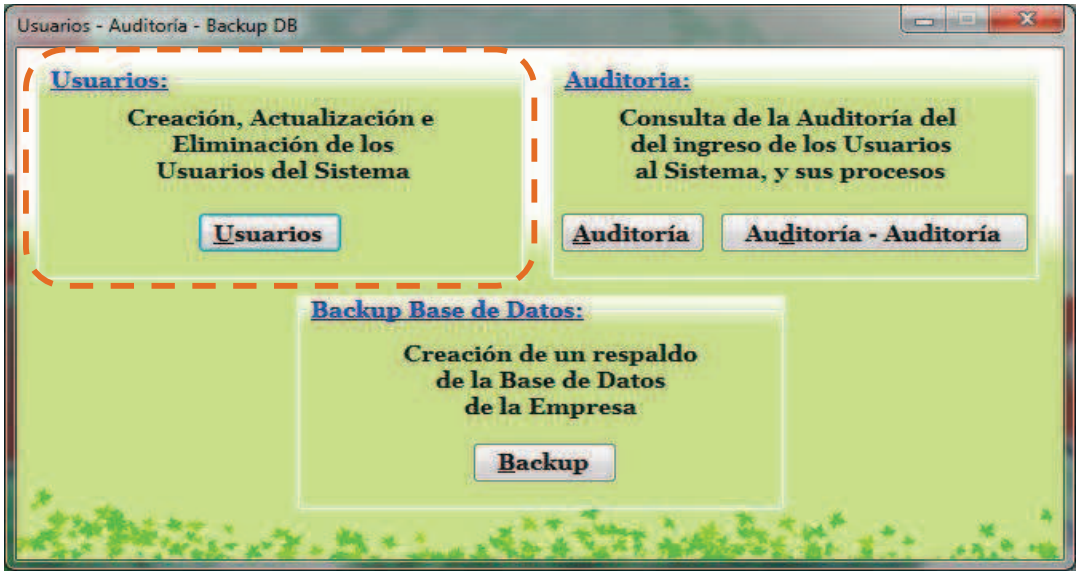
Cuando uno de los dos campos es ingresado incorrectamente, el sistema alerta con un mensaje.



En el caso que tanto el usuario con su password sean ingresados correctamente, el sistema permite el acceso a los módulos desplegando la siguiente pantalla:



El módulo que controla las seguridades es el de “Usuarios”, en este se podrá definir lo siguiente:



5.3.2. ACCESO A LOS MÓDULOS

En esta opción se maneja la creación de usuarios con su password, así como el tipo de permisos que pueden tener para los módulos, sean estos en modo sólo de consulta, o para realizar el proceso de Ventas, Compras, Almacenamiento, Producción.

La seguridad informática debe ser estudiada para que no impida el trabajo de sus operadores, debe ser definida no por los administradores informáticos sino por los responsables jerárquicos. Una buena política de seguridad es indispensable que sea definida y bien interpretada por el administrador del sistema.

Se presenta una sesión general de consulta con los datos del usuario y al lado los módulos a los que accede, las opciones pueden ser:

- Administrador: usuario que tendrá acceso a todos los módulos del sistema.

- Logístico: usuario que tendrá acceso a los módulos de Compras y Ventas.

- Producción: usuario que maneja Fabricación y Almacenamiento.

- Datos generales: siendo estos los datos maestros de la empresa no siempre están disponibles para todos los empleados.

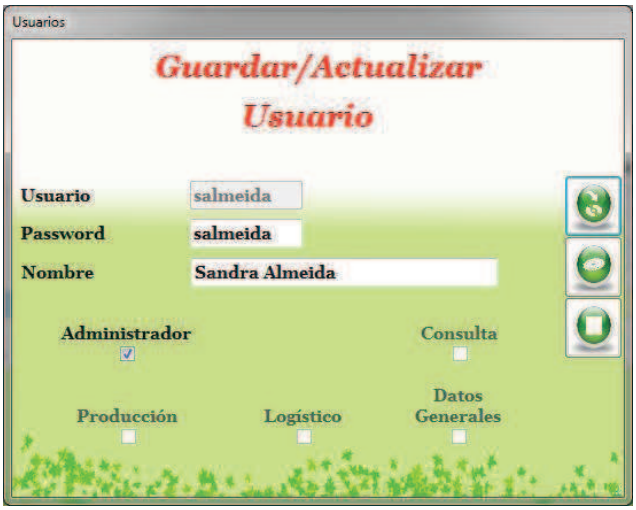
- Consulta: si un usuario tiene este check activado significa que cuenta con todos los módulos pero únicamente de consulta, no podrá insertar, modificar ni eliminar datos.

Si se ha escogido la opción Administrador, automáticamente se deshabilitan las otras cuatro opciones, de igual manera sucede si se ha escogido la opción de Consulta.

Si escoge una de las opciones Logístico, Producción o Datos Generales, no podrá escoger Administrador ni Consulta.

Adicionalmente desde esta sesión se pueden insertar, modificar o eliminar usuarios con los botones de la derecha.

Sesión para insertar / actualizar un nuevo usuario:

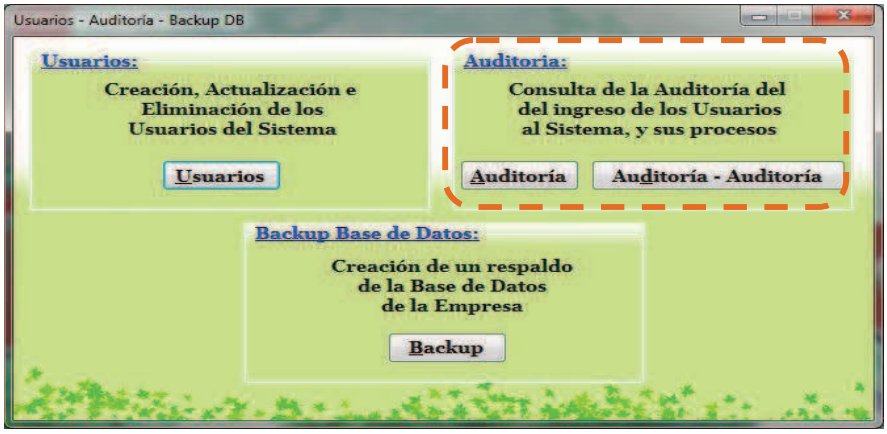


5.3.3. AUDITORÍA

En el módulo de Auditoría se cuenta con dos opciones:

- Auditoría: donde se listará un reporte del ingreso de los usuarios al sistema y los procesos que hayan ejecutado.

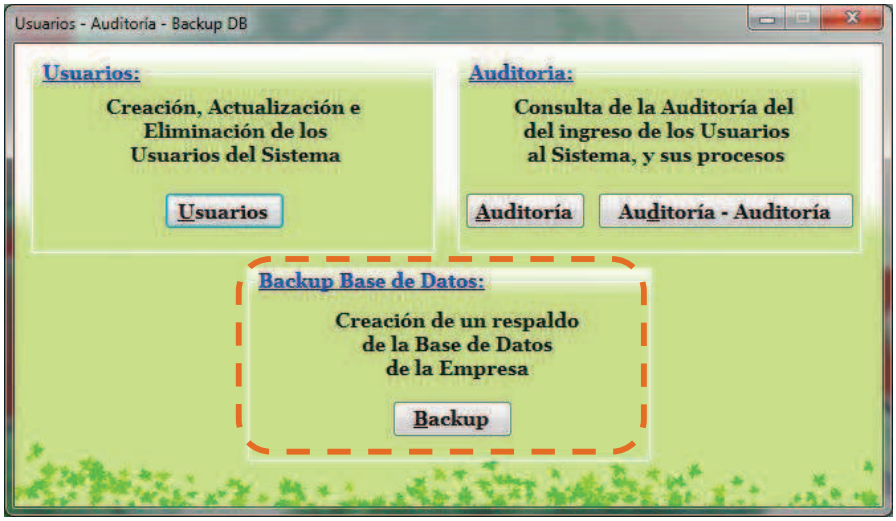
- Auditoría – Auditoría: este reporte presentará la información de los usuarios que hayan ejecutado la primera opción, es decir quienes hayan listado auditoría también pueden ser auditados.



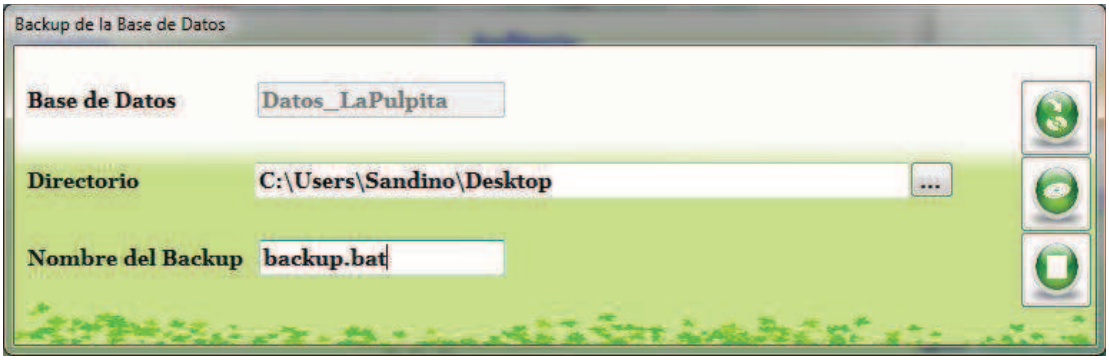
5.3.4. POLÍTICA DE RESPALDOS

Para la protección de los datos la medida más eficiente es definir una buena política de copias de seguridad o backups. Es vital para la empresa elaborar un plan de backup en función del volumen de transaccionabilidad y la cantidad de equipos críticos. En este caso, La Pulpita maneja un volumen medio por lo que la recomendación es que generen backups semanales, por ejemplo, cada lunes a primera hora de la mañana.

Desde la misma sesión, se puede acceder a Backup Base de Datos:



En esta sesión se debe ingresar la ubicación donde se quiere que se almacene elbackup y el nombre del archivo.

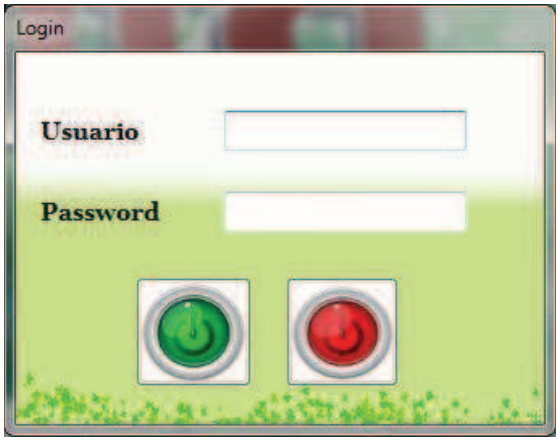


5.4.MANUAL DE USUARIO

El presente manual tratará todos los puntos, navegación, y opciones que presenta el sistema, para los usuarios que tendrán acceso al mismo, para desplazarse por los distintos módulos según sus áreas de trabajo.

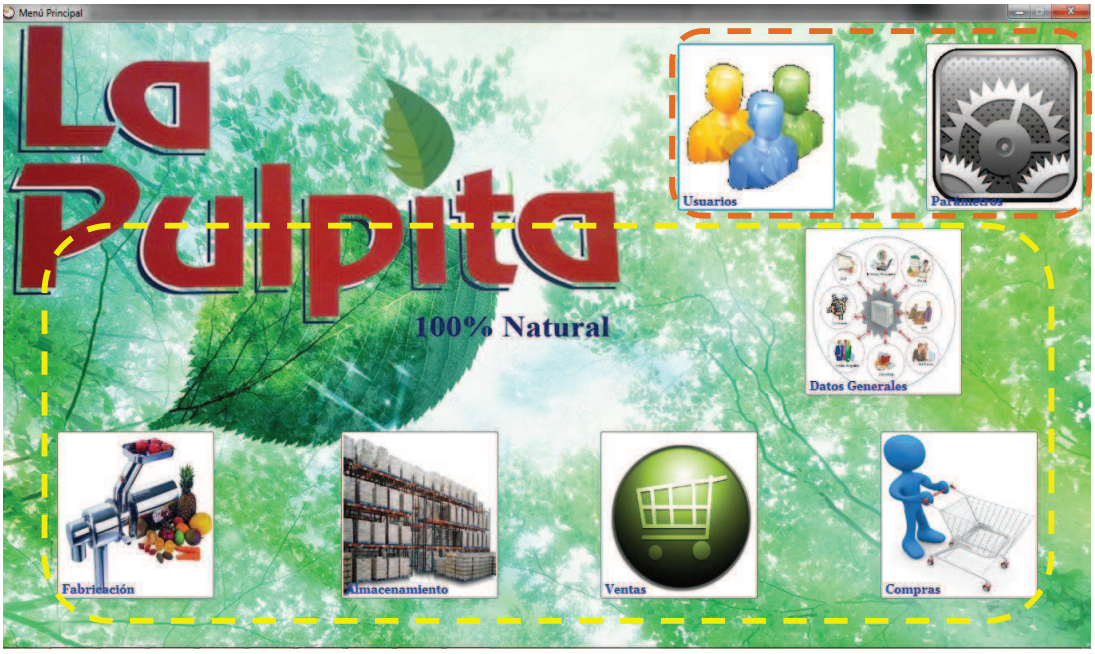
5.4.1. INGRESO AL SISTEMA

Por medio de la sesión de Login, se consigue el ingreso al sistema, una vez que el código del usuario ingresado, en conjunto de su password, son validados si existen registrados. Con el botón de Ingresar, se accede al sistema, y el de Cancelar se sale de la aplicación.



5.4.2. MENÚ PRINCIPAL

El siguiente menú, presenta todas las opciones, que se tiene en la sesión principal. El administrador tiene acceso a todas las opciones,no así los demás que no pueden ingresar a Usuarios, y Parámetros.

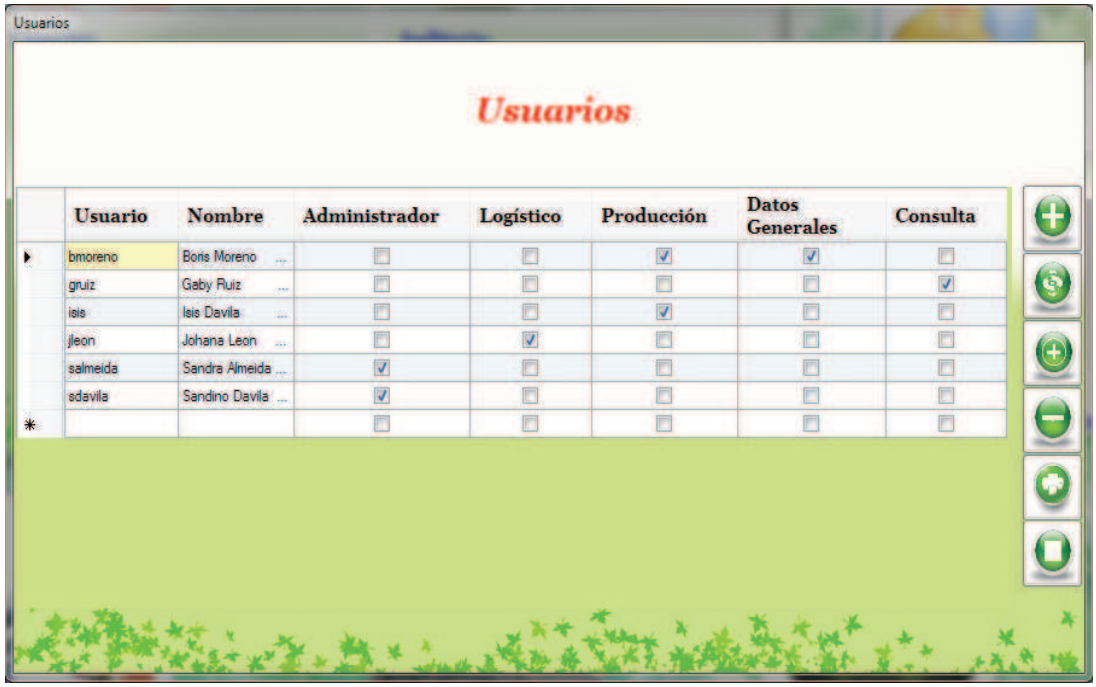


5.4.3. USUARIOS – AUDITORÍA - BACKUP

El siguiente menú contiene las sesiones de Usuarios, donde se consulta, elimina, imprime, agrega, y actualiza, los usuarios del sistema. La impresión de la auditoría del sistema, así como de la auditoría de la auditoría. Y por último sacar un backup de la base de datos que utiliza el sistema, con todos los datos existentes al momento.

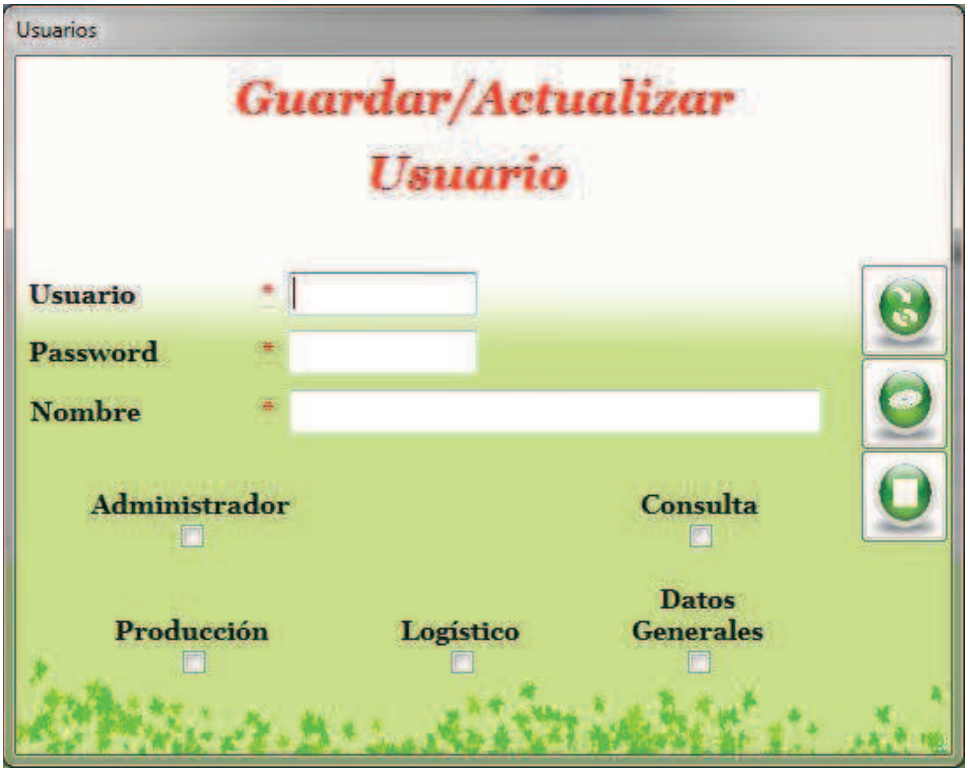
En esta lista de usuarios se puede dar click en uno de los registros con datos, y se accederá a la sesión de actualización, así como con el botón Actualizar, y click en un registro vacío, como en el botón Nuevo, llamará a la sesión de nuevo registro.

Para la eliminación de registros, se debeseleccionar una o varias filas y dar click en el botón Borrar.



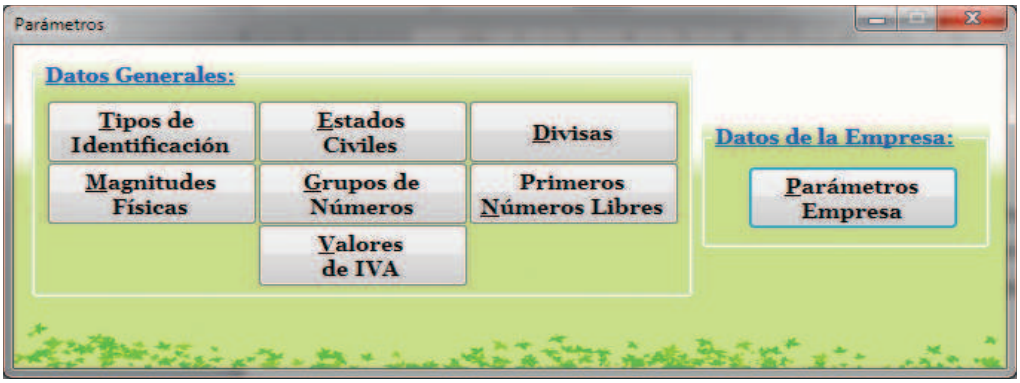
Para copiar uno de los registros, primero hay que seleccionarlo, y luego dar click en el botón Copiar, llevando a la sesión de nuevo registro, con los datos cargados del usuario seleccionado, menos su código y password.

La sesión de Guardar y Actualización de Usuario, contiene los datos requeridos para el registro. Los campos que a su lado izquierdo contienen un asterisco de color rojo, implican que es obligatorio el ingreso de valor. El botón de guardar registra el dato, y el botón de Deshacer, regresa todos los campos a su estado original.



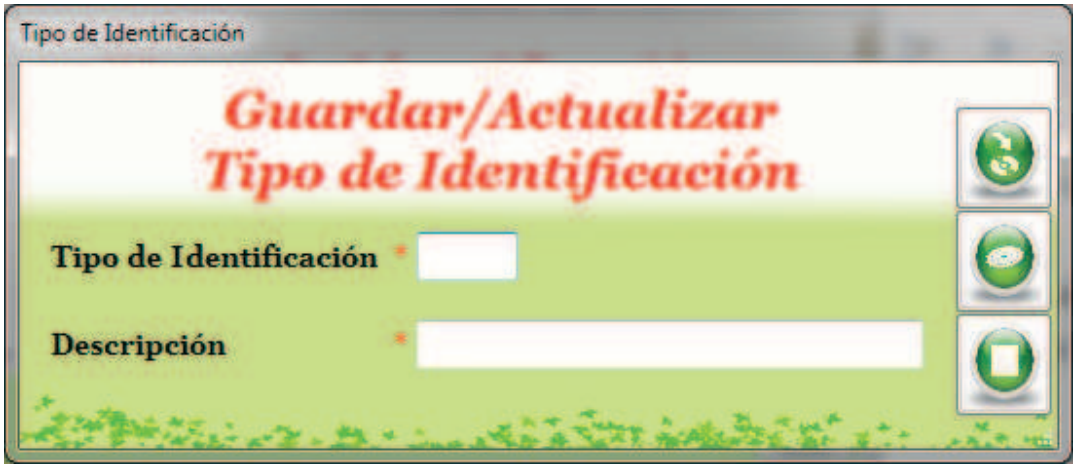
5.4.4.MENÚ PARÁMETROS

En este menú se tiene Datos de la Empresa, Tipos de Identificación, Estado Civil, Divisas, etc., los cuales son necesarios para el correcto funcionamiento del sistema, ya que de estos datos dependerán los demás procesos.



5.4.4.1. TIPOS DE IDENTIFICACIÓN

Por medio de un código de un carácter se tiene el tipo de identificación, como: Cédula, Pasaporte, RUC, los mismos que sirven para datos principales de Partners y Empleados.

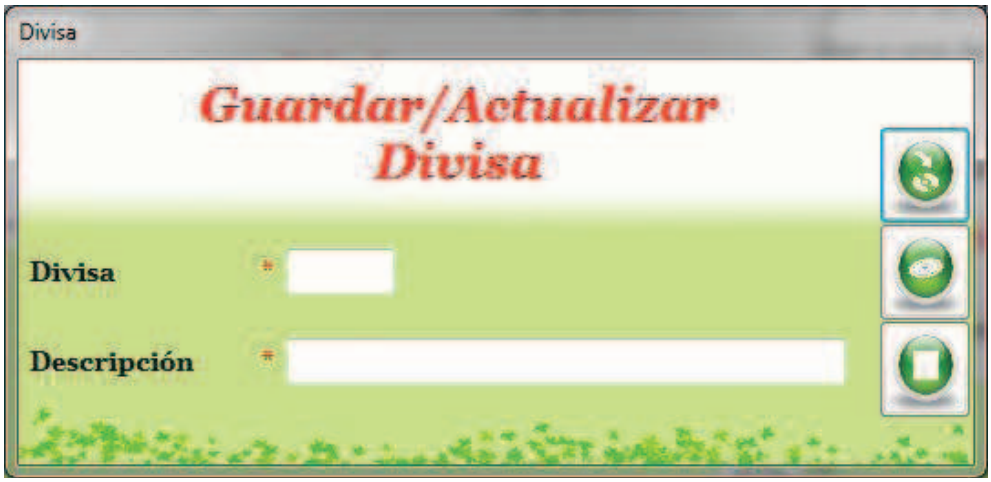


5.4.4.2. ESTADO CIVIL

Es aquel estado civil que puede tener una persona, el mismo que sirven como dato adicional de clientes, proveedores y empleados.

5.4.4.3. DIVISAS

Las divisas no son más que los tipos de moneda que existen en el mundo, especificadas por un código y una descripción.

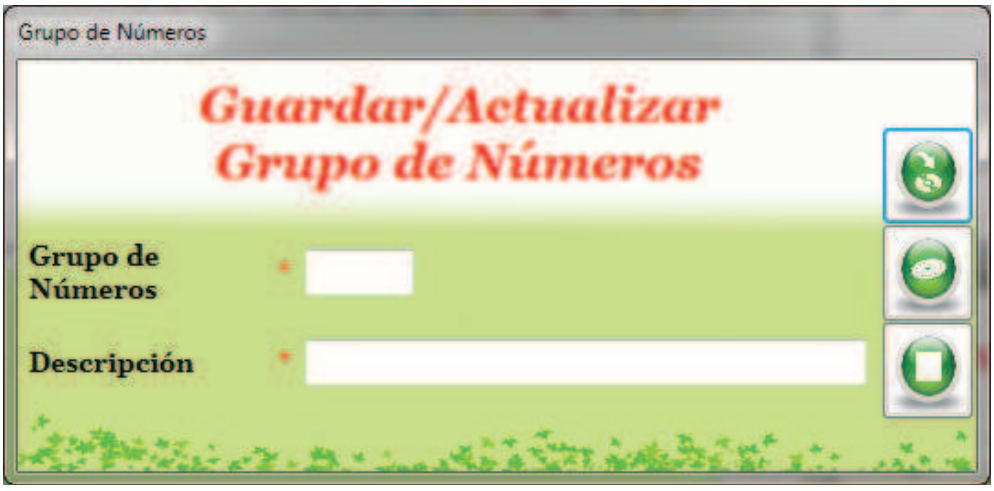


5.4.4.4. MAGNITUDES FÍSICAS

Las magnitudes físicas, son aquellas nomenclaturas internacionales en medidas de peso, que serán parte de la descripción de los artículos, y proceso de datos.

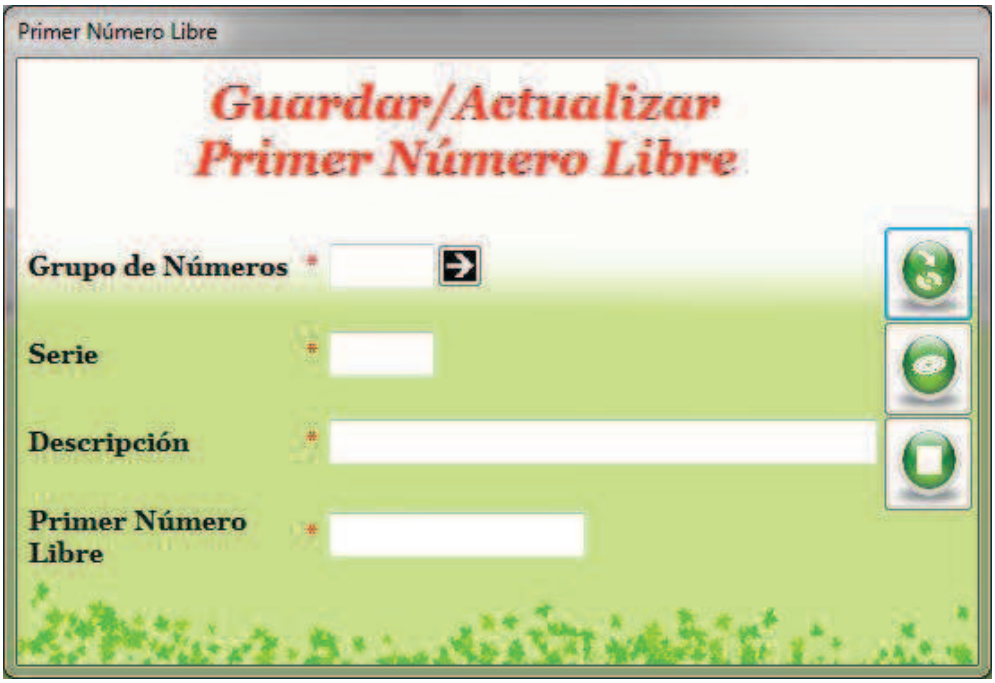
5.4.4.5. GRUPOS DE NÚMEROS

Estos sirven para agrupar las series de los números libres que se utilizan para la secuencia numérica que llevará un proceso, sea esta Orden de Venta, Orden de Compra, Orden de Fabricación, y Orden de Almacenamiento.



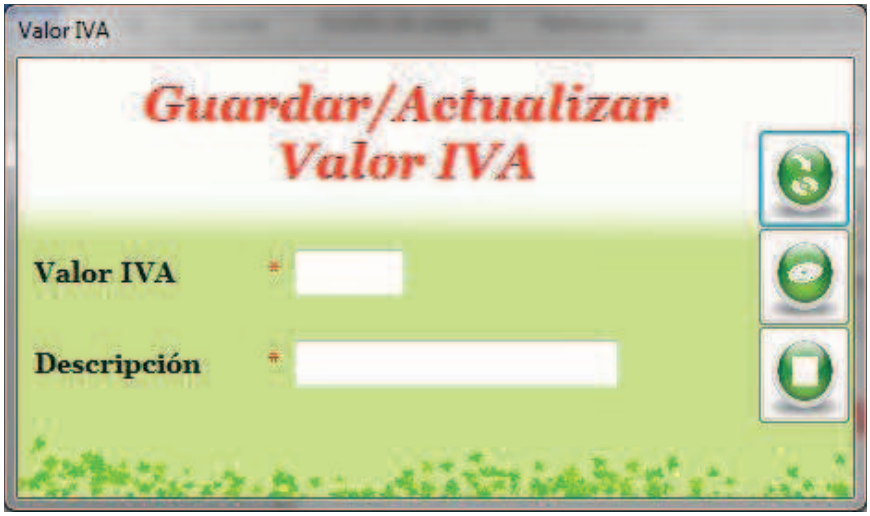
5.4.4.6. PRIMEROS NÚMEROS LIBRES

Estos poseen una serie atada al grupo descrito anteriormente, que contienen un número entero que representa el número libre a usarse, en la creación de todos los tipos de órdenes.



5.4.4.7. VALORES DE IVA

Se tiene todos aquellos valores de iva, que van a ser utilizados tanto en ventas, como en compras



5.4.4.8. PARÁMETROS EMPRESA

Contiene todos aquellos datos principales y secundarios referentes a la empresa.

5.4.5. MENÚ DATOS GENERALES

Contiene el acceso a todos los datos que son considerados como generales para el sistema, por ejemplo: Países, Ciudades, Direcciones, Empleados, etc.

5.4.6. MENÚ FABRICACIÓN

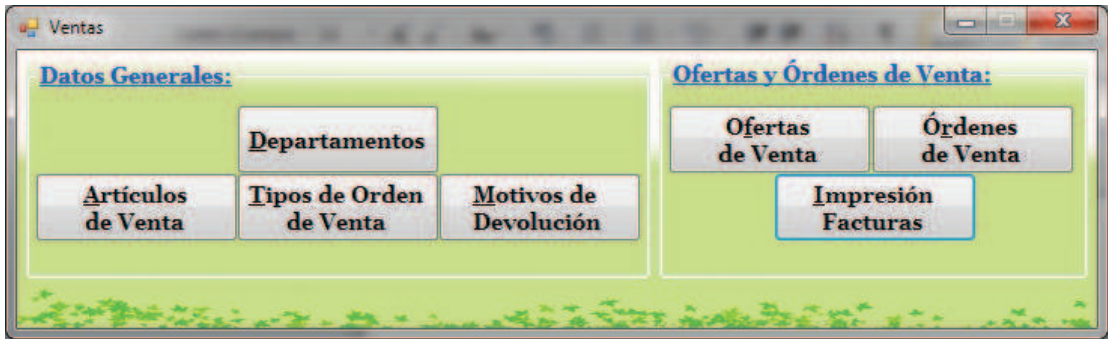
Se puede acceder a todos los datos generales, previos y necesarios para realizar una orden de fabricación.

5.4.7. MENÚ ALMACENAMIENTO

Son todas aquellas opciones en las cuales como primera instancia los datos generales son primordiales, para realizar Órdenes de Almacenamiento, y de Corrección de Stock, de igual manera consultar el stock de un artículo, desde almacenes o ubicaciones.

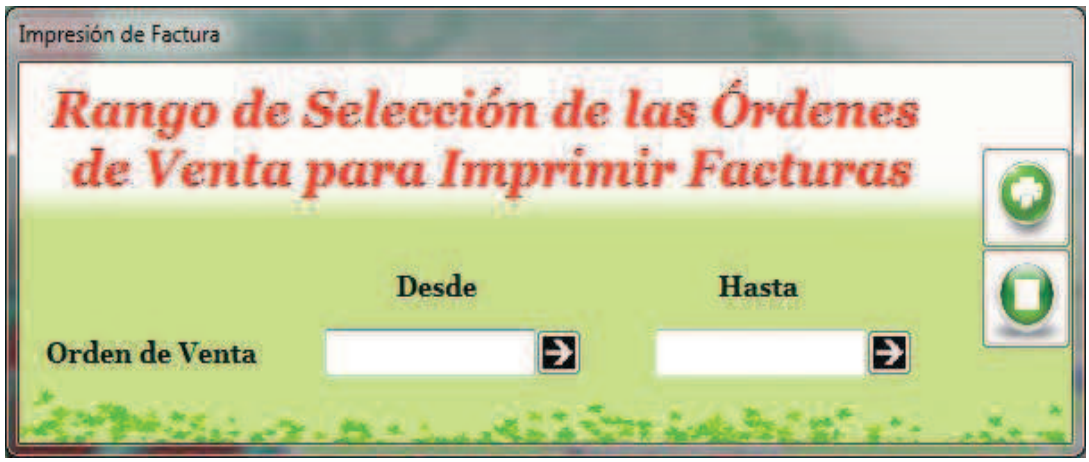
5.4.8. MENÚ VENTAS

Se tiene acceso a todo aquello que es necesario para la creación de Órdenes de Ventas, y Ofertas de Venta.



5.4.8.1. IMPRESIÓN FACTURAS

Para imprimir las facturas en los formatos pre-impresos de la empresa, se parte de la orden de venta una vez aprobada, y finalizada, permitiendo acceder a la opción de impresión con aquellos datos pertenecientes a la misma.



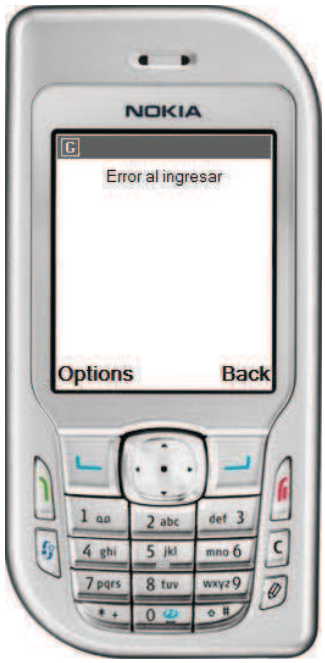
5.4.9. MENÚ COMPRAS

Existen las opciones generales que hacen que una orden de compra se procese sin inconvenientes.

5.4.10. MENÚ WAP

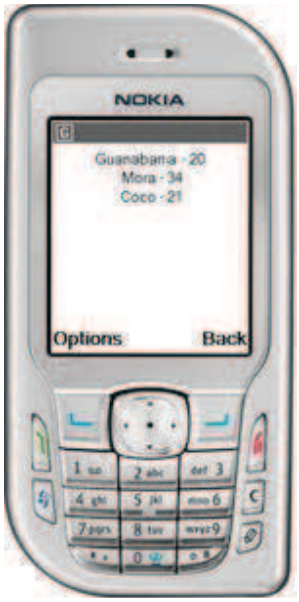
A través de un celular con Internet se dará el ingreso a la parte WAP. Al igual que en el sistema cliente servidor es necesario que el Usuario se identifique, ingresando su nombre de usuario y passwordpara verificar su existencia en el sistema.

Si el usuario no existe en el sistema, o si ingresa una clave errónea, se mostrará la siguiente pantalla de error impidiendo su ingreso:



Una vez que el usuario ingresa al sistema, se presenta el menú, donde se tienen las opciones: Consultar stock e Ingresar pedido.

Al elegir la opción Consultar stock, se presentará la siguiente información:



En esta pantalla aparecerán los artículos que tienen stock, seguidos de la cantidad de stock disponible. Las cantidades se presentarán por default en la unidad de stock que tenga parametrizado cada artículo en el sistema. Si existen artículos con stock disponible cero, no los presentará.

Si se ingresa a la segunda opción: Ingresar pedido, se presenta la siguiente pantalla:

Se debe ingresar los campos:

- Identificación: número de identificación del cliente como por ejemplo: cédula, RUC, pasaporte u otro.

- Producto: el nombre del artículo por el que se está realizando el pedido.

- Cantidad: tomando en cuenta que está dado por la unidad de stock parametrizada, se ingresa la cantidad pedida.

Si el usuario desea ingresar el pedido debe pulsar Aceptar, si por el contrario no desea ingresarlo, aplastará en la opción Cancelar.

5.5.MANUAL DE INSTALACIÓN

5.5.1. REQUERIMIENTOS DEL SISTEMA

a) Sistemas Operativos compatibles:

- Windows XP con Service Pack 2 (mínimo)
- Windows Vista
- Windows 7
- Windows Server 2003

b) Procesador:

- Procesador Pentium VI, Velocidad 2.6 Ghz (equivalente y/o superior).

c) RAM:

- 512 Mb (mínimo).

- 1 Gb o superior (recomendado).

d) Disco Duro:

- 5 Gb disponible en su disco duro (mínimo).

e) Monitor:

- 1024 x 768, 256 colores (mínimo).

- Color de alta densidad de 1024 x 768, 32 bits (recomendado).

f) Tarjeta de Red:

- Tarjeta de red a velocidad 10/100 o superior.

5.5.2. INSTALACIÓN

El proceso de instalación del programa se divide en 2 pasos:

- Comprobación e instalación de requisitos previos.

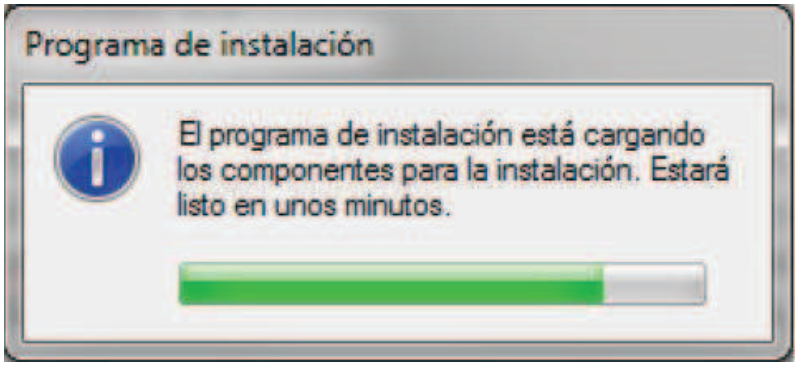
- Instalación del programa CRM – WAP.

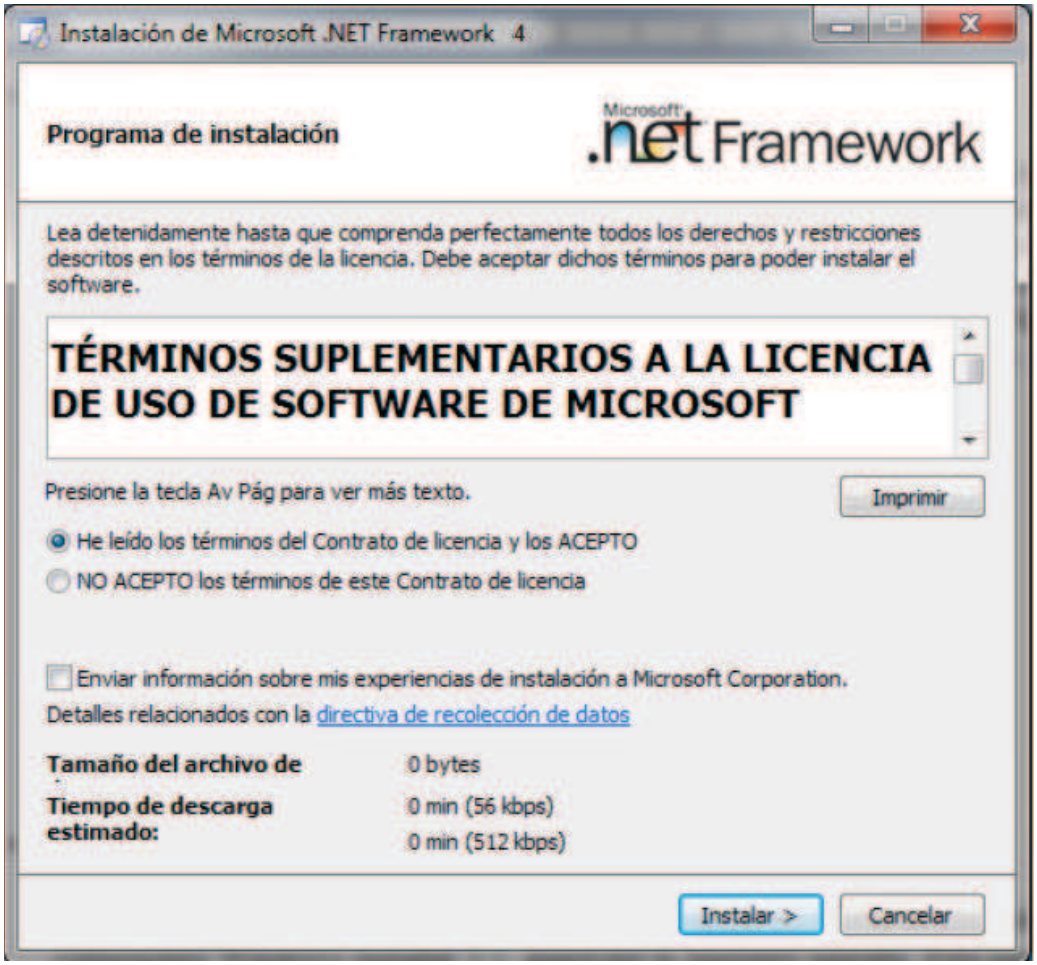
5.5.3. COMPROBACIÓN E INSTALACIÓN DE LOS REQUISITOS PREVIOS

El sistema CRM – WAP, requiere para ejecutarse, que en su ordenador se encuentren instalados los siguientes componentes de Microsoft:

a) Microsoft .NET Framework 4

Si el programa de instalación detecta que no se encuentra instalado el componente Microsoft .NET Framework 4, aparecerá la siguiente pantalla:





Esta pantalla muestra el acuerdo de licencia para instalar el componente Microsoft .Net Framework 4. Si se pulsa aceptar se iniciará la instalación. Este proceso puede durar entre 3 y 15 minutos. Una vez finalizada la instalación del componente se continuará con la instalación del programa.

b) Windows Installer 3.1

Si el programa de instalación llega a detectar que no se encuentra instalado el componente Windows Installer 3.1, aparecerá la siguiente pantalla:

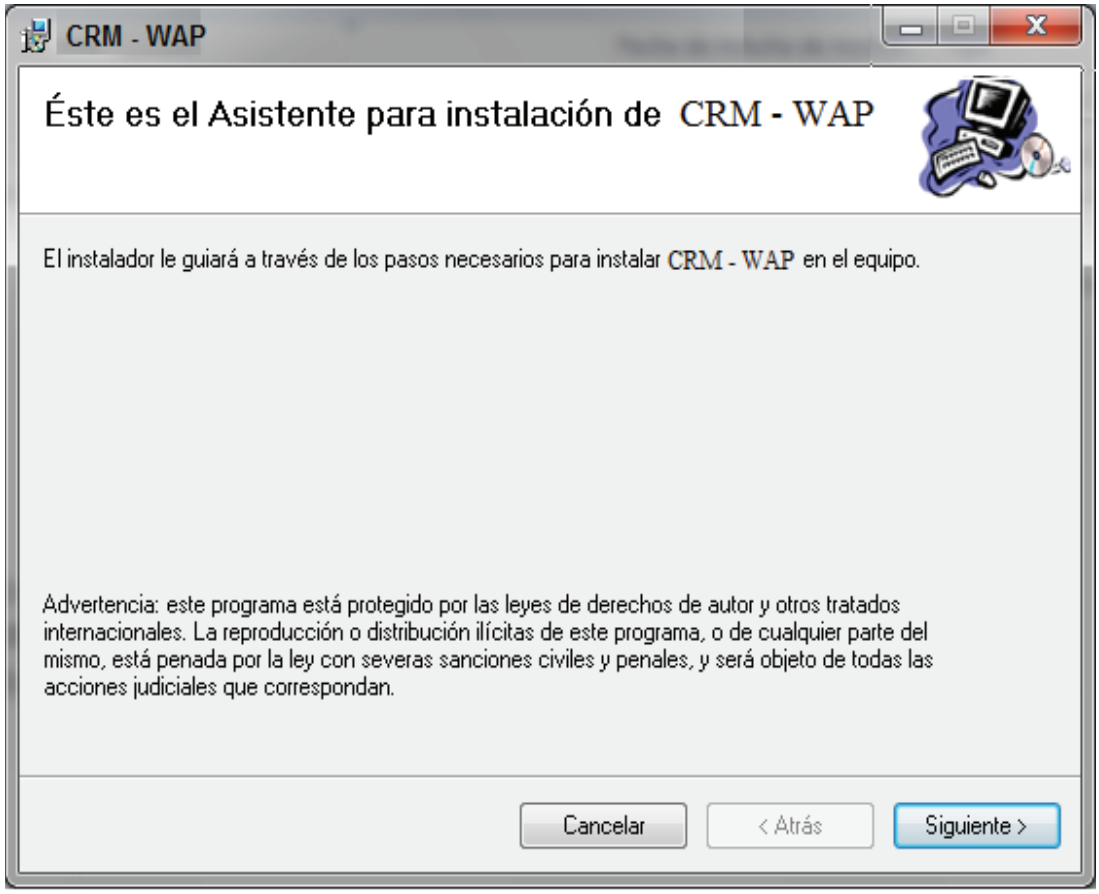
Esta pantalla muestra el acuerdo de licencia para instalar el componente. Si se pulsa aceptar se iniciará la instalación. Este proceso puede durar entre 3 y 5 minutos.

Una vez finalizada la instalación del componente se continuará con la instalación del programa. Si tiene que instalar alguno de estos componentes, es posible, que tenga que reiniciar el ordenador.

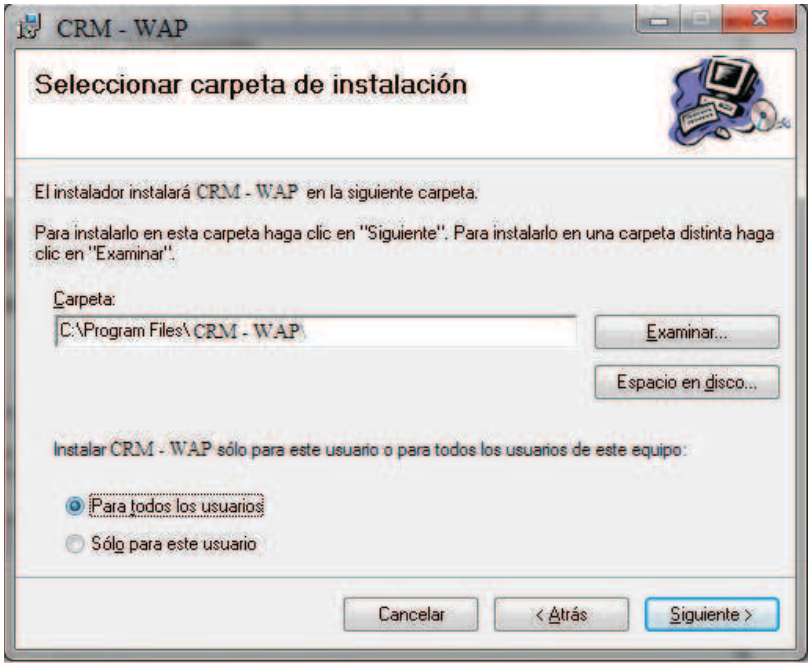
5.5.4. INSTALACIÓN DEL PROGRAMA CRM – WAP

Una vez comprobado que se encuentran instalados los requisitos previos, necesarios para el funcionamiento del programa, comienza la instalación.

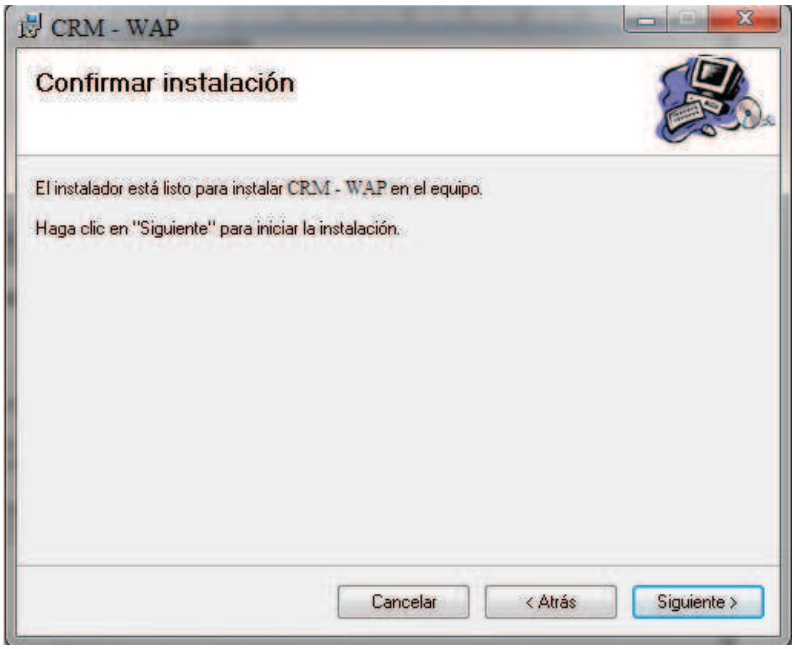
Ejecución del ícono Setup.exe para ejecutar el instalador de CRM – WAP.

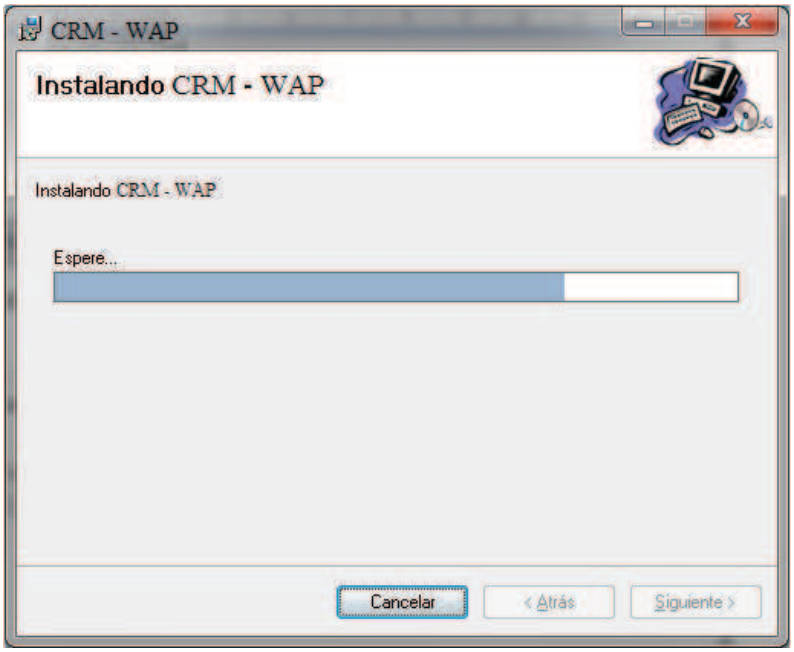


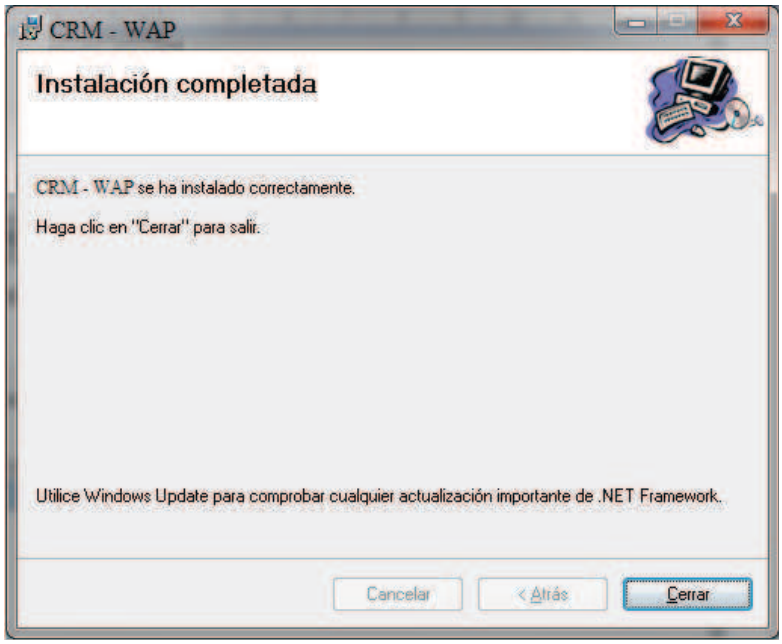
El programa se instala por defecto en el directorio C:\Program Files\CRM – WAP.



Posteriormente dar click en siguiente, esperando que finalice la instalación y cerrar el último cuadro de diálogo.







Si la instalación finaliza correctamente, el ícono de CRM – WAP, aparecerá en el escritorio y también en el menú inicio, como un nuevo programa instalado.

Conclusiones

Uno de los problemas iniciales en un proyecto de desarrollo de software es encontrar la forma adecuada para documentar los requisitos del usuario. En esta área existe una plantilla conocida como la norma IEEE 830, la misma que es resultado de la recopilación de las buenas prácticas de la industria de software y que además no es rígida en su utilización, por lo que es muy buena idea utilizar esta aproximación para no perder tiempo con las generalidades del tema.

Al finalizar el presente trabajo se concluye que, los archivos DLL's son de gran importancia, ya que agrupa las funciones más utilizadas, las cuales son llamadas desde distintas clases, ahorrando espacio en memoria, disco y reduciendo el intercambio de páginas.

En la codificación de la aplicación se evidenció que el modelamiento de una aplicación orientada a objetos debe realizarse de las clases a las tabla, y no de tablas a clases como tradicionalmente se hace, el uso de Hibernate puede hacer que en base a ingeniería inversa pueda crearse la base de datos automáticamente desde la capa de negocio con lo que el modelo orientado a objetos es más preciso. Además de esta manera no se limita las características de la orientación a objetos en la fase de diseño.

El manejo de la información centralizada mediante el sistema, ayudó de una manera muy eficaz al momento de tomar decisiones, para cada uno de los procesos en tiempo de ejecución, permitiendo con esto un mejoramiento notable en el tiempo de respuesta ante problemas, o casos aislados de lo común.

En la actualidad las aplicaciones WAP se han abierto campo en el mercado y son cada vez más valorizadas, debido a que el uso de dispositivos móviles se ha vuelto frecuente y cumple con una gran cantidad de objetivos propios de la vida cotidiana.

La principal dificultad encontrada en la realización de este aplicativo móvil, fue encontrar una herramienta que permita diseñar una página WAP para varios dispositivos, sin caer en un nicho específico, cuya instalación no demande mayores recursos ni el levantamiento de un ambiente complejo.

Referente al emulador utilizado, WAPPROOF, se concluye que cumple con las necesidades para las que fue creado ya que haciendo uso del mismo se pudo realizar todas las pruebas correspondientes al avance del desarrollo de la aplicación.

Recomendaciones

Se recomienda el uso de la Ingeniería de Requisitos, ya que a pesar de ser un proceso largo y arduo, es muy importante identificar a todas las personas implicadas, considerar sus necesidades y asegurar que entienden las implicaciones de los nuevos sistemas.

Se recomienda y se incentiva a la utilización de archivos DLL's, ya que es más fácil de manejarlos, cuando cambien las funciones de un archivo DLL, simplemente se debe compilar y vincular las aplicaciones que las utilizan, si no cambian los argumentos o valores devueltos de la función.

Sin duda el desarrollo de software es tan dinámico y complejo que no puede ser cubierto en su totalidad en un único trabajo, es por eso que del aprendizaje obtenido durante la investigación se puede sugerir mejoras para el sistema que por el alcance del mismo y el tiempo necesario para hacerlo no se encuentran como parte de este proyecto.

En nuevos proyectos se podría trabajar en estos aspectos:

- El encapsulamiento de funciones que realizan el mismo proceso para diferentes opciones, permitiendo agrupar elementos afines del sistema.

- Ocultación de información, ya que una subclase no necesita saber, la fuente de los parámetros que se le pasan como entrada, para que servirán las salidas de esta subclase, y que subprogramas se activan antes y después de él.

- Creación de un módulo de Facturación completo, con relación en las ventas realizadas, y el manejo de cartera.

- Que el módulo WAP, sea más amplio, añadiendo un menú donde el vendedor pueda crear nuevos clientes.

- La creación de un portal WEB, donde el cliente pueda verificar los datos de su compra, fechas de entrega, y valor a ser cancelado. Así como un historial de las compras realizadas por el cliente.

Se recomienda, una vez desarrollada una aplicación WAP, tomar en cuenta la realización de pruebas usando un emulador versátil, para evaluar el desempeño de la aplicación (Portabilidad, escalabilidad, etc), que permita determinar cuáles son sus ventajas, desventajas y características que servirán como guía para su mantenimiento o reingeniería.

Bibliografía

WILLEMEN, Robertus , School timetable construction : algorithms and complexity , 1ra Edición, Universiteit Eindhoven , Holanda, 2002.

CAMPS, Rafael, y otros, Bases de datos, 1ra Edición, UOC, España, 2005.

KENDALL, Kenneth y KENDALL, Julie, Análisis y Diseño de Sistemas, 6ta Edición, Pearson Educación, México, 2005.

QUIÑONES, Ernesto, “Leyendo las estructuras de las tablas”, 2009, 2009, [15 de diciembre 2010]. Disponible en la web:

http://www.eqsoft.net/blog/index.php?/archives/1479-Leyendo-las-estructuras-de-las-tablas.html#extended

IEEE, “Especificaciones de los requerimientos del software”, 1998, 1998, [10 de enero 2011]. Disponible en la web:

http://www.ctr.unican.es/asignaturas/is1/IEEE830_esp.pdf

KÜHNER, Jens, Expert .NET Micro Framework, 2nd Edición.

SHARP, John, Visual C# 2010, 1ra Edición, 2010 – Madrid.

Groussard, Thierry, C#4 Los fundamentos del lenguaje Desarrollar como visual 2010, 1ra Edición, 2011 – Paris.

ASHWORTH, Paúl, Getting, Started, with WAP WML, abril 2001.

LAUVER, Keith D., Desarrollo WAP con WML y WMLScript, 2001 – Anaya Multimedia.

RUMBAUGH, James, Modelado de Orientado a Objetos. Metodología OMT. 1ra Edición. España.

MILES, Rob, C# Programming, Edición 2.1 Enero 2011. Disponible en la web:

http://www.robmiles.com/c-yellow-book/Rob%20Miles%20CSharp%20Yellow%20Book%202010.pdf

ANEXO 1

Impresión en hoja pre-impresa de la factura.

ANEXO 2

Impresión de la auditoria del sistema.

Usuario.- código del usuario que accedió al sistema.

Fecha Ingreso.- fecha y hora de ingreso a la sesión que ejecuto el usuario.

Sesión.- código de la sesión a la que accedió el usuario.

Con.- true → ingreso solo a consultar.

Imp.- true → ingreso a imprimir.

Nue.- true → ingreso de un nuevo registro.

Act.- true → ingreso actualizar registro.

Eli.- true → ingreso a eliminación de registro.

Fecha Salida.- fecha de salida de la sesión.