Universidad del Bío-Bío

Profesor Guía: **Sr. Sergio Bravo Silva**

Facultad de Ciencias Empresariales

Departamento de Sistemas de Información

UNIVERSIDAD DEL BÍO-BÍO

"Reingeniería y migración en Red de una aplicación generalizada para prestación de servicios en línea a diferentes rubros del sector turismo."

Proyecto de Título presentado en conformidad a los

requisitos para obtener el Título de

Ingeniero Civil en Informática

Autores : **Oscar Del Campo Sepúlveda**

Jorge Vergara Vera

Concepción, 25 de febrero de 2010

Dedicatorias

A Dios, por permitirme ser quien soy y guiar mi vida.

A mis padres Oscar y Bernarda, por su fortaleza, apoyo, amor, y por inculcarme todos los valores que llevo conmigo.

A mi hermana Carolina, por todas sus enseñanzas, apoyo y cariño, durante toda mi vida.

A mi tía Marcela y mis abuelos José y Ana, por su apoyo y cariño.

A mis primos Camilo y Diana, los cuales considero como hermanos pequeños.

A mi sobrina Florencia, por ser una gran alegría para mí y mi familia.

Oscar

Gracias a Dios por mantenerme con la mente clara y concentrada en mis objetivos, y por darme la fuerza para lograrlos…

Este gran logro va dedicado a mis padres Jorge y María Cecilia, por su apoyo incondicional, sus consejos siempre necesarios y por el esfuerzo que realizaron día a día, año tras año…

También quisiera nombrar a mi querida abuela Eloisa (Q.E.P.D), que fue un pilar fundamental en mi desarrollo humano y profesional…

Jorge

Agradecimientos

A nuestras familias, por el gran apoyo brindado a lo largo de este camino, al igual que en el transcurso de nuestras vidas.

A nuestros compañeros, por sus grandes ayudas a lo largo de nuestro periodo dentro de esta institución.

A nuestro profesor, por ser un gran guía y trabajar junto a nosotros en este proyecto.

A nuestra Jefa de Carrera, por su disposición al momento de nuestras consultas y necesidades.

A nuestra universidad, por brindarnos la posibilidad de cumplir con nuestras metas y objetivos.

Resumen

El presente proyecto aborda la reingeniería de un antiguo sistema creado por un empresario de la región con apoyo de Corfo (Corporación de Fomento de la Producción), a través de un PROFO (Proyectos Asociativos de Fomento). El sistema orientado a las empresas pymes del sector turismo, intentaba solucionar algunos problemas de orden y rapidez en algunos procesos internos típicos de diversos rubros del sector. El sistema pretendía ser fácilmente aplicable a distintos rubros, sin modificar su código, lo que lograba en muy bajo grado. Funcionaba en forma irregular, sin contar con interfaces apropiadas, haciendo engorrosa su operación y por añadidura no funcionaba en red, lo que impedía la compartición de datos, sin cumplir con múltiples necesidades de las empresas.

La reingeniería se orientó a lograr satisfacer los requerimientos de estandarización y aplicabilidad del sistema a distintos rubros del sector. Fue necesario investigar sobre los temas estandarización y parametrización, sobre los cuales no existen experiencias difundidas. Con algunos elementos recopilados mas discusiones sobre estos temas se adquirieron conocimientos necesarios para realizar una estandarización apropiada de los procesos de rubros del sector, y un método que permitiera parametrizar cambios en el sistema a fin de hacerlo disponible para múltiples empresas dentro del sector turismo, sin cambios (o con cambios mínimos) del código generado. La tecnología que se ofrece hoy en día nos facilita el desarrollo de un sistema que cubra las características antes mencionadas.

Finalmente cabe señalar que al término de este proyecto se logró desarrollar un sistema en red, trabajando con un servidor, que maneja la base de datos, y múltiples clientes que se pueden comunicar con este.

El sistema desarrollado puede ser implantado dentro de variadas empresas con rubros del sector turismo, cumpliendo con la estandarización señalada. Igualmente se han construido métodos para la generación de datos de forma que la parametrización sea óptima.

En el futuro no es irracional pensar que este sistema pueda migrar a diferentes rubros fuera del sector turismo.

Índice

Capítulo 1: Introducción

En el mundo actual se hace cada vez más importante renovar y/o mejorar sistemas que presentan una cierta funcionalidad, debido a que estos, ya sea por el paso del tiempo o nuevas versiones realizadas, van quedando en el pasado. Este proyecto parte de la premisa de que existía un sistema creado hace algunos años atrás, el cual estaba orientado a múltiples empresas pymes del sector turismo, que no satisfacía variadas características, como compartir un sistema mediante una red de equipos computacionales. A esto se le debe añadir que funcionaba de forma irregular, con interfaces poco amistosas para el usuario, lo que hacía engorrosa su operación.

Debido a las limitadas características que poseía este sistema, es necesario realizar una reingeniería, para así gestionar nuevos cambios y añadir nuevas funcionalidades a éste.

De acuerdo a lo anterior, se hace necesario estudiar temas como la estandarización, la cual es necesaria para poder realizar un sistema que sea aplicable a distintos rubros del sector turismo. También, se debe considerar el tema de parametrización, ya que existen múltiples opciones que solo son necesarias en ciertas empresas, pero en otras no. Tomando en cuenta las consideraciones anteriores, es preciso desarrollar un sistema el cual sea capaz de trabajar en cualquier empresa perteneciente al sector turismo, sin importar el rubro que posea, ya sea un hotel o restaurant.

En el futuro no es irracional pensar que este sistema pueda migrar a diferentes rubros fuera del sector turismo, ya que todo negocio maneja similares procesos dentro de su funcionamiento.

Capítulo 2: Problemática

Según el INE, las cifras de hoy en día que se manejan respecto a la actividad turística receptiva demuestran un crecimiento mundial del 10%, distribuido desuniformemente entre las regiones (Asia y el Pacífico: 29% - Medio Oriente: 20% - Africa: 7% - Europa: 4%). Chile se encuentra dentro de los países con mayor crecimiento porcentual de llegadas anuales con un 26% con respecto a años anteriores.

La belleza natural, la diversidad y clima incomparable permiten que nuestro país sea cada vez una opción más atractiva para los turistas a la hora de escoger un buen lugar para visitar. No obstante la sustentabilidad de ese crecimiento, pasa por lograr que la calidad de la oferta de nuestros servicios de alojamientos y alimentación, aeropuertos, movilización y otros, sean compatibles con altos estándares internacionales exigentes.

El problema es cómo lograr mantener y hacer crecer nuestro turismo de llegadas a un cierto ritmo y a niveles sustentables. Ello plantea un desafío que debemos enfrentar como país y región: Generar condiciones que permitan lograr los altos estándares de calidad requeridos para lograr el propósito planteado. Ello implica una gestión interna que apunte a la optimización del uso eficiente de recursos y su protección de recursos, manteniendo el costo dentro de límites que aseguren un margen sobre la venta, y que permita el crecimiento del negocio al ritmo deseado.

2.1 Duplicidad de esfuerzo

Hoy en día, muchas empresas del sector turismo poseen sistemas de administración de reservas y venta de productos que son básicos o más bien arcaicos para la época en la cual vivimos. En muchas ocasiones se ha podido notar que pequeñas, e inclusive medianas empresas que se dedican a este rubro realizan reservas y ventas de sus productos solo registrando con papel y lápiz, método el cual es muy vulnerable, ya que es muy posible que esta información se

pierda, esto requiere la búsqueda de aquella información, perdiendo tiempo preciado para realizar atenciones a los demás clientes. Realizar procesos de esta manera también implica que es factible que los papeles se ensucien debido a caídas de sustancias, como por ejemplo líquidos o comida, lo cual puede distorsionar esta información. Algunas otras empresas realizan estas operaciones en planillas Excel, lo cual implica un cierto mayor orden y menores posibilidades de pérdida de información, pero como es sabido, esta herramienta es muy complicada a la hora de la búsqueda de información, ya que todo queda plasmado en una hoja de cálculo, lo cual genera desorden. Como se menciona anteriormente, este proceso también es complicado para realizar el proceso de búsqueda de información.

Como bien podemos apreciar, las opciones anteriores generar una duplicidad de esfuerzo en procesos como el descrito anteriormente, lo cual genera molestia en los clientes e inclusive en los propios operarios.

2.2 Evolución de la Empresa

La visión que posee cada empresa mayoritariamente está orientada al crecimiento y evolución de esta. Para poder lograr esta visión se necesita tiempo y trabajo. A lo largo del tiempo, las empresas requieren nuevos requerimientos, por lo que se deben manejar nuevos sistemas que logren satisfacer los requerimientos estipulados por la empresa.

Día a día, las empresas generan mayor competencia con sus pares, por lo cual una empresa que se queda en el pasado, poseyendo sistemas antiguos y que no cumplen con los nuevos requerimientos que implica pelear con la competencia, están destinadas a la extinción. Las empresas necesitan renovar sus herramientas de trabajo, para la mejor y más fácil manipulación de sus usuarios, y por sobre todo para facilitar el trabajo y rapidez de este.

2.3 Deficiencias sistema antiguo

Ante todo, debemos mencionar que un sistema anteriormente realizado fue una primera versión de este proyecto, lo cual presentó numerosas insuficiencias en su realización. A continuación se nombran múltiples motivos por lo cual realizar una reingeniería y migración:

- El sistema no presentaba una correcta correspondencia de estandarización y parametrización, ya que se hizo un intento de esto, pero este trabajo se realizó de manera insuficiente.
- El sistema no presentaba una interfaz amigable, por lo que resultaba engorrosa la utilización del sistema.
- El sistema no estaba habilitado para ser operado dentro de una red de computadores, lo que lo hacía bastante limitado en comparación a otros sistemas.
- En la operación del sistema, no existían cuadros de búsqueda que permitieran encontrar fácilmente una cierta información.
- Al momento de abrir un Form dentro del sistema, este abría un nuevo icono en la barra de tareas, por lo que, en cada vez que se abría un nuevo Form, la barra de tareas se iban llenando de nuevos íconos, lo que resultaba molesto para los usuarios.
- Si al momento de abrir una rutina, esta ya estaba anteriormente abierta, igualmente se abría el Form, lo cual ocasiona que la pantalla se pueda llenar de Form con una misma rutina.
- El sistema presentaba un muy complejo manejo de perfiles de usuario y permisos de accesos autorizados.

- El sistema poseía una muy compleja forma de incluir nuevas funcionalidades.
- La mayoría de las eliminaciones que se realizaban en el sistema se hacían con eliminación física (directamente de la base de datos), lo cual generaba múltiples inconsistencias, así como también errores en la interfaz.
- Los nombres de las columnas de los Datagridview correspondían o coincidían con las de las tablas. En un cambio de rubro, el sistema seguía mostrando información del rubro anterior, lo cual generaba molestia en los usuarios.
- La venta de producto no estaba contemplada dentro del sistema.
- Al momento de realizar compras, estas no se podían comprar en lotes, sino que debían comprarse una a una, lo que generaba mucha molestia en los usuarios.

2.3.1 Problemáticas de trabajar en un solo computador

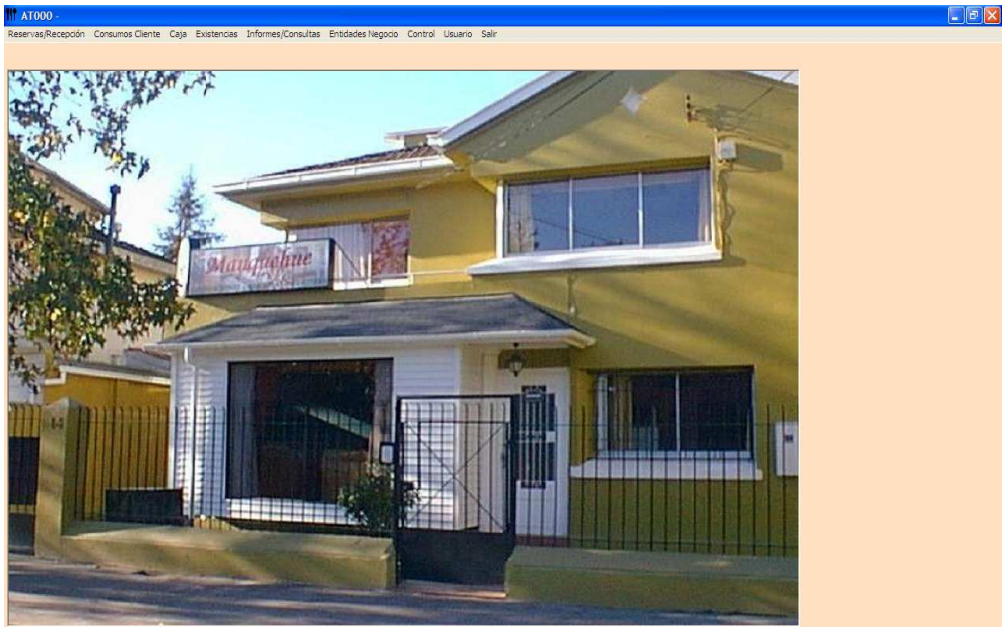
El sistema anteriormente señalado trabajaba en un solo equipo ya que no poseía la capacidad de ser operado en red. Se realizaron solucionar parches para lograr esta característica, dejando la base de datos replicada en 2 equipos, pero esto generaba múltiples problemas, como las que se describen a continuación:

- Inconsistencia en la base de datos, ya que a la postre, al estar la base de datos en 2 equipos, era como estar trabajando con 2 bases de datos distintas.
- Desorden, ya que había información que ingresaba dentro de una base de datos, pero que obviamente no ingresaba a la otra.

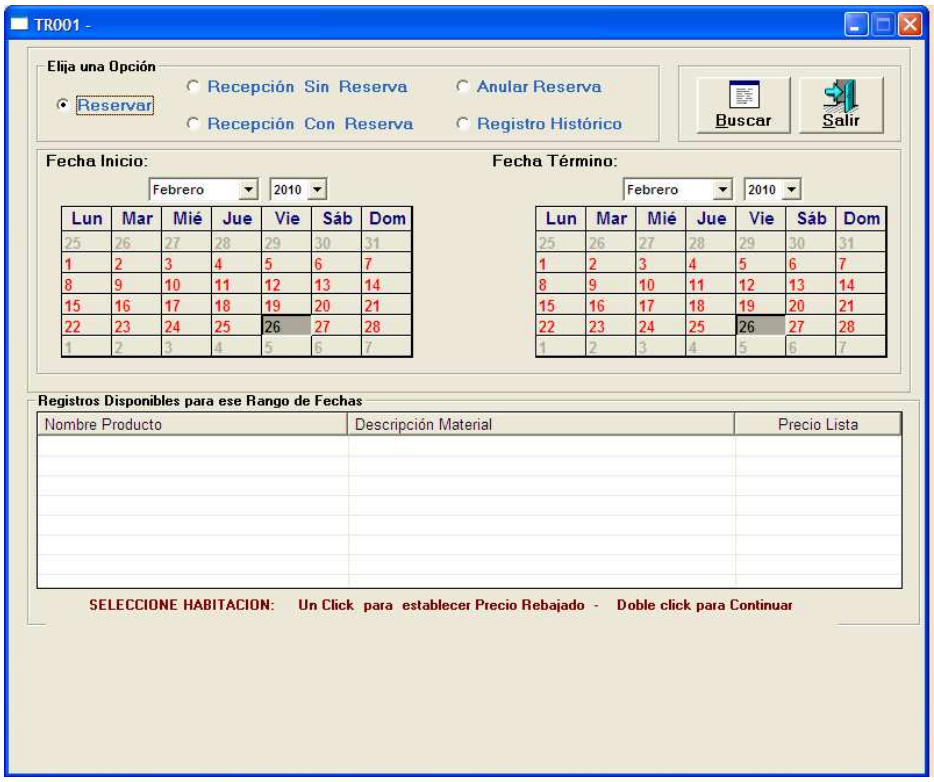
En conclusión, debido a las características anteriormente señaladas, el sistema comprendía un mal e inadecuado desempeño, por lo produjo que rápidamente se volviera a trabajar en un solo equipo.

2.3.2 Pantallas sistema antiguo

Menú Principal



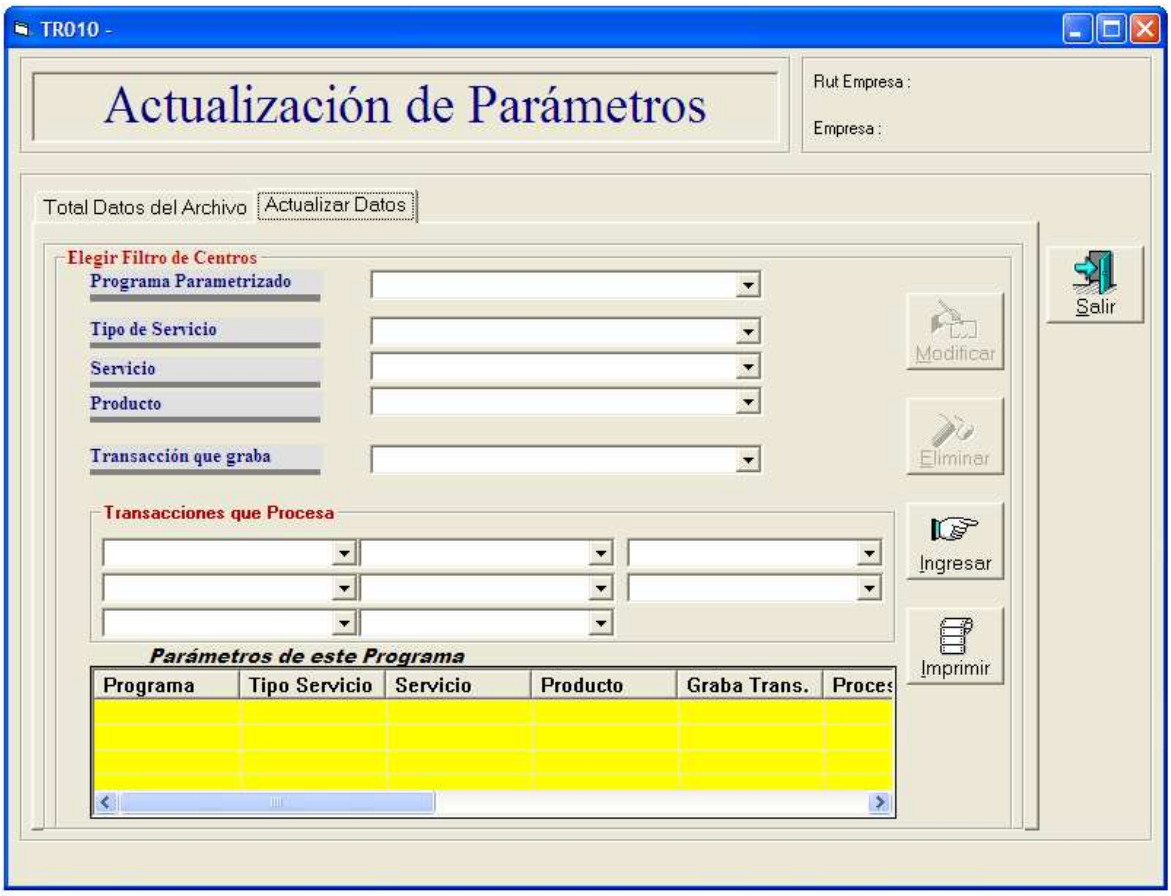
Reservas



Compras

Mantener Clientes

Parametrizador



Capítulo 3: Requerimientos de Solución

3.1 Requerimientos del usuario:

- Lograr un sistema con información integrada para apoyar la gestión. Muchas PYME hoy en día manejan mucha información de negocio a través de Word, Excel y papeles. De esta forma se busca reducir el nivel de utilización de estas herramientas, y se busca utilizar las herramientas que provee el propio sistema para la utilización de estos procesos, de forma de integrar la información dentro de un mismo sistema.

- Proveer de información organizada especialmente para facilitar decisiones en el área operacional y de gestión. Las PYME que manejan información en papel y tablas no disponen de herramientas especiales para el apoyo a la gestión, como por ejemplo reportes estadísticos. Se busca apoyar a la PYME distribuyendo esta información a través de gráficos y reportes que ayuden al nivel operacional y de gestión de la PYME.

- Un sistema intuitivo que permita una auto-capacitación del usuario. El sistema es de fácil manejo por lo que el usuario promedio no debería tener problemas en la utilización de éste. Además al momento de instalar el sistema en la empresa se realiza una capacitación, para que el usuario pueda manejar sin problemas el sistema.

- Un sistema provisto a través de red, accesible a muy bajo costo, para diversas pequeñas empresas. El sistema está implementado para ser manejado a través de una red, en la cual no solo se pueda ocupar el sistema desde un sitio determinado. Este sistema, busca proveer a las pequeñas empresas, es por esto que no se busca una gran utilidad en la venta de este sistema.

- Un sistema que libere una gran cantidad de tiempo que las PYME pierden en la búsqueda y manejo de su información. El sistema contiene toda la información necesaria de los productos, y de la utilización de estos, esto ayuda a la rapidez con la cual se realizan los procesos de prestaciones de servicios o búsqueda de información en general.

- Un sistema aplicable a numerosas PYME que sea fácilmente adaptable a cada una de ellas y fácil de instalar, migrando sus datos y procesos funcionales al sistema. Es la idea principal que se busca con la estandarización y parametrización, con la cual el sistema sea fácilmente adaptable a cualquier tipo de empresa PYME del sector turismo, siendo que algunas empresas poseen diversas características o procesos que otras no. Además se busca que el sistema comprenda un fácil nivel de instalación, para la facilidad del usuario en este proceso.

3.2 Requerimientos que debe cumplir el sistema para ser operado amistosamente dentro de una empresa:

- Poseer Interfaz amigable, con la cual el usuario pueda trabajar de forma amena y clara con el sistema, de forma de no confundir a este en la utilización del sistema.

- Poseer un nivel de privacidad apropiado para la utilización segura del sistema, obteniendo privilegios para cada nivel de usuario.

- Poseer una modularización del sistema, en la cual algunas empresas poseerán algunos módulos que no serán necesarios dentro de otra

empresa, esto ayuda a la innecesaria población de módulos dentro de un sistema destinado a una determinada empresa.

- Realizar transacciones de calidad, en donde se pueda confiar en que cada transacción se realizará como se estipula. El gestor de base de datos debe poseer altos niveles de estabilidad y confiabilidad para garantizar esta característica.

- Operar el sistema a una velocidad admisible. Los tiempos entre actualizaciones y movimiento dentro del sistema debe ser de forma rápida, pues dentro de una empresa los tiempos de atención son muy importantes.

- Poseer un nivel de consistencia óptimo para trabajar con distintos equipos dentro de una misma red. La base de datos debe ser robusta y ser actualizada muy rápidamente cada vez que un equipo dentro de la red realiza funciones de incorporación, edición o eliminación de información.

- Poseer transparencia en la base de datos. El gestor de base de datos debe ser transparente para el usuario, permitiendo que este solo opere con la información provista por la base de datos, pero sin operar dentro del gestor de base de datos.

3.3 Requerimientos técnicos que debe poseer una empresa para hacer uso del sistema:

- Posesión de licencia para la utilización de VisualBasic.Net, ya que el sistema está implementado en este lenguaje de programación, por lo cual la licencia es necesaria.

- Utilización de plataforma Microsoft. Para la correcta utilización del software, es necesario poseer el sistema operativo Microsoft Windows, ya sea en sus versiones XP, Vista o 7. Esto asegura el correcto funcionamiento del sistema, ya que al ser desarrollado en VisualBasic.Net no puede ser ocupado en otro sistema operativo.

- Utilización del gestor de base de datos Postgre SQL, necesario para el correcto funcionamiento de la base de datos, desarrollado en este sistema.

- Posesión de hardware admisible para la utilización del sistema. Las características de hardware mínimo se detallan a continuación:

    - Memoria RAM 256 MB o superior.
    - Disco Duro 80 GB o superior.
    - Procesador 1.6 GHz o superior.
    - Tarjeta de Red.
    - Monitor.
    - Teclado y Mouse.

- Utilización de múltiples computadores dentro de una misma red, para satisfacer las características del sistema en cuanto a la utilización de red.

3.4 Requerimientos operacionales que debe poseer los usuarios de una empresa para hacer uso del sistema:

Poseer personal el cual esté capacitado para manejar el sistema de forma en que este pueda manejar, a lo menos, las funciones básicas del sistema.

Capítulo 4: Objetivos y soluciones

4.1Objetivos

4.1.1 Objetivo general:

Poner a disposición de las pequeñas empresas de diferentes rubros del sector Turismo, una aplicación que apoye su gestión, a costo mínimo, mediante su membrecía a un servicio en red, provisto por un sistema creado por el presente proyecto.

4.1.2 Objetivos específicos:

- Disponer de una aplicación destinada al sector Turismo, parametrizable para adaptarla, sin cambio de código a diversos rubros del sector Turismo.

- Incluir en el Sistema módulos estándares acorde a las disposiciones legales y prácticas comunes en la generación de Información.

- Facilitar el proceso de fabricación de productos, generando apoyo al control de ingredientes y de costos unitarios por producto.

- Permitir el control de compras y control de existencia para activos de diferente naturaleza en una o más bodegas.

- Apoyar el manejo de Cuentas Corrientes de distintos tipos de Clientes.

- Manejo de listas de precios de productos, para ser incluidos automáticamente en el proceso de Facturación.

- Generación de libro de compras y ventas a fin de permitir la declaración mensual del IVA.

4.2 Soluciones

4.2.1 Estandarización

Un antiguo anhelo de los desarrolladores de SW, era disponer de métodos que permitieran el desarrollo de software a la medida de la empresa, sin desaprovechar la experiencia lograda en desarrollos anteriores, en la medida que se lograra garantizar una calidad cada vez mejor de los sistemas resultantes, con un esfuerzo y costos menores, no solo de construcción, sino del necesario mantenimiento futuro, y de riesgos asociados.

Hoy en día, las tecnologías disponibles y el conocimiento adquirido sobre este tema, nos permite definir una estructura con la cual se pueden desarrollar sistemas de software orientados a múltiples empresas con rubros similares.

¿Qué es la Estandarización?

La estandarización o normalización es la redacción y aprobación de normas que se establecen para garantizar el acoplamiento de elementos construidos independientemente, así como garantizar el repuesto en caso de ser necesario, garantizar la calidad de los elementos fabricados y la seguridad de funcionamiento.

La asociación estadounidense para pruebas de materiales (ASTM), define la normalización como el proceso de formular y aplicar reglas para una aproximación ordenada a una actividad específica para el beneficio y con la cooperación de todos los involucrados.

Según la ISO (International Organization for Standarization) la Normalización es la actividad que tiene por objeto establecer, ante problemas reales o potenciales, disposiciones destinadas a usos comunes y repetidos, con el fin de obtener un nivel de ordenamiento óptimo en un contexto dado, que puede ser tecnológico, político o económico.

La normalización persigue fundamentalmente tres objetivos:

- Simplificación: Se trata de reducir los modelos quedándose únicamente con los más necesarios.
- Unificación: Para permitir la intercambiabilidad a nivel internacional.
- Especificación: Se persigue evitar errores de identificación creando un lenguaje claro y preciso.

Las elevadas sumas de dinero que los países desarrollados invierten en los organismos normalizadores, tanto nacionales como internacionales, es una prueba de la importancia que se da a la normalización.

Para qué nos sirve las Estandarización

La mayoría de los sistemas de desarrollo de software que se generar hoy en día realizan la completa labor de comenzar desde cero y no poder aprovechar múltiples elementos desarrollados anteriormente. Lo más común es comenzar aplicando el método de intento-error, de gran atención entre los novatos. En este método cada concepto y esfuerzo se considera como nuevo, desestimando la experiencia anterior.

- La estandarización de modelamiento de procesos y datos es materia de preocupación mundial a nivel universitario y de la industria, dado que un estándar de buen nivel conduce a un correcto y más completo tratamiento de la información. El estandarizar modelos no significa en modo absoluto la pérdida del carácter personalizado de los sistemas, los cuales en todo caso deben cumplir los requerimientos propios de cada usuario. De ser insuficiente el modelo estándar para cubrir situaciones específicas, debería mejorarse.

- Los lenguajes de programación también tienden a estandarizarse, mediante la inclusión por parte de los fabricantes, de librerías para presentación de pantallas, informes y actualización de bases de datos, lo que redunda en menor tiempo de programación (y normalmente mayor tiempo de procesamiento).

Algunos estándares que es urgente aplicar son los que se refieren a seguridad de los datos, materia en la cual con frecuencia los sistemas fallan.

El o los profesionales necesarios para lograr modelos estandarizados deben reunir los siguientes atributos:

- Capacidad para reconocer la amplia variedad de transacciones comunes presentes en los diversos nichos de negocios como: Adquisición – Venta – Almacenamiento, etc, y su tratamiento.
- Capacidad y experiencia para agrupar los negocios en grupos que contengan tipos de transacciones de comportamiento general similar.
- Capacidad de reconocer variabilidades propias de cada negocio en particular respecto de los estándares modelados.

Un ejemplo de cómo podría reducirse el esfuerzo y costo de desarrollos sucesivos de sistemas de Empresas de rubros similares se incluye en la siguiente página.

Reducción en el precio del desarrollo como consecuencia de la estandarización

- Cada nuevo desarrollo de sistema basado en modelos estándares para un rubro dado muestra el siguiente cambio:

  a) Amortiza la fuerte inversión inicial
  b) Se mejora el Software, y experiencia
  c) Se reduce el costo de algunas actividades

- Variación de costo del desarrollo para 1 usuario (Base 100)

Etapa de Desarrollo	Primera vez	Segunda Vez	Tercera y siguientes
1- Levantar Requerimientos	10	10	10
2- Modelar Procesos	10	6	4
3- Modelar Datos	10	6	4
4- Programación y pruebas	30	15	10
5- Capacitación y manuales	15	15	15
6- Instalalar	5	5	5
7- Poner en Marcha	10	10	10
8- Soporte de Garantía	10	5	5
	100	65	59

- Debería determinarse un incremento de costo por cada usuario adicional del sistema.
- Existe incremento de costos cuando los requerimientos varían en forma significativa.

Alcances de la Estandarización en el Sistema ha desarrollar:

Este sistema está destinado mayoritariamente a las empresas u organizaciones que se encargan de recibir reservas y realizar ventas o arriendos de sus productos. Es por esto, que el sistema está en gran parte orientado al sector turismo, pero esto no impide que muchas otras empresas dedicadas sólo a la venta de productos puedan ser participes de obtener este sistema sin la necesidad de cambiar su estandarización.

Los rubros de empresas que permiten la utilización de la Estandarización de módulos, en una primera etapa del proyecto, son los siguientes:

- Hotelería
- Restaurant
- Camping
- Hostelling
- Café
- Pastelería
- Cyber-Café

4.2.2 Parametrización

Dentro de los sistemas que se utilizan hoy en día, la información que se despliega a través de las distintas interfaces debe ser correcta y exacta, sin la necesidad de que muestren datos innecesarios y que no correspondan al proceso el cual es ejecutado

¿Qué es la Parametrización?

La Parametrización es la propiedad de un módulo, o de una construcción sintáctica del lenguaje, para utilizar datos de varios tipos. Es un mecanismo muy útil porque permite aplicar el mismo algoritmo a tipos de datos diferentes; es una facilidad que permite separar los algoritmos de los tipos de datos, aumentando de esta manera la modularidad de los programas y minimizando la duplicación de código. En Ada a la parametrización se la llama Genericidad y se logra mediante el uso de Paquetes Genéricos (generic packages), y en C++ mediante el uso de Plantillas (templates).

Objetivo de la Parametrización

El objetivo de la parametrización es lograr el máximo nivel de flexibilidad del Sistema, permitiendo cambios sin necesidad de agregar o modificar líneas de Código.

Alcances de la Parametrización en el sistema ha desarrollar

En el caso de nuestro sistema, la parametrización permitirá filtrar elementos a considerar en las interfaces, dependiendo del tipo de negocio en que se instale

el software. Esto constituirá una potencia importante del sistema, al permitir instalarlo en diferentes tipos de negocio del sector turismo.

Este sistema contendrá 2 grandes tipos de parametrización, las cuales se detallan a continuación:

- Dependiendo del tipo de negocio en el cual se implementará el sistema, el software dispone de distintos módulos que serán seleccionados por el desarrollador en el momento de implementación del software, utilizando recursos proporcionados por visual Basic. El software dispone de módulos suficientes para ser implementados en los distintos tipos de empresas del sector turismo. Por ejemplo, si el sistema va dirigido para un negocio de tipo restaurante, se omitirán los módulos destinados a funciones que no corresponden a las de un restaurante, pero sí se incluirán los módulos que atienden funciones comunes a cualquier negocio.

- Dependiendo del tipo de negocio en donde se está implementando el sistema, las opciones que se mostrarán dentro del sistema serán parametrizadas a partir de las distintas opciones que se elijan dentro del sistema. Por ejemplo, si el sistema se está desarrollando para el ámbito hotelero, el cual permite la de reserva de habitaciones y también la venta de productos de tipo restaurante, dentro del sistema se podrá seleccionar el tipo de servicio que se desea dar en el momento, así, si el tipo de servicio que se seleccionará es reserva de habitaciones, los servicios que se cargaran a partir del tipo de servicio serán los destinados solo a ese tipo de servicio y no a los demás tipos de servicio, como por ejemplo restaurante.

El sistema incluye un módulo que permite parametrizar las subopciones a incluir en cada opción funcional seleccionada. Esto permite por ejemplo, filtrar los tipos de productos que se incluirán al seleccionar la opción de venta que un determinado negocio expende.

Nuestro sistema tendrá 3 atributos fundamentales, con los cuales se puede lograr un buen nivel de parametrización, estos atributos dependerán del rubro en el cual se va a trabajar.

- Tipo de Servicio: Indica la descripción que posee el rubro propiamente tal, por ejemplo, hotelería, alojamiento, etc.
- Servicio: Indica los servicios que se presentan dentro de la empresa, por ejemplo, cabañas, habitaciones, Restaurant, lavandería.
- Producto: Indica los productos propiamente tal que ofrece la empresa u organización, por ejemplo en el caso de un hotel, las camas, las bebidas, las camisas que se desean lavar.

Obviamente, si una empresa posee 2 o más tipos de servicios, al momento de seleccionar un tipo de servicio específico, dentro de la rutina de servicio solo se desplegarán las opciones pertenecientes a ese tipo de servicio. Lo mismo ocurre con Productos, ya que solo se despliegan las opciones de Productos pertenecientes al servicio especificado.

Capítulo 5: Herramientas y enfoques metodológicos a Utilizar

5.1 PostgreSQL

5.1.1 ¿Qué es PostgreSQL?

PostgreSQL es un sistema de gestión de base de datos relacional orientada a objetos y libre, publicado bajo la licencia BSD.

Como muchos otros proyectos de código abierto, el desarrollo de PostgreSQL no es manejado por una sola empresa sino que es dirigido por una comunidad de desarrolladores y organizaciones comerciales las cuales trabajan en su desarrollo. Dicha comunidad es denominada el PGDG (*PostgreSQL Global Development Group*).

Es un motor de base de datos:

- Avanzado.
- De Código Abierto.
- Sistema de Gestión de Base de Datos Relacional.

5.1.2 Ventajas de PostgreSQL.

- **Instalación ilimitada.**
  En múltiples ocasiones las bases de datos comerciales son instaladas en más servidores de lo que se permite con una cierta licencia. Algunos proveedores comerciales consideran esto como la principal fuente de incumplimiento de licencia. PostgreSQL, al ser libre, no realiza ningún tipo de acción contra el propietario, como demandas o acciones legales asociadas, puesto que PostgreSQL no lleva un costo asociado a la licencia del software.

Esto tiene varias ventajas adicionales:

- Modelos de negocios más rentables con instalaciones a gran escala.
- No existe la posibilidad de ser auditado para verificar cumplimiento de licencia en ningún momento.
- Flexibilidad para hacer investigación y desarrollo sin necesidad de incurrir en costos adicionales de licenciamiento.

- **Posee mejor soporte en comparación a los proveedores comerciales.**

  PostgreSQL ofrece múltiples opciones de soporte on-line, además de una comunidad de profesionales y usuarios entusiastas que poseen y/o conocen múltiples opciones sobre este tema. Además existen especialistas particulares en el tema que pueden solucionar problemas o inconvenientes en cuanto a PostgreSQL.

- **Ahorro en costo de operación.**

  El software ha sido diseñado y creado para tener un mantenimiento y ajuste mucho menor en comparación a los productos de los proveedores comerciales, conservando todas las características, estabilidad y rendimiento.

  Los programas de entrenamiento de usuario son reconocidamente mucho menos costosos, así como también mucho más efectivos, manejables y prácticos en el mundo real, en comparación a los principales proveedores comerciales.

- **Estabilidad y confiabilidad.**

  En contraste a muchos sistemas de base de datos comerciales, es extremadamente común que compañías reporten que PostgreSQL nunca ha presentado caídas en varios años de operación de alta calidad.

- **Extensible.**

  El código fuente está disponible para todos sin costo. Si un usuario necesita extender o personalizar PostgreSQL de alguna manera, pueden hacerlo con un mínimo esfuerzo, sin costos adicionales. Esto es complementado por la comunidad de profesionales de PostgreSQL alrededor del mundo, que también extienden PostgreSQL periódicamente.

- **Multiplataforma.**

  PostgreSQL puede ser ejecutado tanto en Windows, GNU/Linux o Mac OS, lo cual facilita su operación.

- **Alta concurrencia.**

  Mediante un sistema denominado MVCC (Acceso concurrente multiversión) PostgreSQL permite que mientras un proceso escribe en una tabla, otros accedan a la misma tabla sin necesidad de bloqueos. Cada usuario obtiene una visión consistente de lo último a lo cual se le realizó un commit. Esta estrategia es superior al uso de bloqueos por tabla o por filas, común en otras bases, ya que elimina la necesidad del uso de bloqueos explícitos.

- **Múltiples Características técnicas.**

  PostgreSQL posee un gran número de características técnicas en comparación a proveedores comerciales. A continuación se detalla una lista de las principales características técnicas que posee PostgreSQL:

  - Cumple completamente con ACID.
  - Cumple con ANSI SQL.
  - Posee integridad referencial.
  - Replicación (soluciones comerciales y no comerciales) que permiten la duplicación de bases de datos maestras en múltiples sitios de réplica.
  - Puede realizar Vistas.
  - Puede realizar Triggers.
  - Puede realizar Secuencias.
  - Puede realizar Herencias.
  - Outer Joins.
  - Sub-selects.
  - Posee una API abierta.
  - Puede realizar Procedimientos almacenados.
  - Puede realizar un respaldo en caliente.
  - Bloqueo a nivel mejor-que-fila.
  - Índices parciales y funcionales.

- Soporte para consultas con UNION, UNION ALL y EXCEPT.
- Extensiones para SHA1, MD5, XML y otras funcionalidades.
- Funciones de compatibilidad para ayudar en la transición desde otros sistemas menos compatibles con SQL.

5.1.3 Razones para migrar desde Access a PostgreSQL

- **Funcionalidad:** PostgreSQL, a diferencia de Access, es un motor de bases de datos avanzado, el cual es capaz de cumplir con múltiples características que Access no posee.
- **Opción de poder trabajar en múltiples plataformas:** PostgreSQL tiene la característica de poder trabajar en múltiples plataformas, como Windows, GNU/Linux o Mac OS, no así Access, el cual solo puede ser utilizado en Windows.
- **Adherencia a estándares:** PostgreSQL cumple con todos los estándares SQL, con lo cual asegura un buen nivel de estabilidad y confiabilidad en este producto, sobre todo para bases de datos que son de gran tamaño.
- **Simplificación:** Gracias a sus herramientas gráficas como PgAdmin o phpPgAdmin la administración de la base de datos es sencilla.

5.2 Visual Studio

Al momento de desarrollar un sistema computacional, es necesario considerar las múltiples opciones que se disponen para desarrollar este sistema. El sistema de ayuda al sector turismo ha sido construido mediante Visual Studio 2008 utilizando Visual Basic y .NET Framework 3.5. Estas características fueron elegidas debido a que esta es una migración con variadas mejoras de un sistema

anterior, el cual estaba desarrollado en VisualStudio2005 y utilizaba .Net Framework 2.0

5.2.1 ¿Qué es Visual Studio?

Microsoft Visual Studio es un entorno de desarrollo integrado (IDE, por sus siglas en inglés) para sistemas operativos Windows. Soporta varios lenguajes de programación tales como Visual C++, Visual C#, Visual J#, ASP.NET y Visual Basic .NET, aunque actualmente se han desarrollado las extensiones necesarias para muchos otros.

Visual Studio permite a los desarrolladores crear aplicaciones, sitios y aplicaciones web, así como servicios web en cualquier entorno que soporte la plataforma .NET (a partir de la versión net 2002). Así se pueden crear aplicaciones que se intercomuniquen entre estaciones de trabajo, páginas web y dispositivos móviles.

5.2.2 Ventajas de Visual basic en Visual Studio

- Es un lenguaje RAD (desarrollo rápido de aplicaciones).
- Posee una curva de aprendizaje muy rápida.
- Integra el diseño e implementación de formularios de Windows.
- Permite usar con suma facilidad la plataforma de los sistemas Windows dado que tiene acceso prácticamente total a la API de Windows incluidas librerías actuales.
- El código en Visual Basic es fácilmente migrable a otros lenguajes.
- Es un lenguaje muy extendido por lo que resulta fácil encontrar información, documentación y fuentes para los proyectos.

- Fácilmente extensible mediante librerías DLL y componentes ActiveX de otros lenguajes.
- Posibilidad de añadir soporte para ejecución de scripts, VBScript o JScript, en las aplicaciones mediante Microsoft Script Control.
- Acceso a la API multimedia de DirectX (versiones 7 y 8). También está disponible, de forma no oficial, un componente para trabajar con OpenGL 1.1: VBOpenGL type library.
- Es un entorno perfecto para realizar pequeños prototipos rápidos de ideas.

5.3 Crystal Reports

Para la generación de informes de datos estadísticos dentro de la empresa, se requiere de una herramienta la cual sea capaz de generar esta información. La opción más clara, económica y positiva es Crystal Reports.

5.3.1 ¿Qué es Crystal Reports?

Crystal Reports se puede describir como una aplicación de inteligencia empresarial utilizada para diseñar y generar informes desde una amplia gama de fuentes de datos (bases de datos).

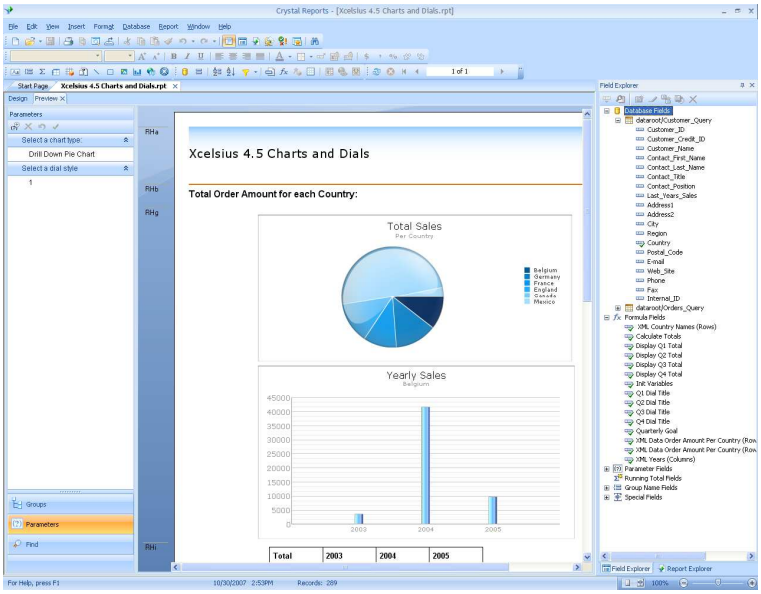


Figura en la cual se aprecia un reporte simple entregado
a través de Crystal Reports mediante Visual Studio

5.4 Arquitectura Cliente-Servidor

Con respecto a la definición de arquitectura cliente/servidor se pueden encontrar múltiples definiciones, de las cuales las más mencionadas son las siguientes:

- Cualquier combinación de sistemas, que pueden colaborar entre sí, para dar a los usuarios toda la información que ellos necesiten, sin que tengan que saber donde está ubicada.
- Es una arquitectura de procesamientos cooperativo donde uno de los componentes pide servicios a otro.
- Es un procesamiento de datos de índole colaborativo entre dos o más computadoras conectadas a una red.
- El término cliente/servidor es originalmente aplicado a la arquitectura de software que describe el procesamiento entre dos o más programas: una aplicación y un servicio soportante.

- IBM define al modelo Cliente/Servidor. "Es la tecnología que proporciona al usuario final el acceso transparente a las aplicaciones, datos, servicios de cómputo o cualquier otro recurso del grupo de trabajo y/o, a través de la organización, en múltiples plataformas. El modelo soporta un medio ambiente distribuido en el cual los requerimientos de servicio hechos por estaciones de trabajo inteligentes o "clientes", resultan en un trabajo realizado por otros computadores llamados servidores".
- "Es un modelo para construir sistemas de información, que se sustenta en la idea de repartir el tratamiento de la información y los datos por todo el sistema informático, permitiendo mejorar el rendimiento del sistema global de información".

En Resumen:

Cliente/Servidor es una relación entre procesos corriendo en máquinas separadas

El servidor es un proveedor de servicios.

El cliente es un consumidor de servicios.

Cliente y Servidor Interactúan por un mecanismo de pasaje de mensajes:

Pedido de servicio - Respuesta.

5.4.1 ¿Qué es una arquitectura?

Una arquitectura es un entramado de componentes funcionales que aprovechando diferentes estándares, convenciones, reglas y procesos, permite integrar una amplia gama de productos y servicios informáticos, de manera que pueden ser utilizados eficazmente dentro de la organización.

Debemos señalar que para seleccionar el modelo de una arquitectura, hay que partir del contexto tecnológico y organizativo del momento y, que la arquitectura

Cliente/Servidor requiere una determinada especialización de cada uno de los diferentes componentes que la integran.

5.4.2 ¿Qué es un cliente?

Es el que inicia un requerimiento de servicio. El requerimiento inicial puede convertirse en múltiples requerimientos de trabajo a través de redes LAN o WAN. La ubicación de los datos o de las aplicaciones es totalmente transparente para el cliente.

5.4.3 ¿Qué es un servidor?

Es cualquier recurso de cómputo dedicado a responder a los requerimientos del cliente. Los servidores pueden estar conectados a los clientes a través de redes LAN o WAN, para proveer de múltiples servicios a los clientes y ciudadanos tales como impresión, acceso a bases de datos, fax, procesamiento de imágenes, etc.

5.4.4 Características del modelo Cliente/Servidor.

En el modelo Cliente/Servidor podemos encontrar las siguientes características:

- El Cliente y el Servidor pueden actuar como una sola entidad y también pueden actuar como entidades separadas, realizando actividades o tareas independientes.

- Las funciones de Cliente y Servidor pueden estar en plataformas separadas, o en la misma plataforma.
- Un servidor da servicio a múltiples clientes en forma concurrente.
- Cada plataforma puede ser escalable independientemente. Los cambios realizados en las plataformas de los Clientes o de los Servidores, ya sean por actualización o por reemplazo tecnológico, se realizan de una manera transparente para el usuario final.
- La interrelación entre el hardware y el software están basados en una infraestructura poderosa, de tal forma que el acceso a los recursos de la red no muestra la complejidad de los diferentes tipos de formatos de datos y de los protocolos.
- Un sistema de servidores realiza múltiples funciones al mismo tiempo que presenta una imagen de un solo sistema a las estaciones Clientes. Esto se logra combinando los recursos de cómputo que se encuentran físicamente separados en un solo sistema lógico, proporcionando de esta manera el servicio más efectivo para el usuario final. También es importante hacer notar que las funciones Cliente/Servidor pueden ser dinámicas. Ejemplo, un servidor puede convertirse en cliente cuando realiza la solicitud de servicios a otras plataformas dentro de la red. Su capacidad para permitir integrar los equipos ya existentes en una organización, dentro de una arquitectura informática descentralizada y heterogénea.
- Además se constituye como el nexo de unión más adecuado para reconciliar los sistemas de información basados en mainframes o minicomputadores, con aquellos otros sustentados en entornos informáticos pequeños y estaciones de trabajo.
- Designa un modelo de construcción de sistemas informáticos de carácter distribuido. Su representación típica es un centro de trabajo (PC), en donde el usuario dispone de sus propias aplicaciones de oficina y sus

propias bases de datos, sin dependencia directa del sistema central de información de la organización, al tiempo que puede acceder a los recursos de este host central y otros sistemas de la organización que se ponen a su servicio.

En conclusión, Cliente/Servidor puede incluir múltiples plataformas, bases de datos, redes y sistemas operativos. Estos pueden ser de distintos proveedores, en arquitecturas propietarias y no propietarias y funcionando todos al mismo tiempo. Por lo tanto, su implantación involucra diferentes tipos de estándares: APPC, TCP/IP, OSI, NFS, DRDA corriendo sobre DOS, OS/2, Windows o PC UNIX, en TokenRing, Ethernet, FDDI o medio coaxial, sólo por mencionar algunas de las posibilidades.

Los clientes realizan generalmente funciones como:

- Manejo de la interfaz de usuario.
- Captura y validación de los datos de entrada.
- Generación de consultas e informes sobre las bases de datos.
- Por su parte los servidores realizan, entre otras, las siguientes funciones:
- Gestión de periféricos compartidos.
- Control de accesos concurrentes a bases de datos compartidas.
- Enlaces de comunicaciones con otras redes de área local o extensa.

5.4.5 Ventajas de utilizar la arquitectura Cliente/Servidor

- Centralización del control: Los accesos, recursos y la integridad de los datos son controlados por el servidor de forma que un programa cliente defectuoso o no autorizado no pueda dañar el sistema. Esta centralización también facilita la tarea de poner al día datos u otros recursos (mejor que en las redes P2P).

- Escalabilidad: Se puede aumentar la capacidad de clientes y servidores por separado. Cualquier elemento puede ser aumentado (o mejorado) en cualquier momento, o se pueden añadir nuevos nodos a la red (clientes y/o servidores).
- Fácil mantenimiento: Al estar distribuidas las funciones y responsabilidades entre varios ordenadores independientes, es posible reemplazar, reparar, actualizar, o incluso trasladar un servidor, mientras que sus clientes no se verán afectados por ese cambio (o se afectarán mínimamente). Esta independencia de los cambios también se conoce como encapsulación.
- Existen tecnologías, suficientemente desarrolladas, diseñadas para el paradigma de Cliente/Servidor que aseguran la seguridad en las transacciones, la amigabilidad de la interfaz, y la facilidad de empleo.
- Aumento de la productividad:
    - Los usuarios pueden utilizar herramientas que le son familiares, como hojas de cálculo y herramientas de acceso a bases de datos.
    - Mediante la integración de las aplicaciones Cliente/Servidor con las aplicaciones personales de uso habitual, los usuarios pueden construir soluciones particularizadas que se ajusten a sus necesidades cambiantes.
    - Una interfaz gráfica de usuario consistente reduce el tiempo de aprendizaje de las aplicaciones.
- Menores costes de operación:
    - Permite un mejor aprovechamiento de los sistemas existentes, protegiendo la inversión. Por ejemplo, la compartición de servidores (habitualmente caros) y dispositivos periféricos (como impresoras) entre máquinas clientes permite un mejor rendimiento del conjunto.

  - Proporcionan un mejor acceso a los datos. La interfaz de usuario ofrece una forma homogénea de ver el sistema, independientemente de los cambios o actualizaciones que se produzcan en él y de la ubicación de la información.
  - El movimiento de funciones desde un ordenador central hacia servidores o clientes locales origina el desplazamiento de los costes de ese proceso hacia máquinas más pequeñas y por tanto, más baratas.
- Mejora en el rendimiento de la red:
  - Las arquitecturas Cliente/Servidor eliminan la necesidad de mover grandes bloques de información por la red hacia los ordenadores personales o estaciones de trabajo para su proceso. Los servidores controlan los datos, procesan peticiones y después transfieren sólo los datos requeridos a la máquina cliente. Entonces, la máquina cliente presenta los datos al usuario mediante interfaces amigables. Todo esto reduce el tráfico de la red, lo que facilita que pueda soportar un mayor número de usuarios.
  - Tanto el cliente como el servidor pueden escalarse para ajustarse a las necesidades de las aplicaciones. Las CPUs utilizadas en los respectivos equipos pueden dimensionarse a partir de las aplicaciones y el tiempo de respuesta que se requiera.
  - La existencia de varias CPUs proporciona una red más fiable: un fallo en uno de los equipos no significa necesariamente que el sistema deje de funcionar.
  - En una arquitectura como ésta, los clientes y los servidores son independientes los unos de los otros, con lo que pueden renovarse para aumentar sus funciones y capacidad de forma independiente, sin afectar al resto del sistema.

- La arquitectura modular de los sistemas Cliente/Servidor permite el uso de ordenadores especializados (servidores de base de datos, servidores de ficheros, estaciones de trabajo para CAD, etc.).
- Permite centralizar el control de sistemas que estaban descentralizados, como por ejemplo la gestión de los ordenadores personales que antes estuvieran aislados.

5.5 Patrones de Diseño

Los sistemas que se desarrollan hoy en día son cada vez más comunes a sistemas ya realizados previamente. Lo que se busca hoy en día es estandarizar los procesos de desarrollo, para esto se crearon los patrones de diseño, los cuales ayudan a llevar en forma ordenada los procesos que se implementan.

Existen 3 tipos primarios de patrones, estos son los patrones creacionales, los patrones estructurales y los patrones de comportamiento.

5.5.1 ¿Qué son los patrones de diseño?

Un patrón de diseño es:

- Una solución estándar para un problema común de programación
- Una técnica para flexibilizar el código haciéndolo satisfacer ciertos criterios
- Un proyecto o estructura de implementación que logra una finalidad determinada
- Un lenguaje de programación de alto nivel
- Una manera más práctica de describir ciertos aspectos de la organización de un programa

- Conexiones entre componentes de programas

5.5.2 Objetivo de utilizar patrones:

Los patrones de diseño pretenden:

- Proporcionar catálogos de elementos reusables en el diseño de sistemas software.
- Evitar la reiteración en la búsqueda de soluciones a problemas ya conocidos y solucionados anteriormente.
- Formalizar un vocabulario común entre diseñadores.
- Estandarizar el modo en que se realiza el diseño.
- Facilitar el aprendizaje de las nuevas generaciones de diseñadores condensando conocimiento ya existente.

Asimismo, no pretenden:

- Imponer ciertas alternativas de diseño frente a otras.
- Eliminar la creatividad inherente al proceso de diseño.

No es obligatorio utilizar los patrones, solo es aconsejable en el caso de tener el mismo problema o similar que soluciona el patrón, siempre teniendo en cuenta que en un caso particular puede no ser aplicable. Abusar o forzar el uso de los patrones puede ser un error.

5.5.3 Categorías de patrones

Los patrones de diseño pueden ser categorizados según la escala o nivel de abstracción:

- **Patrones de arquitectura**: Aquéllos que expresan un esquema organizativo estructural fundamental para sistemas software.
- **Patrones de diseño**: Aquéllos que expresan esquemas para definir estructuras de diseño (o sus relaciones) con las que construir sistemas software.
- **Idiomas**: Patrones de bajo nivel específicos para un lenguaje de programación o entorno concreto.

5.5.4 Ventajas de los patrones de diseño

Son soluciones concretas:

- Un catálogo de patrones es un conjunto de recetas de diseño.
- Aunque se pueden clasificar, cada patrón es independiente del resto.

Son soluciones Técnicas:

- Dada una determinada situación, los patrones indican cómo resolverla mediante un D.O.O.
- Existen patrones específicos para un lenguaje determinado, y otros de carácter más general.

Se aplican en situaciones muy comunes:

- Proceden de la experiencia.

- Han demostrado su utilidad para resolver problemas que aparecen frecuentemente en el D.O.O.

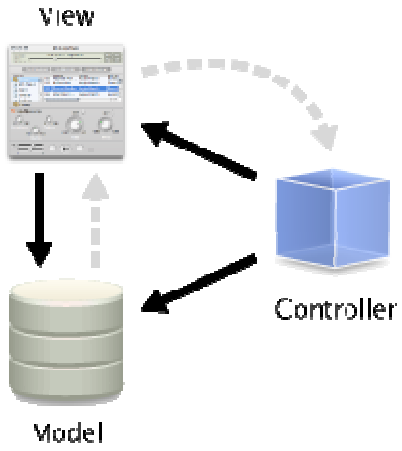
Son soluciones simples:

- Indican cómo resolver un problema particular utilizando un pequeño número de clases relacionadas de forma determinada.
- No indican cómo diseñar un sistema completo, sino sólo aspectos puntuales del mismo.

Facilitan la reutilización de las clases y del propio diseño:

- Los patrones favorecen la reutilización de clases ya existentes y la programación de clases reutilizables.
- La propia estructura del patrón es reutilizada cada vez que se aplica.

5.5.5 Patrón Modelo-Vista-Controlador



Esquema Modelo Vista - Controlador

- **Modelo**: Esta es la representación específica de la información con la cual el sistema opera. En resumen, el modelo se limita a lo relativo de la *vista* y

su *controlador* facilitando las presentaciones visuales complejas. El sistema también puede operar con más datos no relativos a la presentación, haciendo uso integrado de otras lógicas de negocio y de datos afines con el sistema modelado.

- **Vista**: Este presenta el modelo en un formato adecuado para interactuar, usualmente la interfaz de usuario.
- **Controlador**: Este responde a eventos, usualmente acciones del usuario, e invoca cambios en el modelo y, probablemente, en la vista.

Se trata de realizar un diseño que desacople la vista del modelo, con la finalidad de mejorar la reusabilidad. De esta forma las modificaciones en las vistas impactan en menor medida en la lógica de negocio o de datos.

Un modelo puede tener diversas vistas, cada una con su correspondiente controlador. Un ejemplo clásico es el de la información de una base de datos, que se puede presentar de diversas formas: diagrama de tarta, de barras, tabular.

Cada componente se responsabiliza en lo siguiente:

- El modelo es el responsable de:
    - Acceder a la capa de almacenamiento de datos. Lo ideal es que el modelo sea independiente del sistema de almacenamiento.
    - Define las reglas de negocio (la funcionalidad del sistema). Un ejemplo de regla puede ser: "Si la mercancía pedida no está en el almacén, consultar el tiempo de entrega estándar del proveedor".
    - Lleva un registro de las vistas y controladores del sistema.
    - Si estamos ante un modelo activo, notificará a las vistas los cambios que en los datos pueda producir un agente externo (por ejemplo, un fichero bath que actualiza los datos, un temporizador que desencadena una inserción, etc.).

- El controlador es responsable de:
  - Recibe los eventos de entrada (un clic, un cambio en un campo de texto, etc.).
  - Contiene reglas de gestión de eventos, del tipo "SI Evento Z, entonces Acción W". Estas acciones pueden suponer peticiones al modelo o a las vistas. Una de estas peticiones a las vistas puede ser una llamada al método "Actualizar()". Una petición al modelo puede ser "Obtener_tiempo_de_entrega(nueva_orden_de_venta)".

5.6 Metodología a utilizar

En el desarrollo de nuestro sistema utilizaremos la metodología de Prototipado Rápido, ya que posee una gran interacción con el cliente, se realizan múltiples prototipos, a medida que se van desarrollando las distintas funcionalidades.

5.6.1 Definición de Prototipado Rápido

La Metodología de Prototipado Rápido (MPR) está orientada al desarrollo de prototipos y fuertemente apoyada en tecnología de Bases de Datos y herramientas visuales para Desarrollo Orientado a Objetos. En MPR se concentra un gran esfuerzo en la involucración del Usuario en dos fases fundamentales: la definición del problema que se va a abordar y en la ejecución
de las pruebas, donde además se potencia el uso de lenguajes de cuarta generación utilizados como lenguajes de consulta para verificar la estructura y funcionalidad del prototipo desarrollado, asegurándose de que su diseño responde a las definiciones especificadas.
Como se ha expuesto, la idea fundamental de MPR es el desarrollo de prototipos. Un prototipo es un modelo inicial de lo que al final se corresponderá con la Base

de Datos definitiva y sus procedimientos asociados. Este prototipo se someterá a pruebas para comprobar su funcionalidad, de las que surgirán modificaciones que darán origen a un segundo prototipo, versión mejorada y posiblemente ampliada del primero, el cual se volverá a probar, repitiéndose sucesivamente el proceso hasta alcanzar el prototipo definitivo.

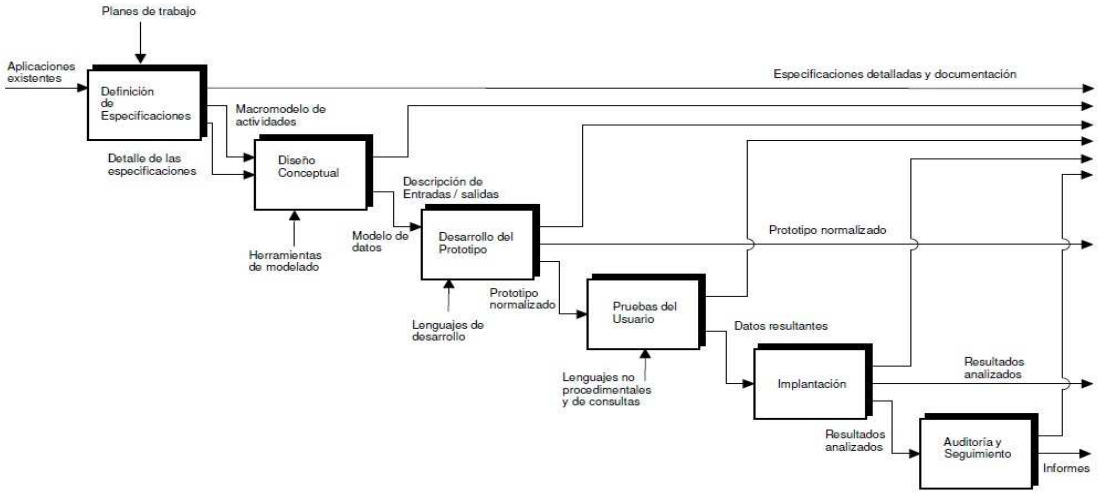
La responsabilidad y ejecución de estas pruebas fundamentalmente recae, como ya se ha mencionado, en el propio usuario, quien deberá de comprobar que el prototipo resultante es capaz de resolver todos los problemas planteados en el momento de la definición de las especificaciones del proyecto.

5.6.2 Representación gráfica de la metodología

Para representar la metodología de Prototipado Rápido en su mayor grado de abstracción, se hace uso de un diagrama, en el que se muestran las seis fases de las que se compone.

La simbología empleada en el diagrama es la siguiente: cada una de las cajas representa “algo que hay que hacer”, en este caso una fase de la Metodología. A una caja pueden llegar flechas por su lado izquierdo (entradas necesarias para ejecutar la fase), por su parte superior (causas por las que se realiza una fase) y por su parte inferior (herramientas y técnicas con las que se hace lo indicado en la fase). Asimismo, por su parte derecha salen flechas que muestran algo que se ha obtenido en la fase y que pasa normalmente a la fase siguiente o sale directamente fuera del sistema, indicando un producto ya terminado y que no necesita más procesos.

Desde cada fase de la Metodología se puede volver a cualquiera de las fases anteriores. En el diagrama SADT se han omitido voluntariamente estas conexiones para facilitar su estudio.



5.6.3 Estudio Detallado de Fases de Metodología

Fase 1: Definición de especificaciones

Esta primera fase tiene por objeto auditar la información de relativa al problema, con el fin de recabar todos los datos necesarios para su resolución.

Como primera tarea podrá ser necesario realizar un estudio de la viabilidad del proyecto, para determinar y justificar la necesidad del mismo. A continuación se hará un análisis previo con el fin de establecer la amplitud y el calendario del proyecto, estimándose el esfuerzo necesario y el tiempo de desarrollo, e identificando los procesos involucrados en el mismo.

A continuación se construirá un prototipo inicial, no necesariamente operativo,

Construyéndose macromodelos de actividad para cada uno de los procesos identificados en la actividad anterior. La intención es disponer de la información necesaria para recabar la aprobación necesaria para comenzar el desarrollo.

Se trata, en suma, de obtener la mayor cantidad de información posible sobre el problema que se intenta resolver, para obtener un tiempo estimado de desarrollo del proyecto y sus costes asociados, con el fin de obtener la aprobación necesaria para llevarlo a cabo.

Fase 2: Diseño Conceptual

El objetivo de esta fase es construir un modelo de información que refleje el esquema conceptual del prototipo. Es muy importante que este modelo esté lo más ajustado posible a la realidad para que el diseño posterior de la Base de Datos pueda cumplir con sus objetivos. Se realizarán entrevistas a los Usuarios y se estudiará y diseñará el primer prototipo operativo, determinando sus puntos fuertes y sus puntos débiles, y se documentarán todas sus funcionalidades. También en esta fase se prepararán los planes de implantación, formación y pruebas, y se desarrollará el Manual del Usuario y el Manual Técnico.

Fase 3: Desarrollo del Prototipo

Como su nombre indica, esta fase tiene por objeto la construcción del primer prototipo operativo de la Aplicación. Esta fase consta de dos actividades principales, una de desarrollo técnico propiamente dicho y otra para desarrollar la documentación asociada. Al finalizar esta fase se dispondrá de un prototipo totalmente funcional y operativo, que será sometido a múltiples pruebas en la siguiente fase para comprobar su validez.

Fase 4: Pruebas del Usuario

En esta fase se realizarán todas las pruebas necesarias para validar el prototipo desarrollado en la fase anterior. Si como resultado de estas pruebas se detectara la necesidad de modificar el prototipo, para corregir defectos o para añadirle funcionalidad, se volverá a la fase anterior y se realizarán todas las iteraciones necesarias hasta que el Usuario se sienta satisfecho con el prototipo, y compruebe que responde a las especificaciones que se habían alcanzado inicialmente. Las pruebas en esta fase pueden ser de dos tipos: pruebas dirigidas, donde los desarrolladores guían y asesoran al usuario durante las mismas, y pruebas no

dirigidas, donde el Usuario actúa libremente y sin la presencia de los desarrolladores.

Fase 5: Implantación

En esta fase se ejecutará el Plan de Formación de los Usuarios y se llevará a cabo el proceso de migración al entorno de ejecución real de la aplicación. Una vez completada la migración, se realizarán las pruebas finales y se llevarán a cabo las actividades correctoras finales, revisándose de paso toda la documentación del proyecto. Como final de esta fase se deberá de obtener la aceptación del Usuario y se emitirá un informe para la Dirección del Proyecto.

Fase 6: Auditoría y Seguimiento

La última de las fases de MPR consiste en realizar una Auditoría del rendimiento y la calidad de la Aplicación, y de determinar y canalizar los mecanismos necesarios para realizar peticiones de modificación y para que estas sean llevadas a cabo por los Equipos de Mantenimiento. Se deberán de identificar parámetros de rendimiento, compromisos de uso/respuesta, verificar la calidad global de la aplicación y efectuar las medidas correctoras oportunas.

Como fin de fase, se comprobará que toda la documentación es la adecuada, y se obtendrá la aprobación definitiva del Usuario.

Capítulo 6: Estudio de Factibilidad

6.1 Estudio de Factibilidad Económico

Todo tipo de empresa, siempre busca maximizar su nivel de utilidad, pero muchas veces para garantizar una utilidad deseada, se deben invertir bienes para mejorar los niveles de producción, de calidad del producto, de satisfacción de usuarios, de satisfacción de personal. Una forma de realizar lo anterior, es mejorar los procesos de búsqueda de información dentro de la empresa. Muchas empresas pierden mucho tiempo en buscar algún tipo de información, dejando así la posibilidad de dejar esperando clientes a medida que estos van llegando. Nuestro país presenta un clima templado y lugares maravillosos para ir a visitar, por lo que cada año más gente viene a visitar el país. Además, cada día nuestro país avanza rumbo a ser un país desarrollado, y es por esto que día a día se realizan una serie de eventos a nivel nacional, siendo éstos de carácter deportivo, como el Rally Dakar que se hace presente cada año y múltiples torneos deportivos; musical, con cientos de conciertos de artistas y grupos destacados que se realizan al año a nivel nacional; de salud con un sin fin de exposiciones, conferencias, congresos; tecnológicos, culturales, de negocio, etc. Cada vez que se realiza uno de estos eventos, dependiendo de su magnitud, el sector turismo cercano a los lugares que albergan estos eventos muchas veces se ven sobrepasados en capacidad, y es necesario así una atención fluida, inmediata y sin interrupciones, sin perder tiempo en búsqueda de información y sin cometer errores. Nuestro sistema se basa básicamente en mejorar este punto y acelerar así este proceso. Es por esto que es necesario determinar si el costo de implementar el sistema anteriormente mencionado será beneficioso para el futuro de la empresa.

6.1.1 Determinación de Precio del Producto

En cuanto al ámbito del equipo desarrollador, el gasto que se realiza en el desarrollo del sistema se determinará a continuación:

Costo mensual equipo desarrollador:

1 Analista	$ 550.000
1 Diseñador	$ 550.000
2 Programadores	$ 1.000.000
1 Tester	$ 400.000
1 Coordinador y Jefe de Proyecto	$ 700.000
Costo mensual de construcción del sistema	$ 3.200.000
Total Costo de construcción del sistema	$ 9.600.000

Considerando un mercado potencial de 1.000 empresas, para efecto de la distribución del costeo del desarrollo del software, utilizaremos un 10% del mercado potencial es decir 1.000 entre las cuales distribuiremos el costo de desarrollar el software. Todo lo que se venda pasando la cantidad de 100 se considera como ganancia.

Usando la siguiente formula tenemos:

CT = CV * Q + CF donde: CT = costo total.

CV= costo variable.

Q = cantidad demandada.

CF = costo fijo.

Por lo tanto haciendo el costo variable igual a cero determinamos el costo fijo de desarrollo:

CF = 9.600.000 / 100 = 96.000

Por concepto de costo variable tenemos lo siguiente:

Manual + CD = $ 1.200 por la unidad.

Instalación = $ 10.000 por unidad.

Transporte = $ 1.000 por unidad (promedio)

CV = $ 12.200

Añadiendo los costos variables a nuestra ecuación de costos nos resulta:

CT = 12.200 * Q + 96.000

Para determinar la curva de utilidad de nuestro producto, y de esta manera poder obtener el precio de nuestro producto estudiamos la curva de demanda de

productos alternativos existentes en el mercado. El siguiente estudio muestra los resultados de 3 productos alternativos.

	Q	P	P * Q	P²
	20	230.000	4.600.000	52.900.000.000
	43	190.000	8.170.000	36.100.000.000
	37	120.000	4.440.000	14.400.000.000
Total	**100**	**540.000**	**17.210.000**	**103.400.000.000**

Donde: Q = Cantidad demanda del Producto

P = Precio del Producto

$\bar{Q} = 100 / 3 = 33{,}33$

$\bar{P} = 540.000 / 3 = 180.000$

$b = \bar{Q} - (a * \bar{P})$ Donde:

$$a = \frac{n * \sum (P * Q) - \sum P * \sum Q}{n * \sum P^2 - (\sum P)^2} = -1{,}27 * 10^{-4}$$

$b = 33{,}33 - (-1{,}27 * 10^{-4} * 180.000) = 56{,}26$

Por lo tanto tenemos:

$CT = 12.200 * Q + 96.000$ y $Q = 56{,}26 - 0{,}000127 * P$

Luego usamos la siguiente ecuación de utilidad para determinar el precio:

$U = P * Q - CV * Q - CF$

$U = P * (56{,}26 - 0{,}000127 * P) - 12200 * (56{,}26 - 0{,}000127 * P) - 96.000$

Lo que queda como:

$$U = -- 0{,}000127 * P^2 + 57{,}82 * P - 745.879{,}56$$

Ahora calculando la derivada a la función de utilidad con respecto a P:

$$\frac{\partial U}{\partial P} = -- 0{,}000254 * P + 57{,}82$$

Igualamos la derivada a cero para calcular el máximo punto de la curva de utilidad:

$$-- 0{,}000254 * P + 57{,}82 = 0$$

$$P = 226.902{,}30$$

Redondeando, el precio final del producto estimado a ser vendido a un mínimo de 100 unidades, es de $ 227.000. Toda venta superior a las 100 unidades se considera como utilidad.

6.1.2 Aspecto Social

Ejemplifiquemos la situación de una mediana empresa que se dedica al rubro de hotelería, ubicado en viña del mar, el cual presenta un número de personal de alrededor de 7 personas (en temporada alta), el cual cuenta con 7 habitaciones dobles y 15 habitaciones simples, siendo un hotel de mediana calidad. Los gastos que se producen mensualmente en temporada alta, a máxima capacidad, son los siguientes:

Sueldos	$ 3.000.000
Luz, Agua, Gas	$ 1.100.000

Artículos Limpieza	$ 200.000
Insumos de Alimentación	$ 1.200.000
Improvistos y otros gastos	$ 500.000
	$ 6.000.000

Ahora debemos de tomar en cuenta los ingresos que se perciben en temporada alta, para esto debemos considerar que el mayor ingreso es el prestación de las habitaciones. Los valores de las habitaciones en temporada alta, por día, son las siguientes:

Habitación simple	$ 15.000
Habitación Doble	$ 25.000

Los ingresos que se perciben mensualmente en temporada alta fluctúan entre los 7 y 10 millones de pesos por concepto de prestación del servicio de habitación, los otros ingresos no llegan a superar el millón de pesos. Por lo que la utilidad estimada en temporada alta es:

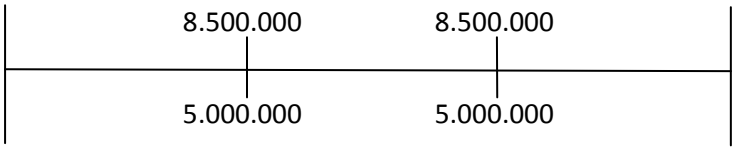
Utilidad máxima esperada = 11.000.000 – 6.000.000 = 5.000.000

Utilidad mínima esperada = 5.500.000 – 4.000.000 = 1.500.000

Los costos promedio serían de $5.000.000

Los ingresos promedio serían de $ 8.500.000

A continuación se representan los valores promedios de costos – ingresos durante los 2 meses de temporada alta



Ahora, si bien estas son las utilidades esperadas, debemos decir que estas podrían ser mayores, puesto que, de las 168 horas semanales, el 10% de estas se dedican a la búsqueda de información. Tiempo el cual puede ser otorgado a otras tareas. Con el sistema ya implantado la búsqueda de información no tomará más del 2% de las horas semanales. Esta reducción se debe a que el sistema es automático, y muy veloz en cuanto a la búsqueda de información necesaria para este tipo de negocios.

6.2 Estudio de Factibilidad Técnica

Este sistema está orientado para ser manejado en red, es por esto que se debe de tener en consideración la utilización de un servidor. Este servidor debe poseer ciertas características, las cuales permitirán al servidor trabajar en forma correcta. Así también, se debe considerar que se requiere de un soporte para mantener en buen estado el servidor a lo largo del tiempo.

Se debe disponer de software básico para el correcto funcionamiento del sistema, es así que tanto sistema operativo como la base de datos del sistema deben poseer un nivel de soporte adecuado. El mercado dispone de corporaciones que realizan soporte, las cuales se dedican al tema. Pero también el soporte puede ser manejado por estudiantes o personas particulares que saben sobre este tema.

6.3 Estudio de Factibilidad Operacional

La introducción de un nuevo sistema dentro de una empresa es un tema que siempre provocan cambios dentro de esta, ya sea a nivel gerencial u operacional. Muchas empresas a lo largo de la historia, al momento de introducir nuevos sistemas han provocado numerosas reducciones de personal. Esto está relacionado a que muchos procesos que se realizaban anteriormente, ya no requieren del mismo tiempo o número de personas trabajando en estos procesos. Además, a veces, la gente que hace ciertas labores de una forma determinada no le gusta cambiar la forma en que hace su trabajo, por lo cual algunas veces estos cambios son desechados.

Estos cambios siempre llevan asociados un cierto nivel de aceptación por lo trabajadores dentro de la empresa. Un buen sistema que acorte los tiempos de los procesos sustancialmente, facilite las labores en gran porcentaje y permita así obtener mayor utilidad, siempre obtendrán un gran nivel de aceptación dentro de la empresa.

Según los requerimientos del usuario vistos anteriormente, los cambios que se producirán dentro de una empresa promedio del rubro de hotelería serán los siguientes:

- Se obtendrá un mayor nivel de orden de la información.
- La información será tratada de manera óptima.
- Se adquirirán niveles insuperables en tiempo de búsqueda de información.

El sistema puede ser ocupado tanto en una Internet como en una red local, es por esto que, quien disponga del sistema se verá obligado a decidir si ocupar este sistema dentro de una red local o Internet. Este es un tema en el cual principalmente se considera el tamaño de la empresa (nivel de sucursales de la empresa) y el nivel de seguridad que se presenta dentro de la empresa. Muchos usuarios prefieren realizar transacciones a nivel local, dentro de una red intranet, a permitir manejar datos dentro de Internet, los cuales podrían ser replicados o robados. Es por esto que se tiene un cierto nivel de aceptación o rechazo de trabajar con bases de datos vía Internet. Por el momento el sistema será implantado en su primera versión en una red local.

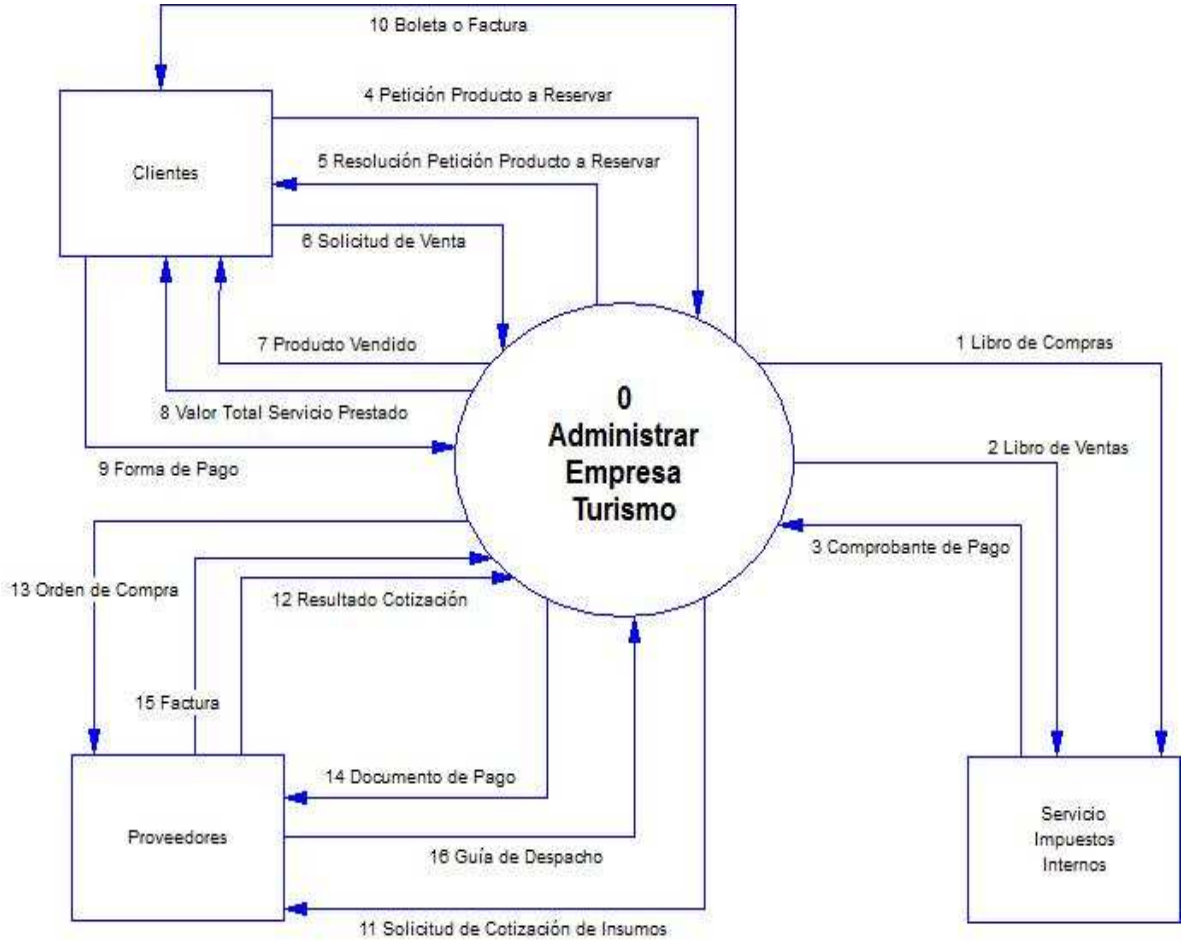
Cuando el sistema ya sea usado por los usuarios este debe presentar un diseño e interfaz amigable, con la cual el usuario sepa diferenciar fácilmente entre pantallas usando colores adecuados para la vista, así como también debe presentar facilidad para el usuario, mostrando la información correspondiente y clara dentro de la pantalla.

Al momento de lanzar el sistema, los menús pertenecientes a este deben ser claros, en un despliegue jerárquico comprensible, de tal manera de que si se desea realizar una acción, los menús deben ser desplegados de tal manera de colar solo lo que el usuario desea y, evitar así información redundante o que no sea necesaria, para así satisfacer lo que el usuario desea realizar.

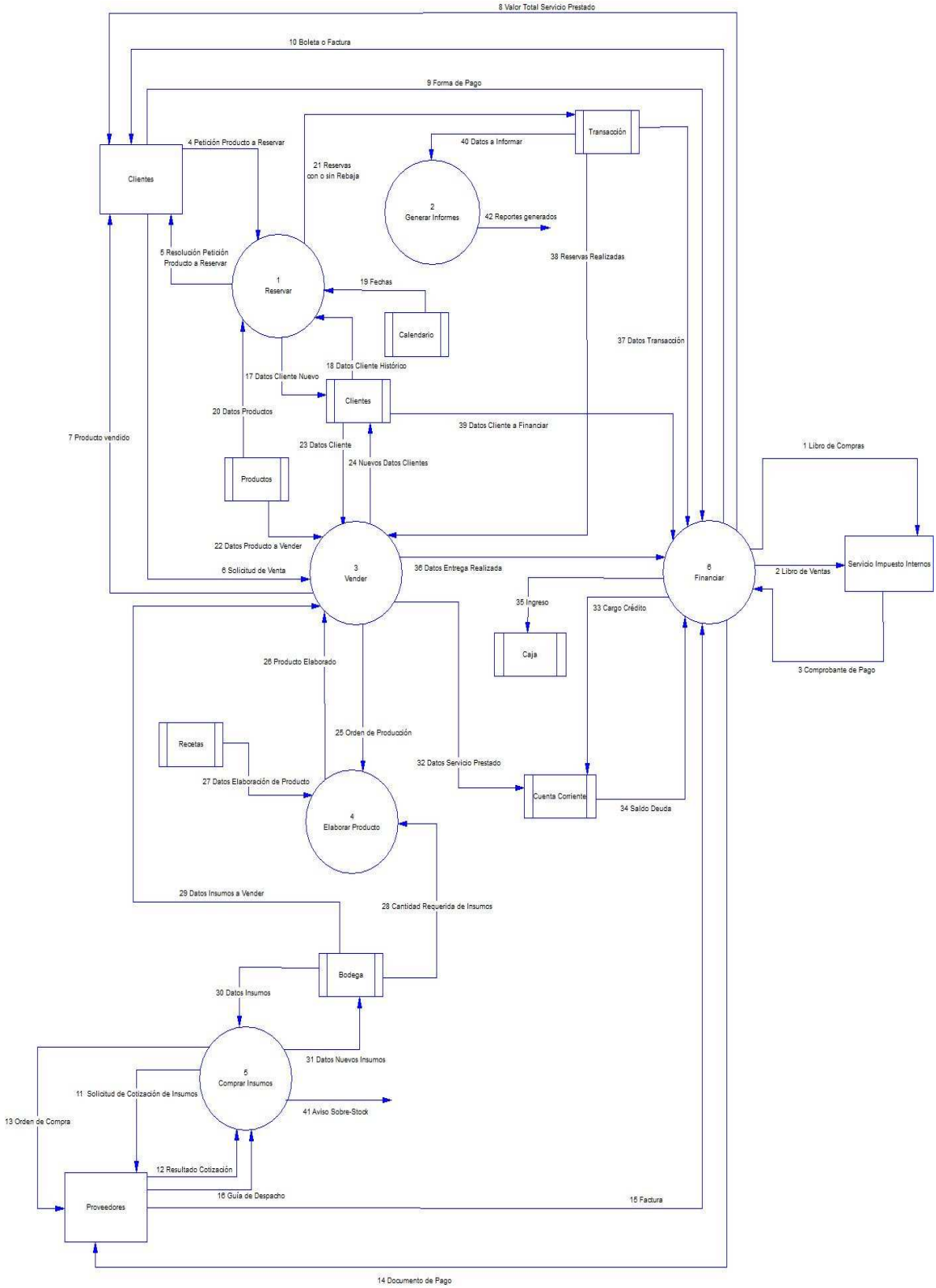
Capítulo 7:
Diseño Lógico

7.1 Diagramas de flujo de datos de procesos de negocio

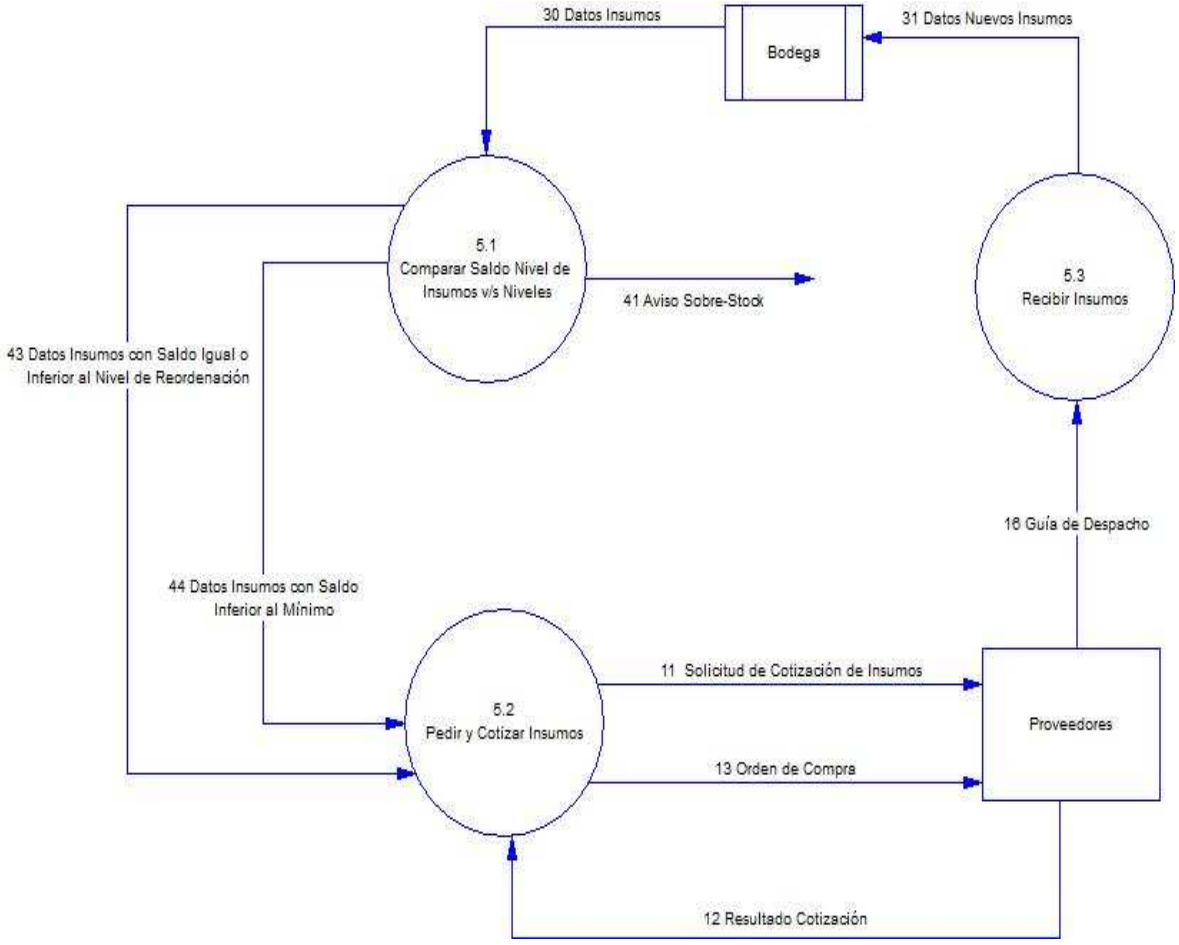
7.1.1 Diagrama de Contexto



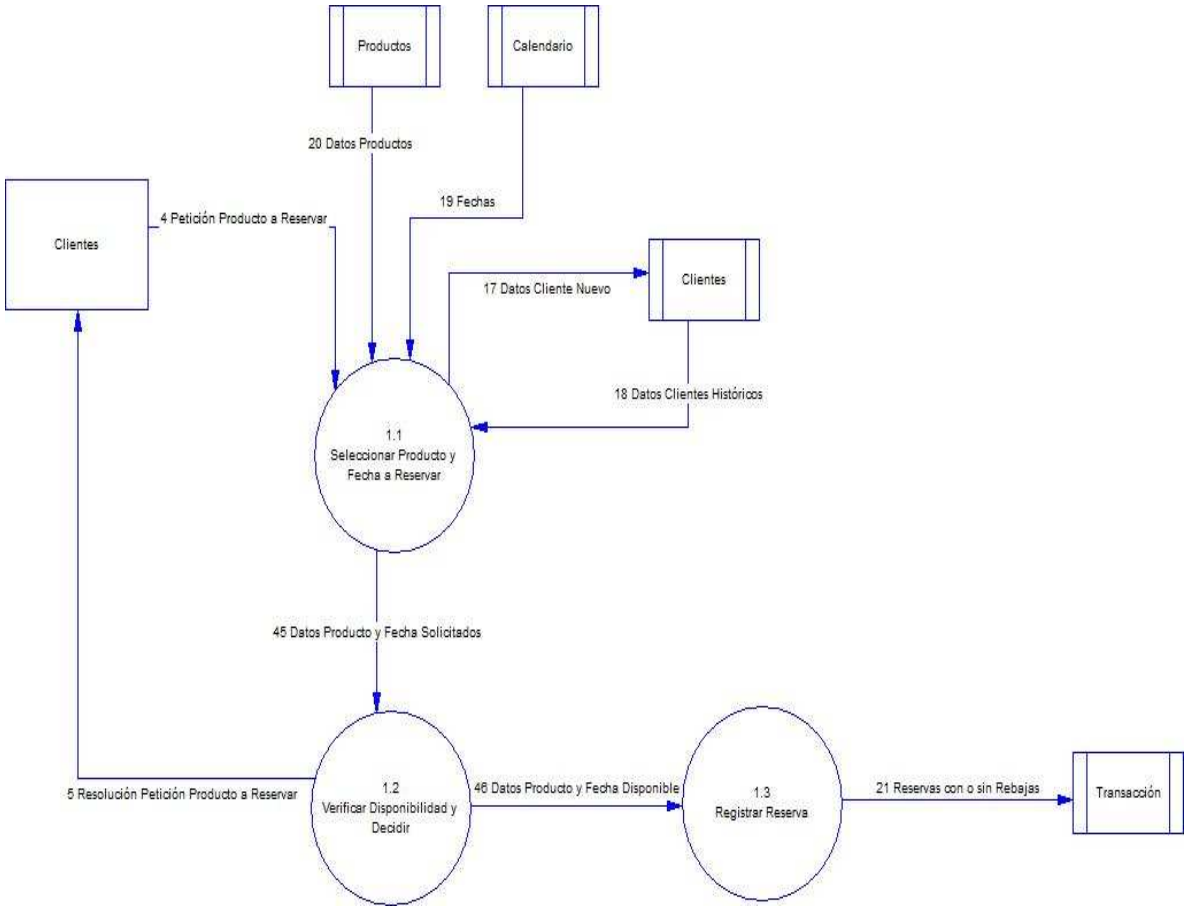
7.1.2 Diagrama Superior



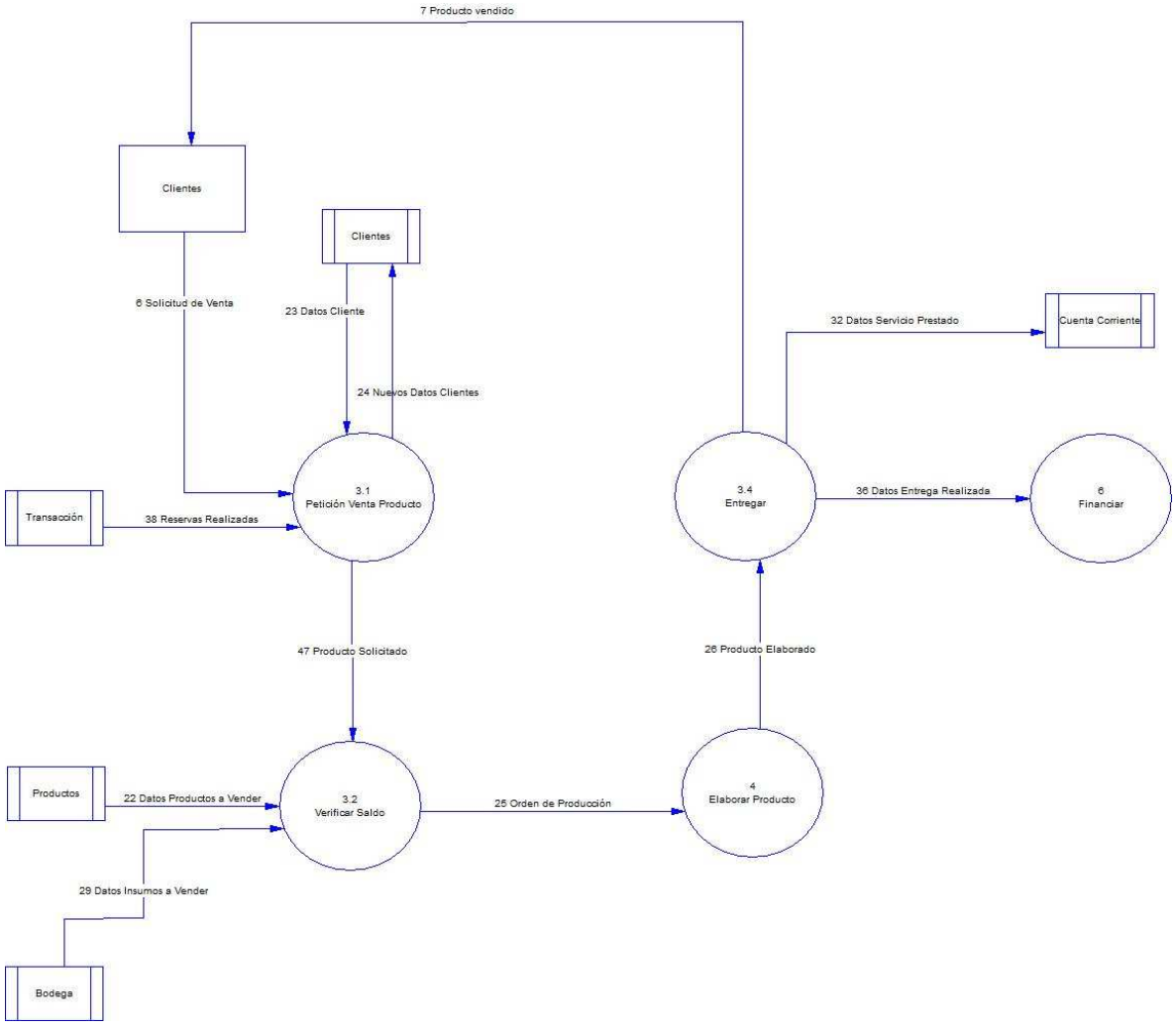
7.1.3 Diagrama de Detalle (Proceso Comprar Insumos)



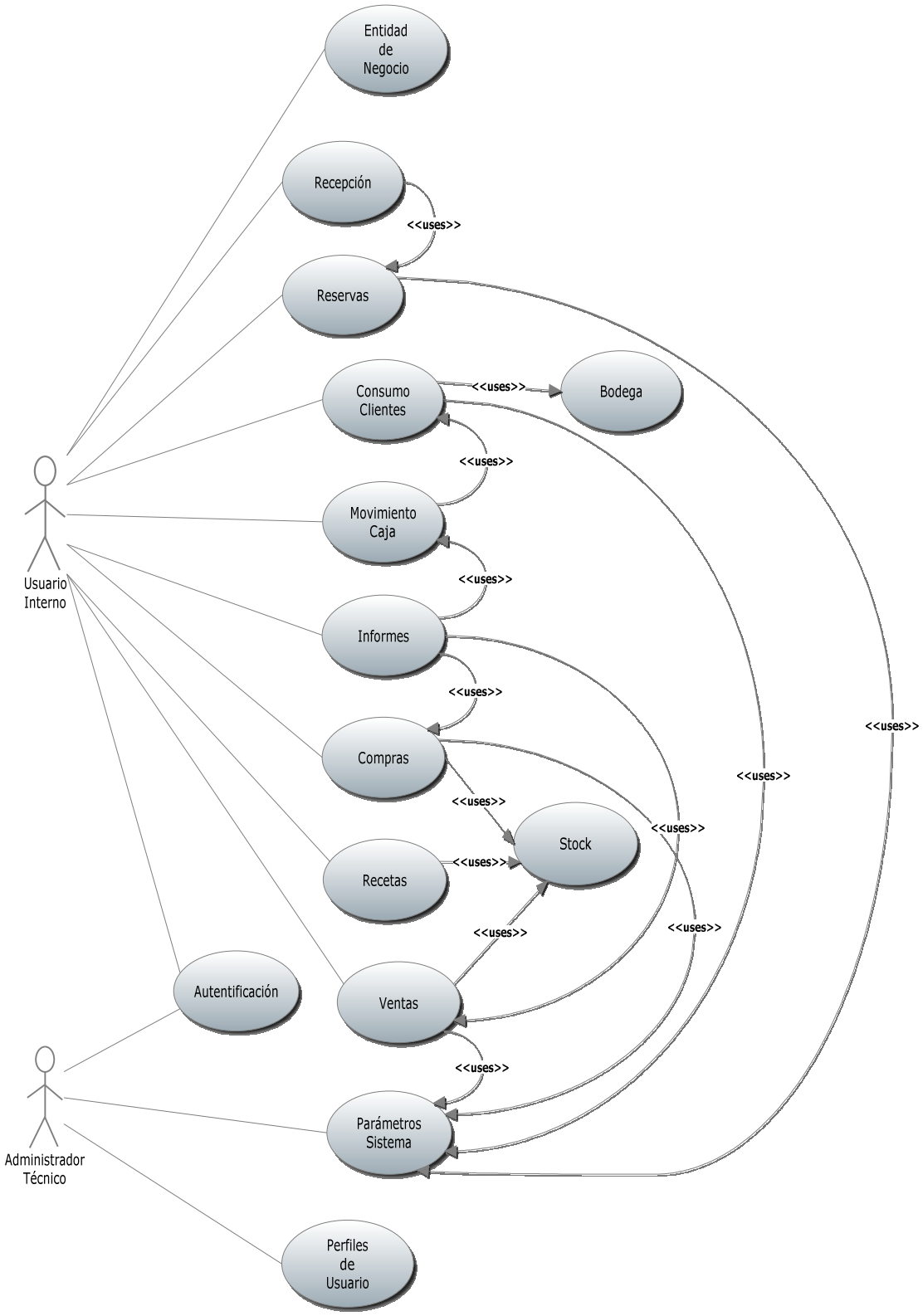
7.1.4 Diagrama de Detalle (Proceso Reservar)



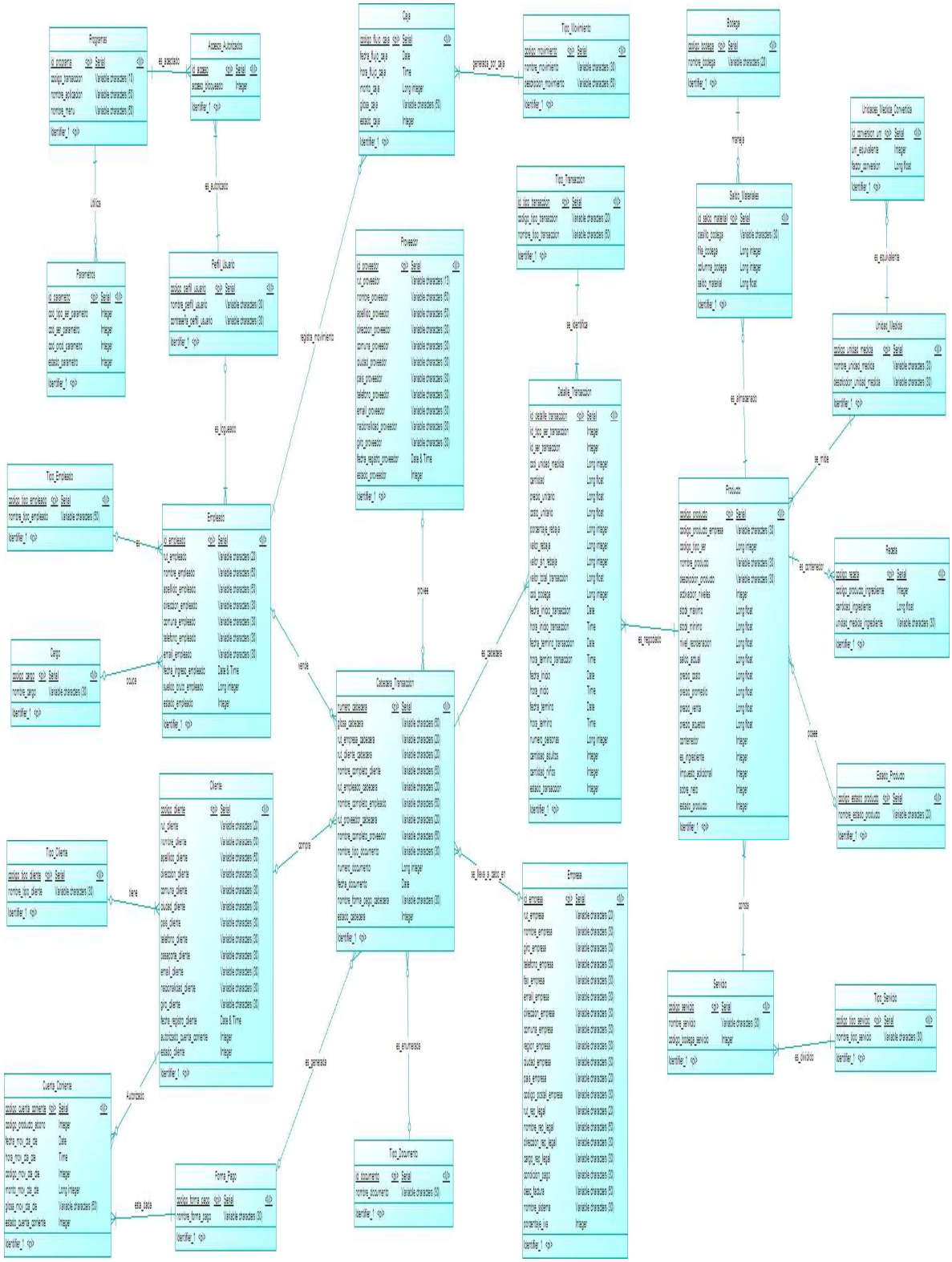
7.1.5 Diagrama de Detalle (Proceso Vender)



7.2 Diagrama de Casos de Uso



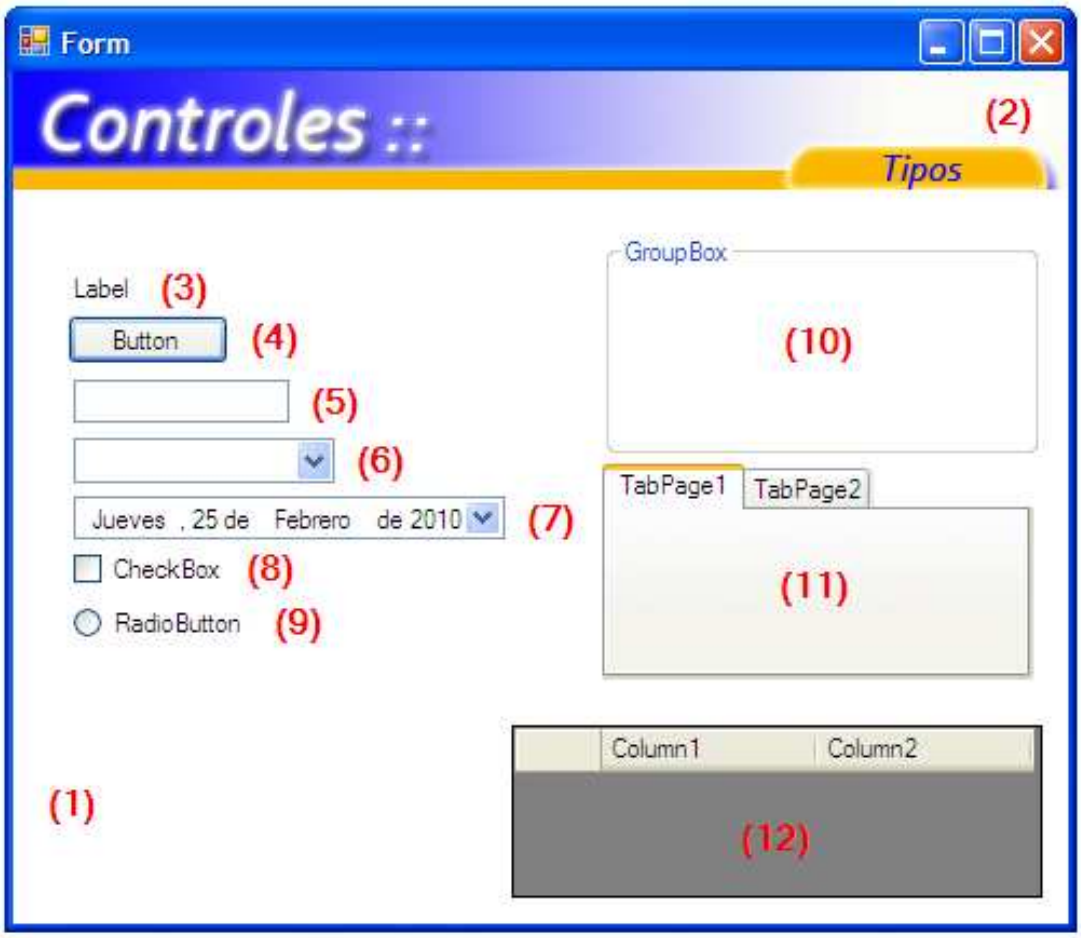
7.3 Modelo Conceptual (Entidad Relación)



7.4 Diseño de Entradas / Salidas

7.4.1 Interfaces Hombre – Máquina

Controles que provee Visual Studio



(1) Form.

(2) Picture Box.

(3) Label.

(4) Button.

(5) Text Box.

(6) Combo Box.

(7) Date Time Picker.

(8) Check Box.

(9) Radio Button.
(10) Group Box.
(11) Tab Control.
(12) Data Grid View.

Tipos de botones que utiliza el sistema.

a) Todos los formularios del sistema utilizan alguna combinación de los siguientes botones:

- Aceptar: para confirmar cualquier operación que se está generando.
- Insertar: para insertar algún ítem sobre un DataGridview (y confirmar la operación).
- Modificar: para modificar los datos desplegados.
- Eliminar: para eliminar un registro de la base de datos (confirmando previamente la acción).
- Limpiar: para limpiar todos los controles del formulario.
- Cancelar: para cancelar cualquier operación, o cerrar un formulario.

b) Todos los formularios poseen los botones estándar , que se muestran a continuación:

c) Algunos formularios contienen botones de operación sobre un DataGridView:

- Agregar un elemento al DataGridView.
- Editar un ítem del DataGridView (seleccionado previamente).
- Eliminar un ítem del DataGridView (seleccionado previamente).

Validaciones

Visual Studio provee una herramienta llamada “Error Provider”, que permite asignar un icono de error a un determinado control, y un mensaje de error (mostrado al posicionar el mouse sobre el icono). El icono a utilizar es el que viene dado por defecto:


.

Todos los formularios de nuestro sistema utilizan esta herramienta para efectuar algunas validaciones sobre los cuadros de texto. Las validaciones son:

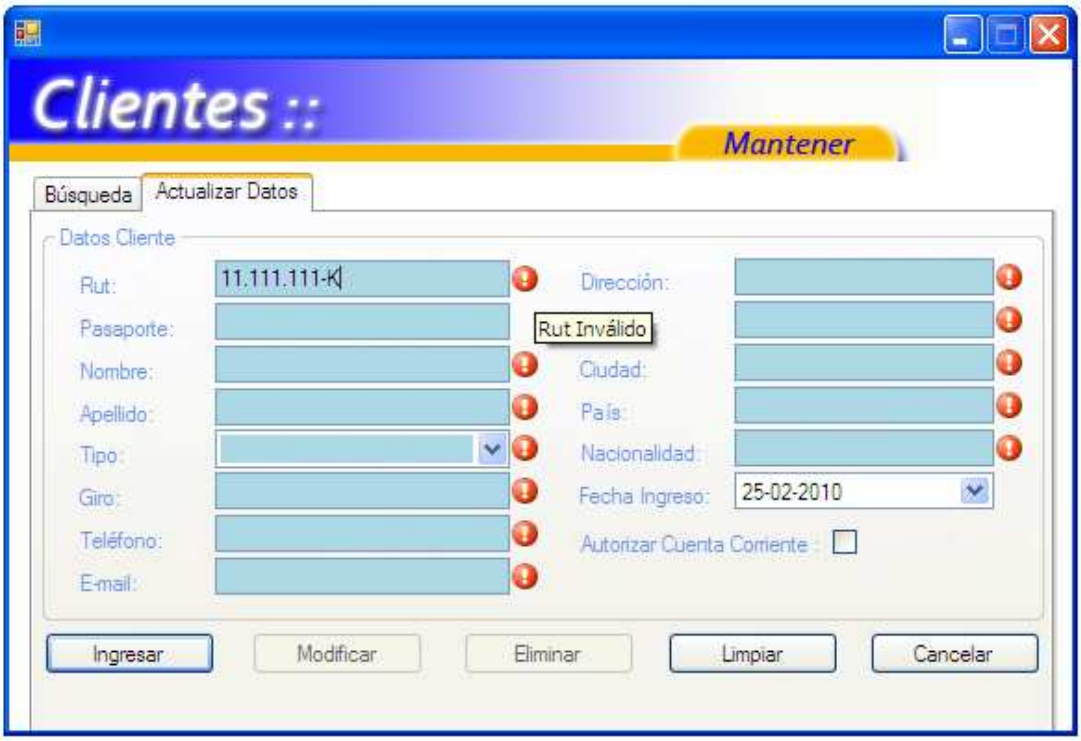
a) Campos sólo numéricos: en caso de error, se muestra el mensaje: “Sólo Campos Numéricos”.

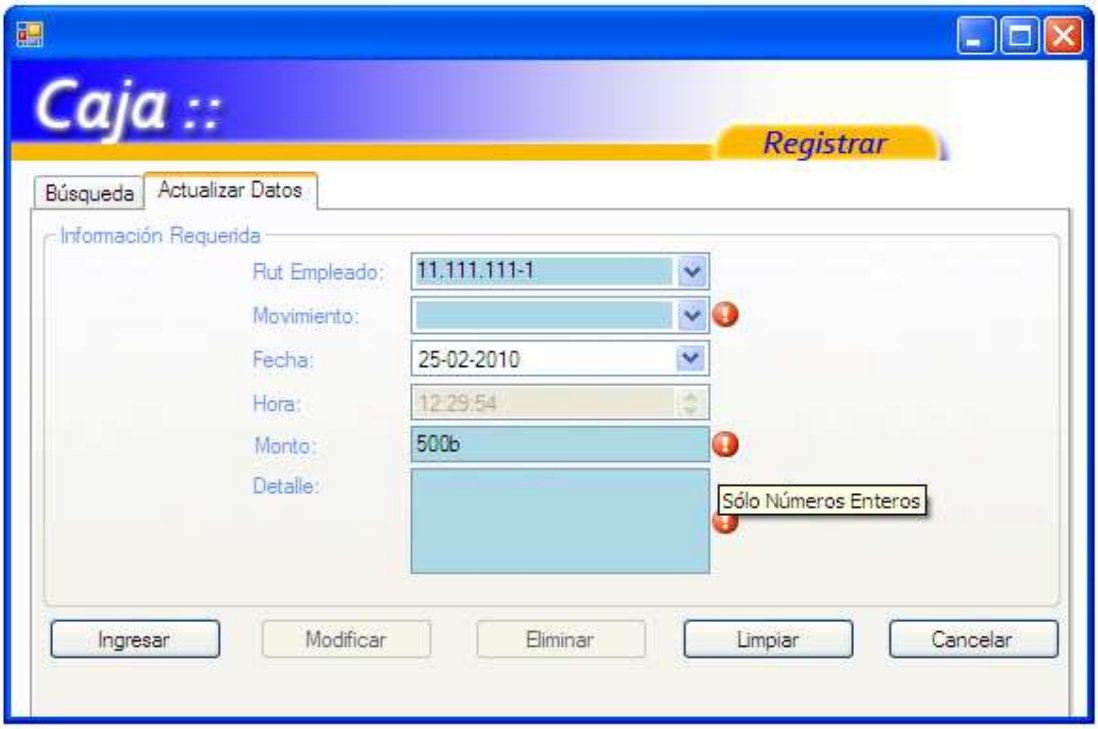
b) Campos no nulos: en caso de error, se muestra el mensaje: “Campo Requerido”.

c) Validar rut: en caso de error, se muestra el mensaje: “Rut Inválido”.

d) Validar email: en caso de error, se muestra el mensaje: “Email Inválido”.

Ejemplo: Validación de rut, email,campos no nulos y campos numéricos.





Mensajes

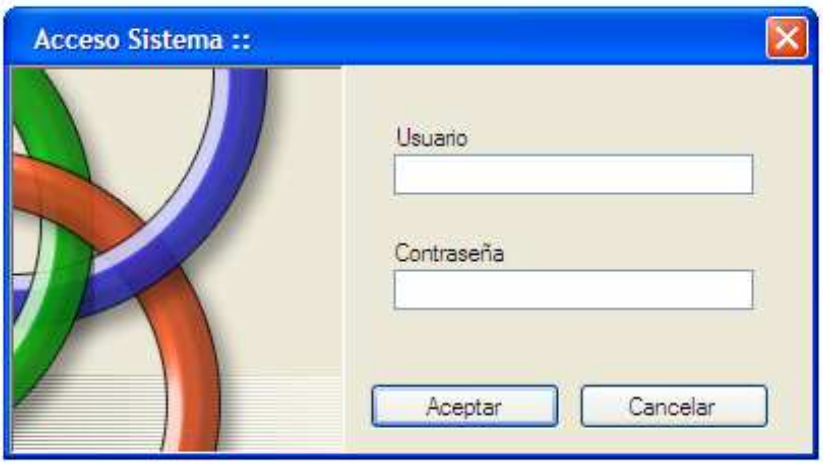
Los formularios del sistema generan mensajes con la siguiente estructura:

a) Información: mensajes acerca del estado de la operación que se está generando.

b) Pregunta: para confirmar o anular la operación que se está generando.

Ejemplos:

Autentificación del sistema



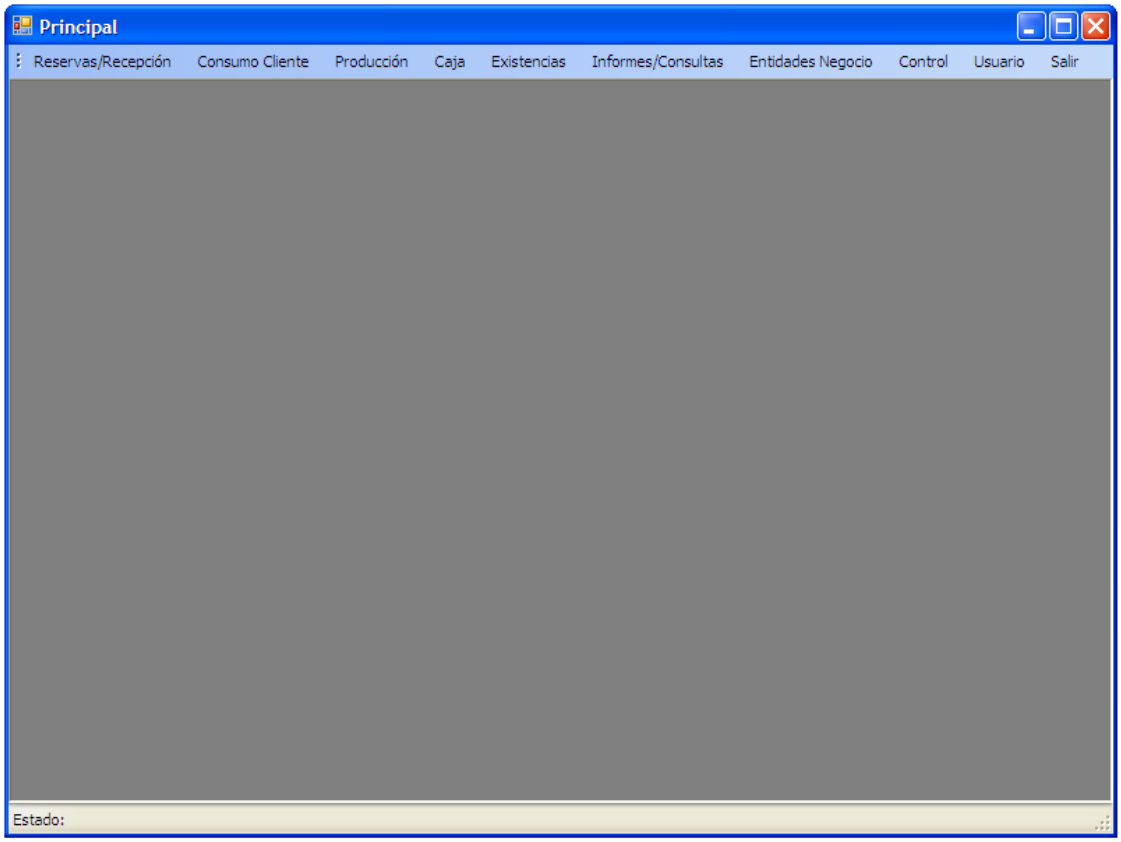
El administrador del sistema es el encargado de crear las cuentas de usuario (Perfiles) y definir los privilegios correspondientes, es decir, la visibilidad del usuario sobre las funciones del sistema.

En esta ventana, el usuario debe ingresar el nombre de su cuenta (Usuario), su contraseña, y presionar “Aceptar” para entrar al sistema. Si desea salir debe presionar sobre “Cancelar”.

Si los datos de usuario están correctos, se abre el formulario principal del sistema, acotado de acuerdo a los privilegios definidos por el administrador. Si los datos están incorrectos, se sitúa el “error provider” sobre el o los campos erróneos y se muestra un mensaje de información de acuerdo al tipo de error:

a) “Debe ingresar usuario y contraseña”.
b) “Usuario o contraseña incorrectos”.

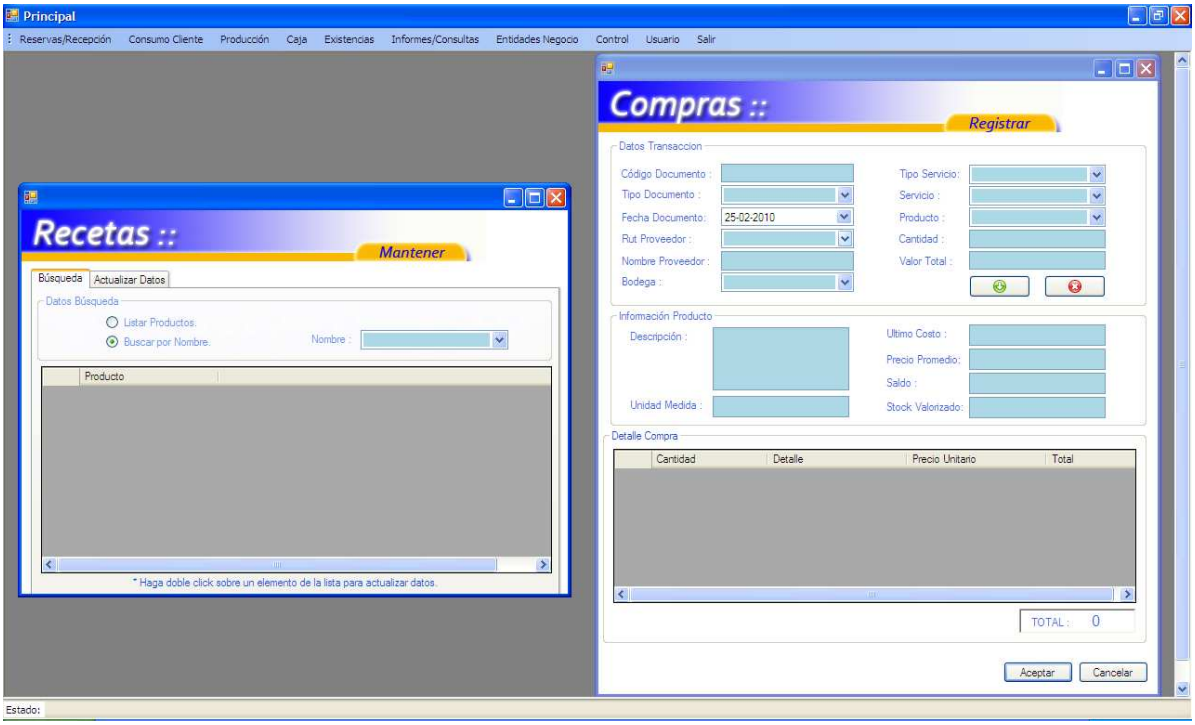
Formulario Principal.



De acuerdo a los privilegios del usuario que está ingresando al sistema (definidos en la tabla “Accesos Autorizados”), se cargan los ítems del menú y los formularios que podrán ser visibles. En la figura anterior se muestra la vista del administrador.

Para abrir un formulario, se debe hacer click sobre un ítem de la barra de menú, y acceder a la funcionalidad correspondiente. Para salir del sistema se debe hacer click sobre el ítem “Salir”, o sobre el botón cerrar de la esquina superior derecha. Con esto se lanzará un mensaje de pregunta para confirmar la acción.

El formulario principal tiene la propiedad “IsMdiContainer”, lo que permite abrir diversos formularios hijos (“Mdi Child”) dentro de los límites del formulario principal (“Mdi Parent”).



Formularios de Búsqueda y Actualización.

Estos formularios poseen un tabcontrol con 2 pestañas, una para la búsqueda de datos, y otra para la actualización de datos.

- Búsqueda:

En general, la pestaña de búsqueda contiene varios radiobutton, uno para listar todos los datos, y uno o más para buscar un dato con un valor específico (determinado por un combo box).

Al hacer click sobre el radiobutton “Listar”, o sobre cualquier radiobutton de búsqueda, se despliega el resultado de la consulta en un datagridview asignado

para ello. En este datagridview se puede navegar con el mouse, o con las flechas “Up” y “Down”.

- Actualización:

Esta pestaña tiene 2 formas de trabajar, una en modo ingreso de datos, y otra en de modificación de datos.

La primera se genera al entrar directamente a la pestaña, sin que previamente se haya hecho Doble Click, o presionado Enter sobre un elemento listado en el datagridview (pestaña de búsqueda). En este modo se habilita el botón “Ingresar” o “Aceptar”, y se deshabilitan los botones “Modificar” o “Eliminar”.

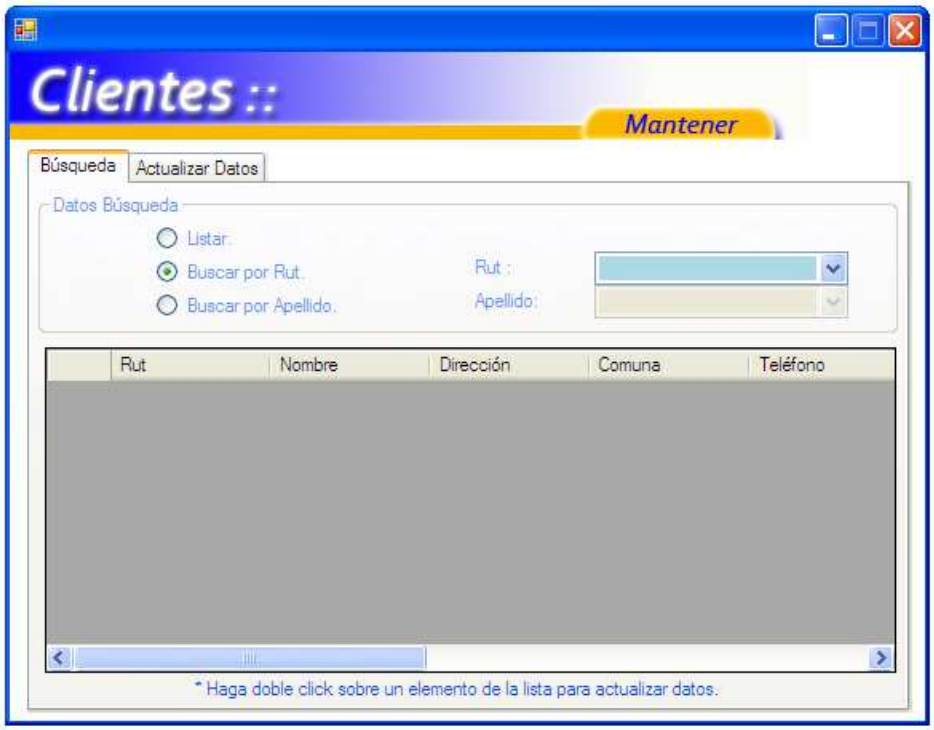
De forma contraria, en modificar datos, se deshabilita el botón “Ingresar” o “Aceptar”, y se habilitan los botones “Modificar” o “Eliminar”. En este modo, se carga a información asociada al dato buscado en la pestaña de búsqueda.

Los formularios del sistema que poseen estas características son:

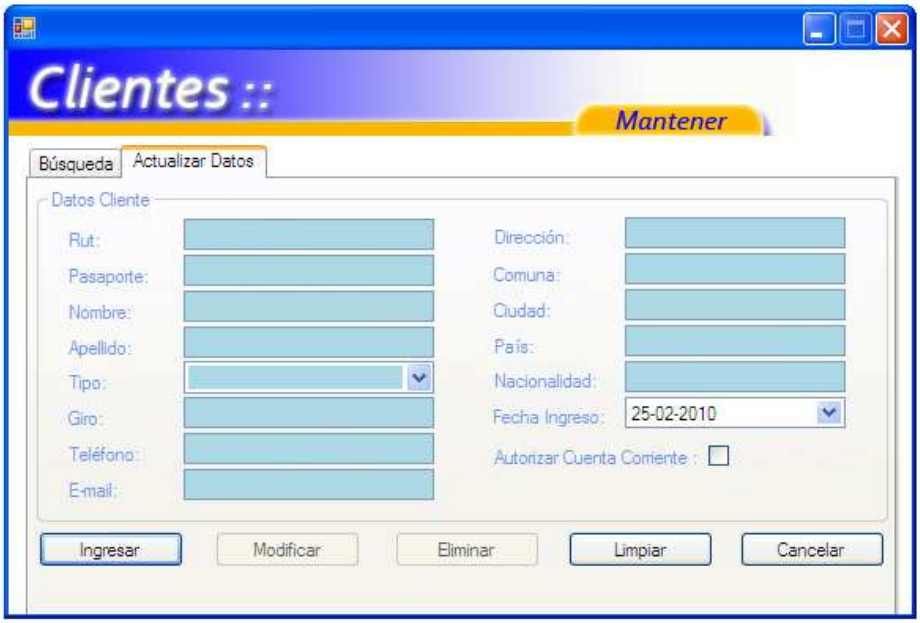
- Menú Entidades de Negocio: Cliente, Empleados, Proveedores, Productos, Servicios.
- Menú Caja: Mantener Caja, Tipos de Movimiento.
- Menú Control: Perfiles Usuario.
- Menú Usuario: Cambiar Contraseña.

Ejemplo:

Pestaña Búsqueda:



Pestaña Actualizar Datos:



Formularios de Estructura Simple.

Estos formularios contemplan de manera simple, y en una sola vista las funcionalidades de ingreso y la modificación de datos. Para confirmar la acción, está el botón “Aceptar”, y para cancelar o salir, “Cancelar” (o el botón cerrar de la esquina superior derecha). Entre los controles más relevantes que contienen, están:

- GroupBox.
- TextBox.
- Combo Box.
- Button.
- Date Time Picker.

Los formularios que poseen esta estructura son:

- Menú Existencias: Ajustes, Movimientos entre Códigos
- Menú Entidades de Negocio: Empresa.

Ejemplo:

Formulario con Estructura de Factura.

Estos formularios contemplan, en su mayoría, los datos que componen una factura:

- Datos cliente, proveedor o empleado.
- Detalle de documento.
- Detalle de los productos seleccionados.
- Cantidades y valores de los productos que se están transando.
- Lista de los productos que se están transando, y el monto total.

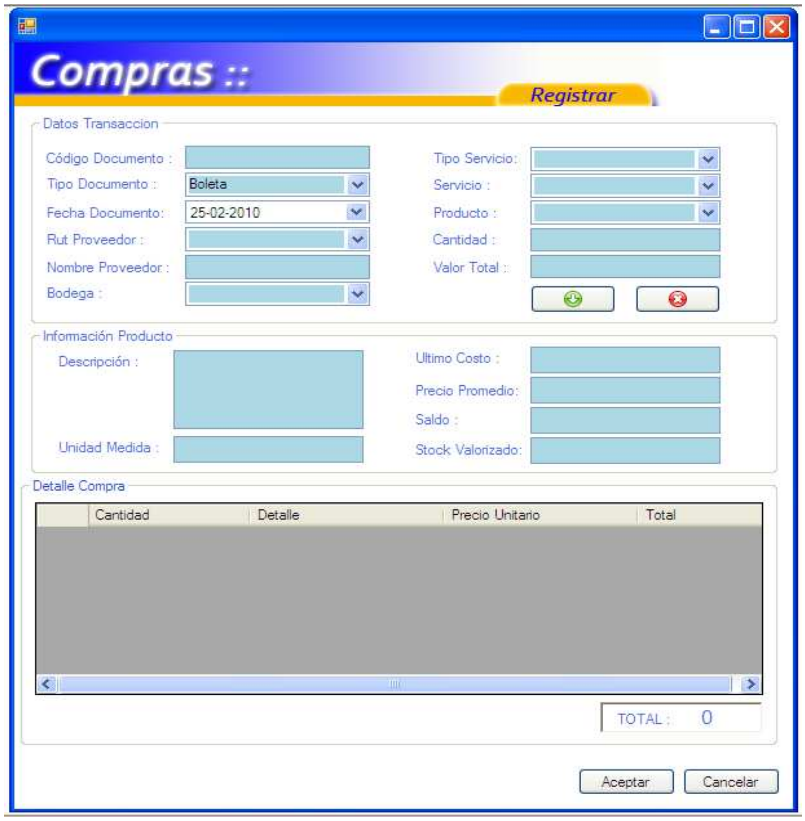
Contienen también, botones para agregar () y eliminar () productos al listado (DataGridView), los que inciden en el monto total de la transacción.

Además de los botones “Aceptar” o “Cancelar”, para confirmar o cancelar la transacción.

Los formularios que poseen esta estructura son:

- Menú Existencias: Compras, Ventas, Entrega de Materiales.

Ejemplo:



Formulario con Búsqueda y Actualización: Con un dato asociado a los ítems de un DataGridView.

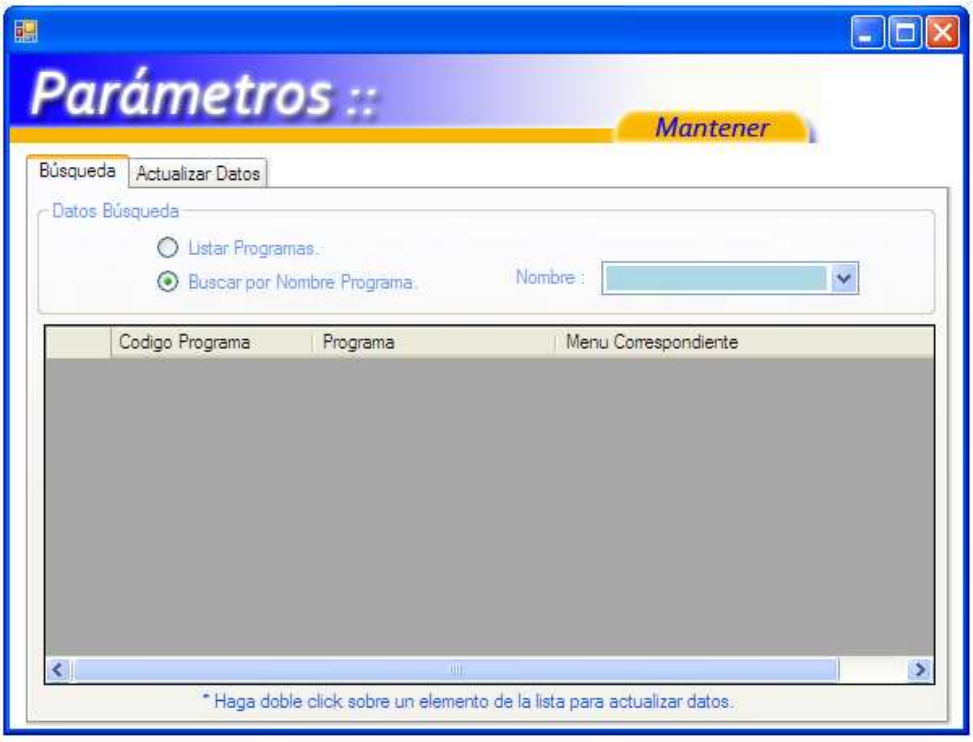
Este tipo de formulario es una variante del anterior de Búsqueda y Actualización, con la diferencia que en la pestaña de actualización posee un DataGridView en el que se pueden “Agregar” o “Eliminar” ítems, todos asociados a un dato en particular.

Los siguientes formularios poseen esta estructura:

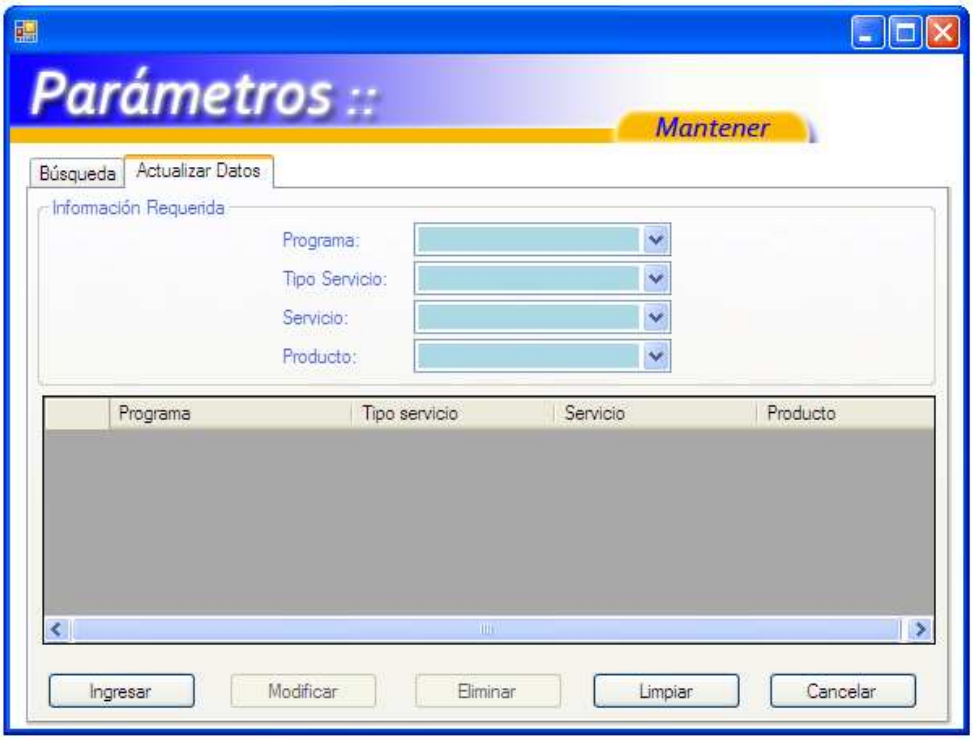
- Menú Control: Parámetros (parámetros asociados a un programa), Accesos Autorizados (programas asociados a un perfil de usuario).
- Menú Producción: Recetas (ingredientes asociados a un producto).

Ejemplo:

Búsqueda.



Actualizar Datos.



Menú Consumo Cliente.

El programa de consumo cliente está diseñado para abarcar los tipos de servicio Hotel y Restaurant (o cualquiera que implique una reserva). Por este motivo el combo box de selección está parametrizado con productos asociados a dichos tipos de servicio.

Para registrar el consumo de un cliente, debemos seleccionar el producto previamente reservado (Ej: Habitaciones, Mesas, etc.); luego presionar el botón “Aceptar”. Para cerrar o cancelar, se debe presionar el botón cerrar de la esquina superior derecha o presionar el botón “Cancelar”.

En el caso de haber un error se lanzan los siguientes mensajes de información:

a) “Debe seleccionar un producto antes de continuar”.
b) “No hay ninguna reserva asociada”.

Si la información está correcta, se da paso a la siguiente pantalla:

La información de consumo se carga automáticamente, es decir, “Consumo en”, “Fecha Inicio” y “Hora Inicio”.

Los datos del empleado y del cliente se generan de forma similar. Al seleccionar un Rut del combo box se cargan el “Nombre” y el “Apellido” asociados.

El “Tipo de servicio”, “Servicio” y “Producto”, pertenecientes a los datos del producto, se encuentran parametrizados. El “Precio Unitario” se carga automáticamente al seleccionar un producto.

Los datos de producto descritos anteriormente, junto con la “Cantidad” a consumir son necesarios y obligatorios para agregar un producto del gridview “Detalle Consumo”. Para esto, se debe presionar el botón “Agregar” (), si la información está correcta el producto se lista en dicho gridview, en caso contrario se lanza un mensaje de error.

Para modificar o eliminar un producto listado en el gridview “Detalle Consumo”, se debe hacer click sobre dicho producto, y luego presionar el botón “Editar” () o “Borrar” (), según corresponda. Resultado de estas operaciones se actualiza el “Total” del consumo.

Para confirmar el consumo, se necesitan obligatoriamente los datos del empleado y los del cliente, y por lo menos un producto listado en el gridview. Finalmente, se debe presionar el botón “Aceptar”. Si la información está correcta, se muestra un mensaje de éxito, en caso contrario, uno de fracaso.

Los mensajes de información generados al aceptar son:

a) “Ingrese los campos correctamente”.
b) “Debe ingresar al menos un producto”.
c) “No se ha podido registrar consumo”.

Las validaciones:

a) Sólo números.

b) Campos vacíos o nulos.

Si se quiere finiquitar el pago, o simplemente revisar el detalle de la deuda, se debe hacer click sobre el botón “Datos Pago”. Con lo que se dará paso a la siguiente pantalla:

En esta pantalla, se carga automáticamente tanto la información del cliente (“Rut”, “Nombre” y “Autorizado Crédito”), como el “detalle del pago”.

Para finiquitar el pago, se necesita definir la “forma de pago”; el “tipo de documento” con su respectivo número; y opcionalmente el descuento a aplicar, ya sea según porcentaje, o valor (presionar botón “Aplicar” según sea el caso)

Si toda la información esta correcta, se lanza un mensaje de éxito, en caso contrario, de fracaso.

Los mensajes de información generados al finiquitar son:

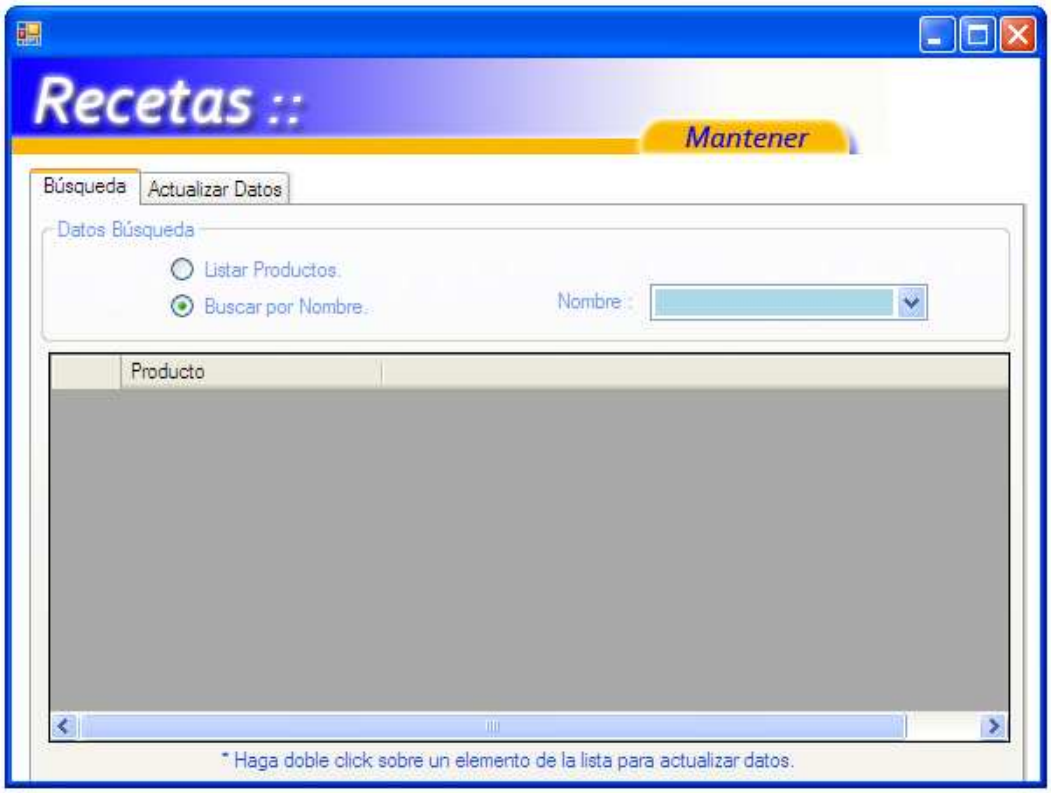
d) “Ingrese los campos correctamente”.
e) “Finiquito realizado con éxito”.
f) “No se ha podido realizar finiquito”.

Las validaciones:

a) Sólo números.

b) Campos vacíos o nulos.

Menú Producción – Recetas.



Al abrir el formulario “Recetas” se selecciona por defecto la pestaña “Búsqueda” del tabcontrol. En esta pestaña se pueden listar todos los productos registrados previamente con el atributo “Es Contenedor”, haciendo click sobre el radiobutton “Listar Productos”.

Para buscar un producto específico, se debe hacer click sobre el radiobutton “Buscar por Nombre”; luego seleccionar o escribir el nombre en el combo box “Nombre”.

En ambos casos, si se tiene éxito, se cargan los productos en el gridview. Dentro del gridview se puede navegar por los productos con las teclas “Arriba” o “Abajo” según corresponda. Para modificar o actualizar la información de la receta del producto, se debe hacer doble click, o presionar enter sobre el producto

seleccionado. Con esto último, se da paso a la segunda pestaña, “Actualizar Datos”.

La pestaña “Actualizar datos” posee 2 funcionalidades, dependiendo de la forma en que se active la pestaña. Una de ellas es “Ingreso de receta”, que se activará al hacer click sobre la pestaña, sin que previamente se haya hecho doble click, o enter sobre un producto (en la pestaña de búsqueda). En caso contrario se accede a “Modificar receta”.

Formulario Reservas/ Recepción.

El siguiente formulario contiene todas las opciones de reserva que maneja el sistema. Esta rutina está orientada a tipos de servicio con productos susceptibles de ser “Reservados” (Ej: Hotel, Restaurant, etc.).

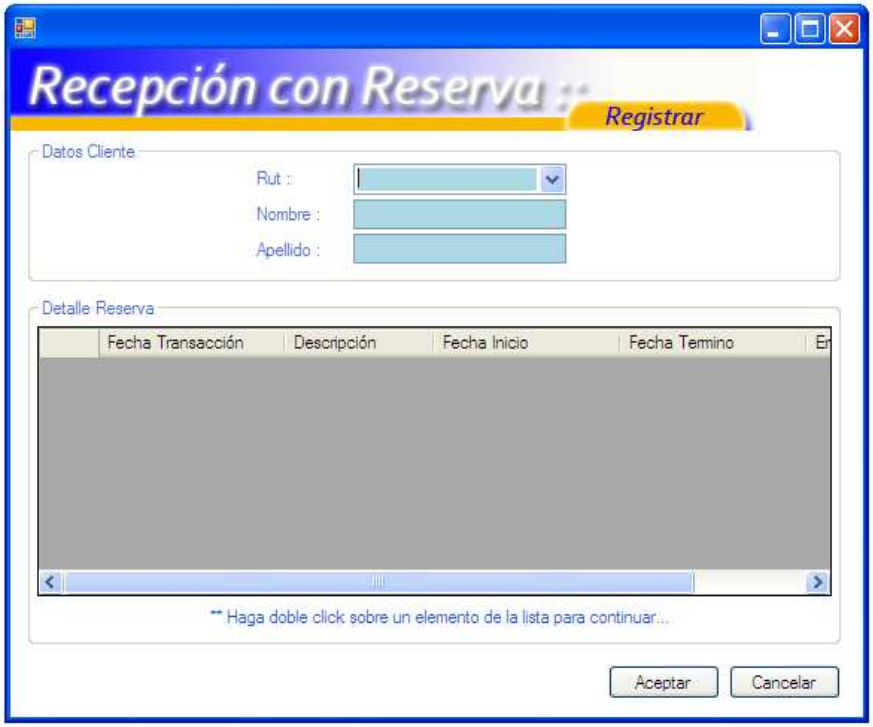
a) Reservar.

Para iniciar el proceso de reserva, se debe en primera instancia, presionar el botón “Buscar”. Con esto se desplegarán en el DataGridView todos los “Productos” disponibles (en este caso Habitaciones).

Luego, se debe hacer doble click, o presionar la tecla Enter sobre el ítem listado en el DataGridView, con esto se dará paso a la siguiente pantalla:

En esta pantalla se deben ingresar todos los datos requeridos. Para confirmar la reserva se debe presionar el botón “Aceptar”, en caso contrario, “Cancelar”. Con cualquiera de los dos botones se vuelve a la pantalla anterior.

b) Recepción con Reserva.



En esta ventana se debe seleccionar el RUT de un cliente que previamente realizó una reserva. Con esto se cargarán los datos de dicho cliente, además del respectivo detalle de reserva (en el DataGridView).

Para confirmar la acción se debe presionar el botón “Aceptar”, o hacer doble click sobre un ítem del DataGridView. Posteriormente se cargará la siguiente ventana:

Esta ventana es similar a la mostrada en “Reserva”, con la diferencia que se cargan los datos (no se ingresan). Para confirmar la recepción, se debe presionar el botón “Aceptar”, en caso contrario, “Cancelar”. Con cualquiera de los dos botones se vuelve a la pantalla anterior.

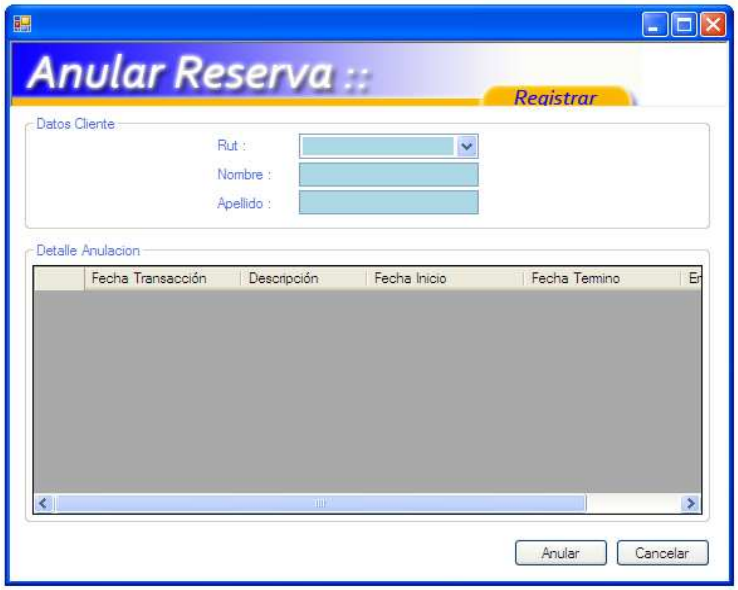
c) Recepción Sin Reserva.

Al igual que en la “Reserva”, se debe en primera instancia, presionar el botón “Buscar”. Con esto se desplegarán en el DataGridView todos los “Productos” disponibles (en este caso Habitaciones).

Luego, se debe hacer doble click, o presionar la tecla Enter sobre el ítem listado en el DataGridView, con esto se dará paso a la siguiente pantalla:

En esta pantalla se deben ingresar todos los datos requeridos. Para confirmar la reserva se debe presionar el botón “Aceptar”, en caso contrario, “Cancelar”. Con cualquiera de los dos botones se vuelve a la pantalla anterior.

d) Anular Reserva.



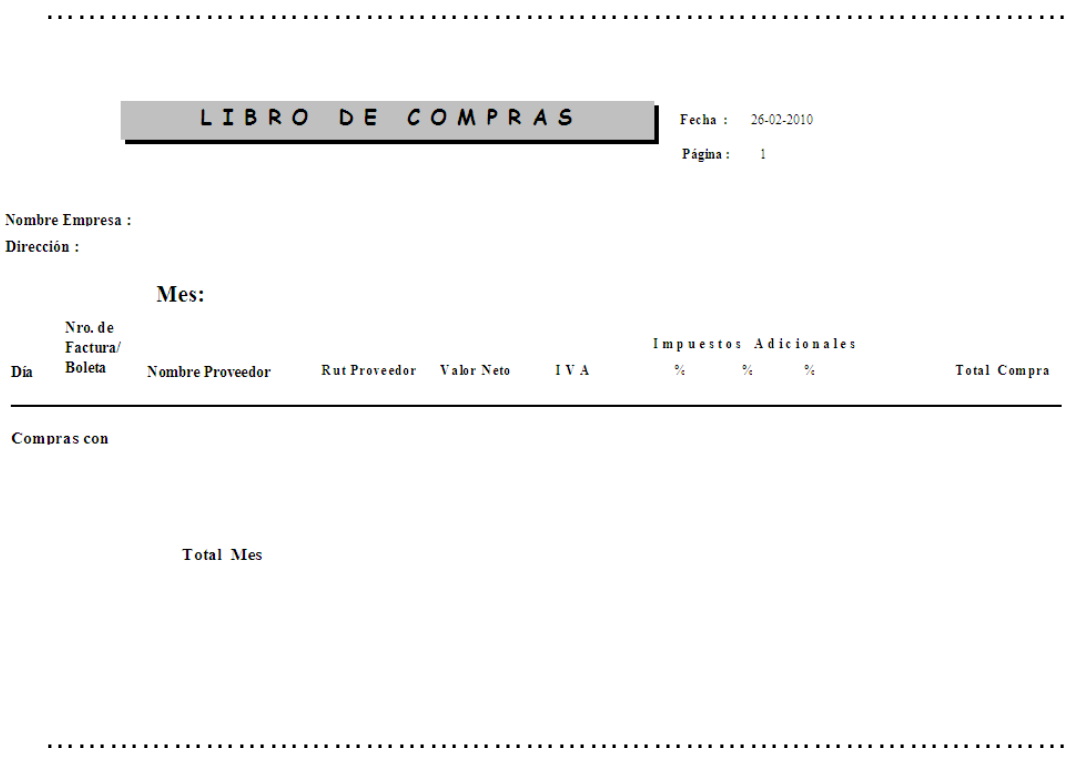
En esta ventana se debe seleccionar el RUT de un cliente que previamente realizó una reserva. Con esto se cargarán los datos de dicho cliente, además del respectivo detalle de reserva (en el DataGridView). Para anular la reserva, se debe presionar el botón “Anular”, o en caso contrario, “Cancelar”.

e) Registro Histórico.

Esta ventana muestra el historial de reservas sobre un producto determinado.

7.4.2 Informes de salida y consultas

Informes



LIBRO DE COMPRAS

Fecha : 26-02-2010

Página : 1

Nombre Empresa :

Dirección :

Mes:

Día	Nro. de Factura/ Boleta	Nombre Proveedor	Rut Proveedor	Valor Neto	I V A	Impuestos Adicionales %	%	%	Total Compra

Compras con

Total Mes

Detalle de Compras por Fechas

Fecha : 26-02-2010
Página : 1

Nombre Empresa : HOTEL MANQUEHUE
Dirección : Anibal Pinto 14

Fecha Inicio : **Fecha Término :**

Nombre del Producto	U.M.	Cantidad	Valor	Total Producto
01-01--4713				
0	Rut:	0		
		0.00	0,00	0
	Total 0			**0**
	Totals			**0**
	Total 01-01--4713			**0**
	Total Compras para Período			**0**

.....

Libro de Ventas con Boleta

Fecha : 26-02-2010

Página : 1

Nombre Empresa :

Dirección :

Mes : Año :

Dia	Boletas		Total Venta Diaria
	Del N°	Al N°	

Total Período

.....

Libro de Ventas con Factura

Fecha : 26-02-2010

Página : 1

Nombre Empresa :

Dirección :

Mes : Año :

Dia	Número Factura	Cliente	Rut Cliente	Neto	Iva	Tot

Total Mes

.....

Ventas Por Fecha

Fecha : 26-02-201
Página : 1

Nombre Empresa :
Dirección :

Fecha Inicio : 01-02-2010 **Fecha Término :** 28-02-2010

Día	Hora	Lugar Consumo	Numero Personas	Nombre Empleado	Valor Cuenta	Forma De Pago

Resumen Día : **Total Personas :** **Total Día :** $

Total Período : $

.....

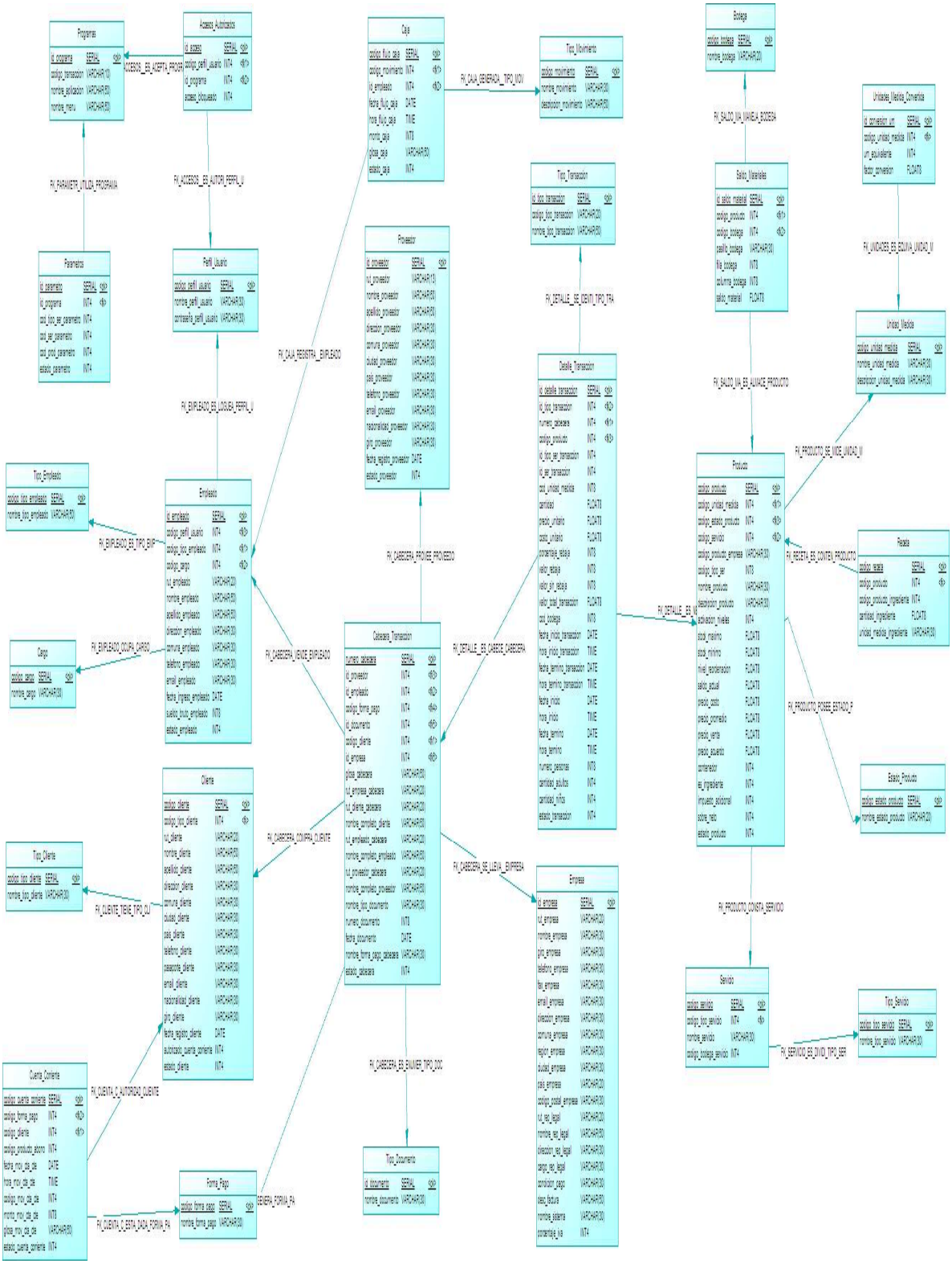
Consultas

a) Stock Crítico: muestra el nivel de stock que ha alcanzado un producto (Mínimo, Medio, Máximo).

b) Saldo en bodega: Muestra el saldo que posee un producto en una determinada bodega.

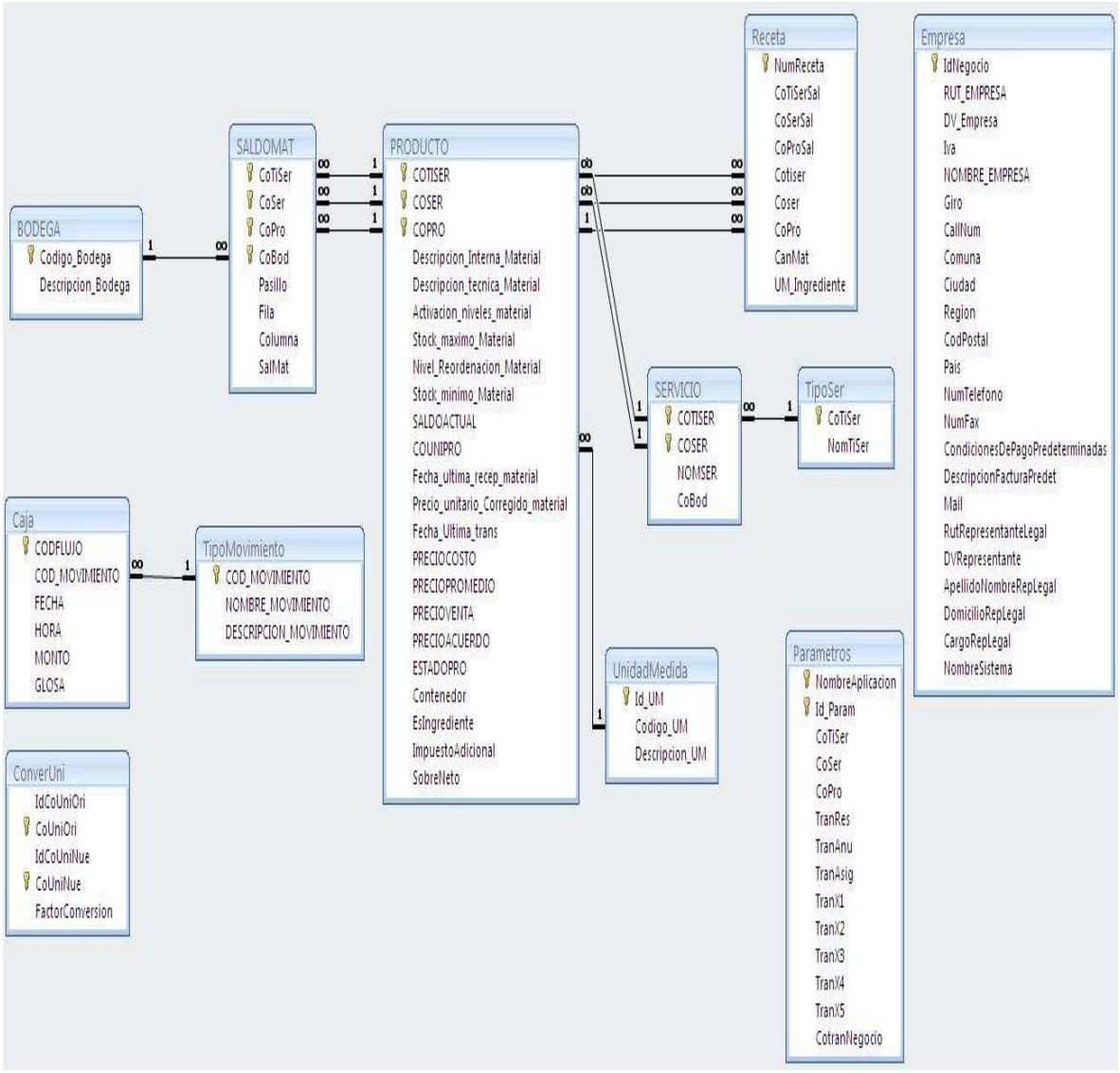
Capítulo 8:
Diseño Físico

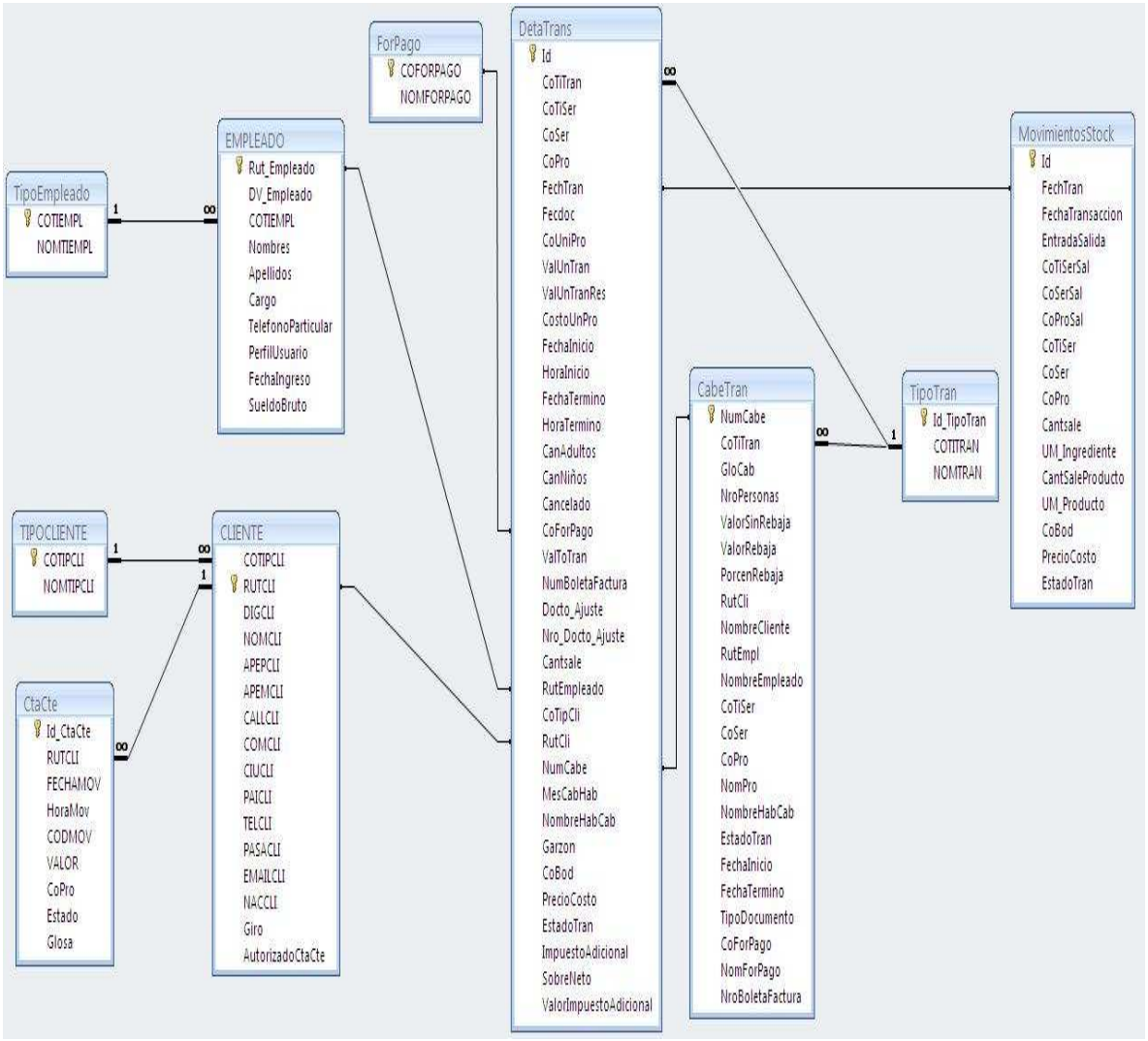
8.1 Modelo de datos físico



8.2 Diseño de base de datos

8.2.1 Diseño de tablas de Base de Datos





8.2.2 Script Base de Datos (código de generación básico)

```
/*==============================================================*/
/* DBMS name:      PostgreSQL 8                                 */
/* Created on:     23-02-2010 16:56:46                          */
/*==============================================================*/

/*==============================================================*/
/* Table: ACCESOS_AUTORIZADOS                                   */
/*==============================================================*/
create table ACCESOS_AUTORIZADOS (
   ID_ACCESO            SERIAL               not null,
   CODIGO_PERFIL_USUARIO INT4                 not null,
   ID_PROGRAMA          INT4                 not null,
   ACCESO_BLOQUEADO     INT4                 null,
   constraint PK_ACCESOS_AUTORIZADOS primary key (ID_ACCESO)
);

/*==============================================================*/
/* Index: ACCESOS_AUTORIZADOS_PK                                */
/*==============================================================*/
create unique index ACCESOS_AUTORIZADOS_PK on ACCESOS_AUTORIZADOS (
ID_ACCESO
);

/*==============================================================*/
/* Index: ES_AUTORIZADO_FK                                      */
/*==============================================================*/
create  index ES_AUTORIZADO_FK on ACCESOS_AUTORIZADOS (
CODIGO_PERFIL_USUARIO
);

/*==============================================================*/
/* Index: ES_ACEPTADO_FK                                        */
/*==============================================================*/
create  index ES_ACEPTADO_FK on ACCESOS_AUTORIZADOS (
ID_PROGRAMA
);

/*==============================================================*/
/* Table: BODEGA                                                */
/*==============================================================*/
create table BODEGA (
   CODIGO_BODEGA        SERIAL               not null,
   NOMBRE_BODEGA        VARCHAR(20)          null,
   constraint PK_BODEGA primary key (CODIGO_BODEGA)
);

/*==============================================================*/
/* Index: BODEGA_PK                                             */
/*==============================================================*/
create unique index BODEGA_PK on BODEGA (
CODIGO_BODEGA
```

```
);

/*==============================================================*/
/* Table: CABECERA_TRANSACCION                                  */
/*==============================================================*/
create table CABECERA_TRANSACCION (
   NUMERO_CABECERA      SERIAL               not null,
   ID_PROVEEDOR         INT4                 null,
   ID_EMPLEADO          INT4                 null,
   CODIGO_FORMA_PAGO    INT4                 null,
   ID_DOCUMENTO         INT4                 null,
   CODIGO_CLIENTE       INT4                 null,
   ID_EMPRESA           INT4                 null,
   GLOSA_CABECERA       VARCHAR(50)          null,
   RUT_EMPRESA_CABECERA VARCHAR(20)          null,
   RUT_CLIENTE_CABECERA VARCHAR(20)          null,
   NOMBRE_COMPLETO_CLIENTE VARCHAR(50)          null,
   RUT_EMPLEADO_CABECERA VARCHAR(20)          null,
   NOMBRE_COMPLETO_EMPLEADO VARCHAR(50)          null,
   RUT_PROVEEDOR_CABECERA VARCHAR(20)          null,
   NOMBRE_COMPLETO_PROVEEDOR VARCHAR(50)          null,
   NOMBRE_TIPO_DOCUMENTO VARCHAR(30)          null,
   NUMERO_DOCUMENTO     INT8                 null,
   FECHA_DOCUMENTO      DATE                 null,
   NOMBRE_FORMA_PAGO_CABECERA VARCHAR(30)          null,
   ESTADO_CABECERA      INT4                 null,
   constraint PK_CABECERA_TRANSACCION primary key (NUMERO_CABECERA)
);

/*==============================================================*/
/* Index: CABECERA_TRANSACCION_PK                               */
/*==============================================================*/
create unique index CABECERA_TRANSACCION_PK on CABECERA_TRANSACCION (
NUMERO_CABECERA
);

/*==============================================================*/
/* Index: COMPRA_FK                                             */
/*==============================================================*/
create  index COMPRA_FK on CABECERA_TRANSACCION (
CODIGO_CLIENTE
);

/*==============================================================*/
/* Index: VENDE_FK                                              */
/*==============================================================*/
create  index VENDE_FK on CABECERA_TRANSACCION (
ID_EMPLEADO
);

/*==============================================================*/
/* Index: PROVEE_FK                                             */
/*==============================================================*/
create  index PROVEE_FK on CABECERA_TRANSACCION (
ID_PROVEEDOR
);
```

```
/*==============================================================*/
/* Index: ES_GENERADA_FK                                        */
/*==============================================================*/
create  index ES_GENERADA_FK on CABECERA_TRANSACCION (
CODIGO_FORMA_PAGO
);

/*==============================================================*/
/* Index: ES_ENUMERADA_FK                                       */
/*==============================================================*/
create  index ES_ENUMERADA_FK on CABECERA_TRANSACCION (
ID_DOCUMENTO
);

/*==============================================================*/
/* Index: SE_LLEVA_A_CABO_EN_FK                                 */
/*==============================================================*/
create  index SE_LLEVA_A_CABO_EN_FK on CABECERA_TRANSACCION (
ID_EMPRESA
);

/*==============================================================*/
/* Table: CAJA                                                  */
/*==============================================================*/
create table CAJA (
   CODIGO_FLUJO_CAJA    SERIAL               not null,
   CODIGO_MOVIMIENTO    INT4                 not null,
   ID_EMPLEADO          INT4                 not null,
   FECHA_FLUJO_CAJA     DATE                 null,
   HORA_FLUJO_CAJA      TIME                 null,
   MONTO_CAJA           INT8                 null,
   GLOSA_CAJA           VARCHAR(50)          null,
   ESTADO_CAJA          INT4                 null,
   constraint PK_CAJA primary key (CODIGO_FLUJO_CAJA)
);

/*==============================================================*/
/* Index: CAJA_PK                                               */
/*==============================================================*/
create unique index CAJA_PK on CAJA (
CODIGO_FLUJO_CAJA
);

/*==============================================================*/
/* Index: GENERADA_POR_CAJA_FK                                  */
/*==============================================================*/
create  index GENERADA_POR_CAJA_FK on CAJA (
CODIGO_MOVIMIENTO
);

/*==============================================================*/
/* Index: REGISTRA_MOVIMIENTO_FK                                */
/*==============================================================*/
create  index REGISTRA_MOVIMIENTO_FK on CAJA (
ID_EMPLEADO
```

```
);

/*==============================================================*/
/* Table: CARGO                                                 */
/*==============================================================*/
create table CARGO (
   CODIGO_CARGO         SERIAL               not null,
   NOMBRE_CARGO         VARCHAR(30)          null,
   constraint PK_CARGO primary key (CODIGO_CARGO)
);

/*==============================================================*/
/* Index: CARGO_PK                                              */
/*==============================================================*/
create unique index CARGO_PK on CARGO (
CODIGO_CARGO
);

/*==============================================================*/
/* Table: CLIENTE                                               */
/*==============================================================*/
create table CLIENTE (
   CODIGO_CLIENTE       SERIAL               not null,
   CODIGO_TIPO_CLIENTE  INT4                 null,
   RUT_CLIENTE          VARCHAR(20) unique   null,
   NOMBRE_CLIENTE       VARCHAR(50)          null,
   APELLIDO_CLIENTE     VARCHAR(50)          null,
   DIRECCION_CLIENTE    VARCHAR(30)          null,
   COMUNA_CLIENTE       VARCHAR(30)          null,
   CIUDAD_CLIENTE       VARCHAR(30)          null,
   PAIS_CLIENTE         VARCHAR(30)          null,
   TELEFONO_CLIENTE     VARCHAR(30)          null,
   PASAPORTE_CLIENTE    VARCHAR(30)          null,
   EMAIL_CLIENTE        VARCHAR(30)          null,
   NACIONALIDAD_CLIENTE VARCHAR(30)          null,
   GIRO_CLIENTE         VARCHAR(30)          null,
   FECHA_REGISTRO_CLIENTE DATE                 null,
   AUTORIZADO_CUENTA_CORRIENTE INT4                 null,
   ESTADO_CLIENTE       INT4                 null,
   constraint PK_CLIENTE primary key (CODIGO_CLIENTE)
);

/*==============================================================*/
/* Index: CLIENTE_PK                                            */
/*==============================================================*/
create unique index CLIENTE_PK on CLIENTE (
CODIGO_CLIENTE
);

/*==============================================================*/
/* Index: TIENE_FK                                              */
/*==============================================================*/
create  index TIENE_FK on CLIENTE (
CODIGO_TIPO_CLIENTE
);
```

```
/*==============================================================*/
/* Table: CUENTA_CORRIENTE                                      */
/*==============================================================*/
create table CUENTA_CORRIENTE (
   CODIGO_CUENTA_CORRIENTE SERIAL                not null,
   CODIGO_FORMA_PAGO    INT4                  not null,
   CODIGO_CLIENTE       INT4                  not null,
   CODIGO_PRODUCTO_ABONO INT4                  null,
   FECHA_MOV_CTA_CTE    DATE                  null,
   HORA_MOV_CTA_CTE     TIME                  null,
   CODIGO_MOV_CTA_CTE   INT4                  null,
   MONTO_MOV_CTA_CTE    INT8                  null,
   GLOSA_MOV_CTA_CTE    VARCHAR(50)           null,
   ESTADO_CUENTA_CORRIENTE INT4                  null,
   constraint PK_CUENTA_CORRIENTE primary key (CODIGO_CUENTA_CORRIENTE)
);

/*==============================================================*/
/* Index: CUENTA_CORRIENTE_PK                                   */
/*==============================================================*/
create unique index CUENTA_CORRIENTE_PK on CUENTA_CORRIENTE (
CODIGO_CUENTA_CORRIENTE
);

/*==============================================================*/
/* Index: AUTORIZADO_FK                                         */
/*==============================================================*/
create  index AUTORIZADO_FK on CUENTA_CORRIENTE (
CODIGO_CLIENTE
);

/*==============================================================*/
/* Index: ESTA_DADA_FK                                          */
/*==============================================================*/
create  index ESTA_DADA_FK on CUENTA_CORRIENTE (
CODIGO_FORMA_PAGO
);

/*==============================================================*/
/* Table: DETALLE_TRANSACCION                                   */
/*==============================================================*/
create table DETALLE_TRANSACCION (
   ID_DETALLE_TRANSACCION SERIAL                 not null,
   ID_TIPO_TRANSACCION  INT4                  not null,
   NUMERO_CABECERA      INT4                  not null,
   CODIGO_PRODUCTO      INT4                  not null,
   ID_TIPO_SER_TRANSACCION INT4                  null,
   ID_SER_TRANSACCION   INT4                  null,
   COD_UNIDAD_MEDIDA    INT8                  null,
   CANTIDAD             FLOAT8                null,
   PRECIO_UNITARIO      FLOAT8                null,
   COSTO_UNITARIO       FLOAT8                null,
   PORCENTAJE_REBAJA    INT8                  null,
   VALOR_REBAJA         INT8                  null,
   VALOR_SIN_REBAJA     INT8                  null,
   VALOR_TOTAL_TRANSACCION FLOAT8                null,
```

```
   COD_BODEGA           INT8                null,
   FECHA_INICIO_TRANSACCION DATE               null,
   HORA_INICIO_TRANSACCION TIME               null,
   FECHA_TERMINO_TRANSACCION DATE               null,
   HORA_TERMINO_TRANSACCION TIME               null,
   FECHA_INICIO         DATE                null,
   HORA_INICIO          TIME                null,
   FECHA_TERMINO        DATE                null,
   HORA_TERMINO         TIME                null,
   NUMERO_PERSONAS      INT8                null,
   CANTIDAD_ADULTOS     INT4                null,
   CANTIDAD_NINOS       INT4                null,
   ESTADO_TRANSACCION   INT4                null,
   constraint PK_DETALLE_TRANSACCION primary key (ID_DETALLE_TRANSACCION)
);

/*==============================================================*/
/* Index: DETALLE_TRANSACCION_PK                                */
/*==============================================================*/
create unique index DETALLE_TRANSACCION_PK on DETALLE_TRANSACCION (
ID_DETALLE_TRANSACCION
);

/*==============================================================*/
/* Index: ES_CABECERA_FK                                        */
/*==============================================================*/
create  index ES_CABECERA_FK on DETALLE_TRANSACCION (
NUMERO_CABECERA
);

/*==============================================================*/
/* Index: SE_IDENTIFICA_FK                                      */
/*==============================================================*/
create  index SE_IDENTIFICA_FK on DETALLE_TRANSACCION (
ID_TIPO_TRANSACCION
);

/*==============================================================*/
/* Index: ES_NEGOCIADO_FK                                       */
/*==============================================================*/
create  index ES_NEGOCIADO_FK on DETALLE_TRANSACCION (
CODIGO_PRODUCTO
);

/*==============================================================*/
/* Table: EMPLEADO                                              */
/*==============================================================*/
create table EMPLEADO (
   ID_EMPLEADO          SERIAL              not null,
   CODIGO_PERFIL_USUARIO INT4                null,
   CODIGO_TIPO_EMPLEADO INT4                null,
   CODIGO_CARGO         INT4                null,
   RUT_EMPLEADO         VARCHAR(20) unique  null,
   NOMBRE_EMPLEADO      VARCHAR(50)         null,
   APELLIDO_EMPLEADO    VARCHAR(50)         null,
   DIRECCION_EMPLEADO   VARCHAR(30)         null,
```

```
   COMUNA_EMPLEADO      VARCHAR(30)          null,
   TELEFONO_EMPLEADO    VARCHAR(30)          null,
   EMAIL_EMPLEADO       VARCHAR(30)          null,
   FECHA_INGRESO_EMPLEADO DATE                 null,
   SUELDO_BRUTO_EMPLEADO INT8                 null,
   ESTADO_EMPLEADO      INT4                 null,
   constraint PK_EMPLEADO primary key (ID_EMPLEADO)
);

/*==============================================================*/
/* Index: EMPLEADO_PK                                           */
/*==============================================================*/
create unique index EMPLEADO_PK on EMPLEADO (
ID_EMPLEADO
);

/*==============================================================*/
/* Index: ES_FK                                                 */
/*==============================================================*/
create  index ES_FK on EMPLEADO (
CODIGO_TIPO_EMPLEADO
);

/*==============================================================*/
/* Index: OCUPA_FK                                              */
/*==============================================================*/
create  index OCUPA_FK on EMPLEADO (
CODIGO_CARGO
);

/*==============================================================*/
/* Index: ES_LOGUEADO_FK                                        */
/*==============================================================*/
create  index ES_LOGUEADO_FK on EMPLEADO (
CODIGO_PERFIL_USUARIO
);

/*==============================================================*/
/* Table: EMPRESA                                               */
/*==============================================================*/
create table EMPRESA (
   ID_EMPRESA           SERIAL               not null,
   RUT_EMPRESA          VARCHAR(20)          null,
   NOMBRE_EMPRESA       VARCHAR(30)          null,
   GIRO_EMPRESA         VARCHAR(30)          null,
   TELEFONO_EMPRESA     VARCHAR(30)          null,
   FAX_EMPRESA          VARCHAR(30)          null,
   EMAIL_EMPRESA        VARCHAR(30)          null,
   DIRECCION_EMPRESA    VARCHAR(30)          null,
   COMUNA_EMPRESA       VARCHAR(30)          null,
   REGION_EMPRESA       VARCHAR(30)          null,
   CIUDAD_EMPRESA       VARCHAR(30)          null,
   PAIS_EMPRESA         VARCHAR(20)          null,
   CODIGO_POSTAL_EMPRESA VARCHAR(30)          null,
   RUT_REP_LEGAL        VARCHAR(20)          null,
   NOMBRE_REP_LEGAL     VARCHAR(50)          null,
```

```
   DIRECCION_REP_LEGAL  VARCHAR(30)          null,
   CARGO_REP_LEGAL      VARCHAR(30)          null,
   CONDICION_PAGO       VARCHAR(30)          null,
   DESC_FACTURA         VARCHAR(50)          null,
   NOMBRE_SISTEMA       VARCHAR(30)          null,
   PORCENTAJE_IVA       INT4                 null,
   constraint PK_EMPRESA primary key (ID_EMPRESA)
);

/*==============================================================*/
/* Index: EMPRESA_PK                                            */
/*==============================================================*/
create unique index EMPRESA_PK on EMPRESA (
ID_EMPRESA
);

/*==============================================================*/
/* Table: ESTADO_PRODUCTO                                       */
/*==============================================================*/
create table ESTADO_PRODUCTO (
   CODIGO_ESTADO_PRODUCTO SERIAL               not null,
   NOMBRE_ESTADO_PRODUCTO VARCHAR(20)          null,
   constraint PK_ESTADO_PRODUCTO primary key (CODIGO_ESTADO_PRODUCTO)
);

/*==============================================================*/
/* Index: ESTADO_PRODUCTO_PK                                    */
/*==============================================================*/
create unique index ESTADO_PRODUCTO_PK on ESTADO_PRODUCTO (
CODIGO_ESTADO_PRODUCTO
);

/*==============================================================*/
/* Table: FORMA_PAGO                                            */
/*==============================================================*/
create table FORMA_PAGO (
   CODIGO_FORMA_PAGO    SERIAL               not null,
   NOMBRE_FORMA_PAGO    VARCHAR(30)          null,
   constraint PK_FORMA_PAGO primary key (CODIGO_FORMA_PAGO)
);

/*==============================================================*/
/* Index: FORMA_PAGO_PK                                         */
/*==============================================================*/
create unique index FORMA_PAGO_PK on FORMA_PAGO (
CODIGO_FORMA_PAGO
);

/*==============================================================*/
/* Table: PARAMETROS                                            */
/*==============================================================*/
create table PARAMETROS (
   ID_PARAMETRO         SERIAL               not null,
   ID_PROGRAMA          INT4                 not null,
   COD_TIPO_SER_PARAMETRO INT4                 null,
   COD_SER_PARAMETRO    INT4                 null,
```

```
   COD_PROD_PARAMETRO   INT4                 null,
   ESTADO_PARAMETRO     INT4                 null,
   constraint PK_PARAMETROS primary key (ID_PARAMETRO)
);

/*==============================================================*/
/* Index: PARAMETROS_PK                                         */
/*==============================================================*/
create unique index PARAMETROS_PK on PARAMETROS (
ID_PARAMETRO
);

/*==============================================================*/
/* Index: UTILIZA_FK                                            */
/*==============================================================*/
create  index UTILIZA_FK on PARAMETROS (
ID_PROGRAMA
);

/*==============================================================*/
/* Table: PERFIL_USUARIO                                        */
/*==============================================================*/
create table PERFIL_USUARIO (
   CODIGO_PERFIL_USUARIO SERIAL                not null,
   NOMBRE_PERFIL_USUARIO VARCHAR(30) unique        null,
   CONTRASENA_PERFIL_USUARIO VARCHAR(30)           null,
   constraint PK_PERFIL_USUARIO primary key (CODIGO_PERFIL_USUARIO)
);

/*==============================================================*/
/* Index: PERFIL_USUARIO_PK                                     */
/*==============================================================*/
create unique index PERFIL_USUARIO_PK on PERFIL_USUARIO (
CODIGO_PERFIL_USUARIO
);

/*==============================================================*/
/* Table: PRODUCTO                                              */
/*==============================================================*/
create table PRODUCTO (
   CODIGO_PRODUCTO      SERIAL               not null,
   CODIGO_UNIDAD_MEDIDA INT4                 not null,
   CODIGO_ESTADO_PRODUCTO INT4                 null,
   CODIGO_SERVICIO      INT4                 not null,
   CODIGO_PRODUCTO_EMPRESA VARCHAR(30) unique    null,
   CODIGO_TIPO_SER      INT8                 null,
   NOMBRE_PRODUCTO      VARCHAR(30)          null,
   DESCRIPCION_PRODUCTO VARCHAR(30)          null,
   ACTIVACION_NIVELES   INT4                 null,
   STOCK_MAXIMO         FLOAT8               null,
   STOCK_MINIMO         FLOAT8               null,
   NIVEL_REORDENACION   FLOAT8               null,
   SALDO_ACTUAL         FLOAT8               null,
   PRECIO_COSTO         FLOAT8               null,
   PRECIO_PROMEDIO      FLOAT8               null,
   PRECIO_VENTA         FLOAT8               null,
```

```
   PRECIO_ACUERDO       FLOAT8              null,
   CONTENEDOR           INT4                null,
   ES_INGREDIENTE       INT4                null,
   IMPUESTO_ADICIONAL   INT4                null,
   SOBRE_NETO           INT4                null,
   ESTADO_PRODUCTO      INT4                null,
   constraint PK_PRODUCTO primary key (CODIGO_PRODUCTO)
);

/*==============================================================*/
/* Index: PRODUCTO_PK                                           */
/*==============================================================*/
create unique index PRODUCTO_PK on PRODUCTO (
CODIGO_PRODUCTO
);

/*==============================================================*/
/* Index: SE_MIDE_FK                                            */
/*==============================================================*/
create  index SE_MIDE_FK on PRODUCTO (
CODIGO_UNIDAD_MEDIDA
);

/*==============================================================*/
/* Index: CONSTA_FK                                             */
/*==============================================================*/
create  index CONSTA_FK on PRODUCTO (
CODIGO_SERVICIO
);

/*==============================================================*/
/* Index: POSEE_FK                                              */
/*==============================================================*/
create  index POSEE_FK on PRODUCTO (
CODIGO_ESTADO_PRODUCTO
);

/*==============================================================*/
/* Table: PROGRAMAS                                             */
/*==============================================================*/
create table PROGRAMAS (
   ID_PROGRAMA          SERIAL              not null,
   CODIGO_TRANSACCION   VARCHAR(10)         null,
   NOMBRE_APLICACION    VARCHAR(50)         null,
   NOMBRE_MENU          VARCHAR(50)         null,
   constraint PK_PROGRAMAS primary key (ID_PROGRAMA)
);

/*==============================================================*/
/* Index: PROGRAMAS_PK                                          */
/*==============================================================*/
create unique index PROGRAMAS_PK on PROGRAMAS (
ID_PROGRAMA
);

/*==============================================================*/
```

```
/* Table: PROVEEDOR                                             */
/*==============================================================*/
create table PROVEEDOR (
   ID_PROVEEDOR         SERIAL               not null,
   RUT_PROVEEDOR        VARCHAR(13) unique      null,
   NOMBRE_PROVEEDOR     VARCHAR(50)          null,
   APELLIDO_PROVEEDOR   VARCHAR(50)          null,
   DIRECCION_PROVEEDOR  VARCHAR(30)          null,
   COMUNA_PROVEEDOR     VARCHAR(30)          null,
   CIUDAD_PROVEEDOR     VARCHAR(30)          null,
   PAIS_PROVEEDOR       VARCHAR(30)          null,
   TELEFONO_PROVEEDOR   VARCHAR(30)          null,
   EMAIL_PROVEEDOR      VARCHAR(30)          null,
   NACIONALIDAD_PROVEEDOR VARCHAR(30)          null,
   GIRO_PROVEEDOR       VARCHAR(30)          null,
   FECHA_REGISTRO_PROVEEDOR DATE                null,
   ESTADO_PROVEEDOR     INT4                 null,
   constraint PK_PROVEEDOR primary key (ID_PROVEEDOR)
);

/*==============================================================*/
/* Index: PROVEEDOR_PK                                          */
/*==============================================================*/
create unique index PROVEEDOR_PK on PROVEEDOR (
ID_PROVEEDOR
);

/*==============================================================*/
/* Table: RECETA                                                */
/*==============================================================*/
create table RECETA (
   CODIGO_RECETA        SERIAL               not null,
   CODIGO_PRODUCTO      INT4                 not null,
   CODIGO_PRODUCTO_INGREDIENTE INT4                    null,
   CANTIDAD_INGREDIENTE FLOAT8               null,
   UNIDAD_MEDIDA_INGREDIENTE VARCHAR(30)          null,
   constraint PK_RECETA primary key (CODIGO_RECETA)
);

/*==============================================================*/
/* Index: RECETA_PK                                             */
/*==============================================================*/
create unique index RECETA_PK on RECETA (
CODIGO_RECETA
);

/*==============================================================*/
/* Index: ES_CONTENEDOR_FK                                      */
/*==============================================================*/
create  index ES_CONTENEDOR_FK on RECETA (
CODIGO_PRODUCTO
);

/*==============================================================*/
/* Table: SALDO_MATERIALES                                      */
/*==============================================================*/
```

```
create table SALDO_MATERIALES (
   ID_SALDO_MATERIAL    SERIAL               not null,
   CODIGO_PRODUCTO      INT4                 not null,
   CODIGO_BODEGA        INT4                 not null,
   PASILLO_BODEGA       VARCHAR(30)          null,
   FILA_BODEGA          INT8                 null,
   COLUMNA_BODEGA       INT8                 null,
   SALDO_MATERIAL       FLOAT8               null,
   constraint PK_SALDO_MATERIALES primary key (ID_SALDO_MATERIAL)
);

/*==============================================================*/
/* Index: SALDO_MATERIALES_PK                                   */
/*==============================================================*/
create unique index SALDO_MATERIALES_PK on SALDO_MATERIALES (
ID_SALDO_MATERIAL
);

/*==============================================================*/
/* Index: ES_ALMACENADO_FK                                      */
/*==============================================================*/
create  index ES_ALMACENADO_FK on SALDO_MATERIALES (
CODIGO_PRODUCTO
);

/*==============================================================*/
/* Index: MANEJA_FK                                             */
/*==============================================================*/
create  index MANEJA_FK on SALDO_MATERIALES (
CODIGO_BODEGA
);

/*==============================================================*/
/* Table: SERVICIO                                              */
/*==============================================================*/
create table SERVICIO (
   CODIGO_SERVICIO      SERIAL               not null,
   CODIGO_TIPO_SERVICIO INT4                 not null,
   NOMBRE_SERVICIO      VARCHAR(30)          null,
   CODIGO_BODEGA_SERVICIO INT4                  null,
   constraint PK_SERVICIO primary key (CODIGO_SERVICIO)
);

/*==============================================================*/
/* Index: SERVICIO_PK                                           */
/*==============================================================*/
create unique index SERVICIO_PK on SERVICIO (
CODIGO_SERVICIO
);

/*==============================================================*/
/* Index: ES_DIVIDIDO_FK                                        */
/*==============================================================*/
create  index ES_DIVIDIDO_FK on SERVICIO (
CODIGO_TIPO_SERVICIO
);
```

```
/*==============================================================*/
/* Table: TIPO_CLIENTE                                          */
/*==============================================================*/
create table TIPO_CLIENTE (
   CODIGO_TIPO_CLIENTE  SERIAL               not null,
   NOMBRE_TIPO_CLIENTE  VARCHAR(30)          null,
   constraint PK_TIPO_CLIENTE primary key (CODIGO_TIPO_CLIENTE)
);

/*==============================================================*/
/* Index: TIPO_CLIENTE_PK                                       */
/*==============================================================*/
create unique index TIPO_CLIENTE_PK on TIPO_CLIENTE (
CODIGO_TIPO_CLIENTE
);

/*==============================================================*/
/* Table: TIPO_DOCUMENTO                                        */
/*==============================================================*/
create table TIPO_DOCUMENTO (
   ID_DOCUMENTO         SERIAL               not null,
   NOMBRE_DOCUMENTO     VARCHAR(30)          null,
   constraint PK_TIPO_DOCUMENTO primary key (ID_DOCUMENTO)
);

/*==============================================================*/
/* Index: TIPO_DOCUMENTO_PK                                     */
/*==============================================================*/
create unique index TIPO_DOCUMENTO_PK on TIPO_DOCUMENTO (
ID_DOCUMENTO
);

/*==============================================================*/
/* Table: TIPO_EMPLEADO                                         */
/*==============================================================*/
create table TIPO_EMPLEADO (
   CODIGO_TIPO_EMPLEADO SERIAL               not null,
   NOMBRE_TIPO_EMPLEADO VARCHAR(50)          null,
   constraint PK_TIPO_EMPLEADO primary key (CODIGO_TIPO_EMPLEADO)
);

/*==============================================================*/
/* Index: TIPO_EMPLEADO_PK                                      */
/*==============================================================*/
create unique index TIPO_EMPLEADO_PK on TIPO_EMPLEADO (
CODIGO_TIPO_EMPLEADO
);

/*==============================================================*/
/* Table: TIPO_MOVIMIENTO                                       */
/*==============================================================*/
create table TIPO_MOVIMIENTO (
   CODIGO_MOVIMIENTO    SERIAL               not null,
   NOMBRE_MOVIMIENTO    VARCHAR(30)          null,
   DESCRIPCION_MOVIMIENTO VARCHAR(50)          null,
```

```
   constraint PK_TIPO_MOVIMIENTO primary key (CODIGO_MOVIMIENTO)
);

/*==============================================================*/
/* Index: TIPO_MOVIMIENTO_PK                                    */
/*==============================================================*/
create unique index TIPO_MOVIMIENTO_PK on TIPO_MOVIMIENTO (
CODIGO_MOVIMIENTO
);

/*==============================================================*/
/* Table: TIPO_SERVICIO                                         */
/*==============================================================*/
create table TIPO_SERVICIO (
   CODIGO_TIPO_SERVICIO SERIAL               not null,
   NOMBRE_TIPO_SERVICIO VARCHAR(30)          null,
   constraint PK_TIPO_SERVICIO primary key (CODIGO_TIPO_SERVICIO)
);

/*==============================================================*/
/* Index: TIPO_SERVICIO_PK                                      */
/*==============================================================*/
create unique index TIPO_SERVICIO_PK on TIPO_SERVICIO (
CODIGO_TIPO_SERVICIO
);

/*==============================================================*/
/* Table: TIPO_TRANSACCION                                      */
/*==============================================================*/
create table TIPO_TRANSACCION (
   ID_TIPO_TRANSACCION  SERIAL               not null,
   CODIGO_TIPO_TRANSACCION VARCHAR(20)          null,
   NOMBRE_TIPO_TRANSACCION VARCHAR(50)          null,
   constraint PK_TIPO_TRANSACCION primary key (ID_TIPO_TRANSACCION)
);

/*==============================================================*/
/* Index: TIPO_TRANSACCION_PK                                   */
/*==============================================================*/
create unique index TIPO_TRANSACCION_PK on TIPO_TRANSACCION (
ID_TIPO_TRANSACCION
);

/*==============================================================*/
/* Table: UNIDADES_MEDIDA_CONVERTIDA                            */
/*==============================================================*/
create table UNIDADES_MEDIDA_CONVERTIDA (
   ID_CONVERSION_UM     SERIAL               not null,
   CODIGO_UNIDAD_MEDIDA INT4                 not null,
   UM_EQUIVALENTE       INT4                 null,
   FACTOR_CONVERSION    FLOAT8               null,
   constraint PK_UNIDADES_MEDIDA_CONVERTIDA primary key
(ID_CONVERSION_UM)
);

/*==============================================================*/
```

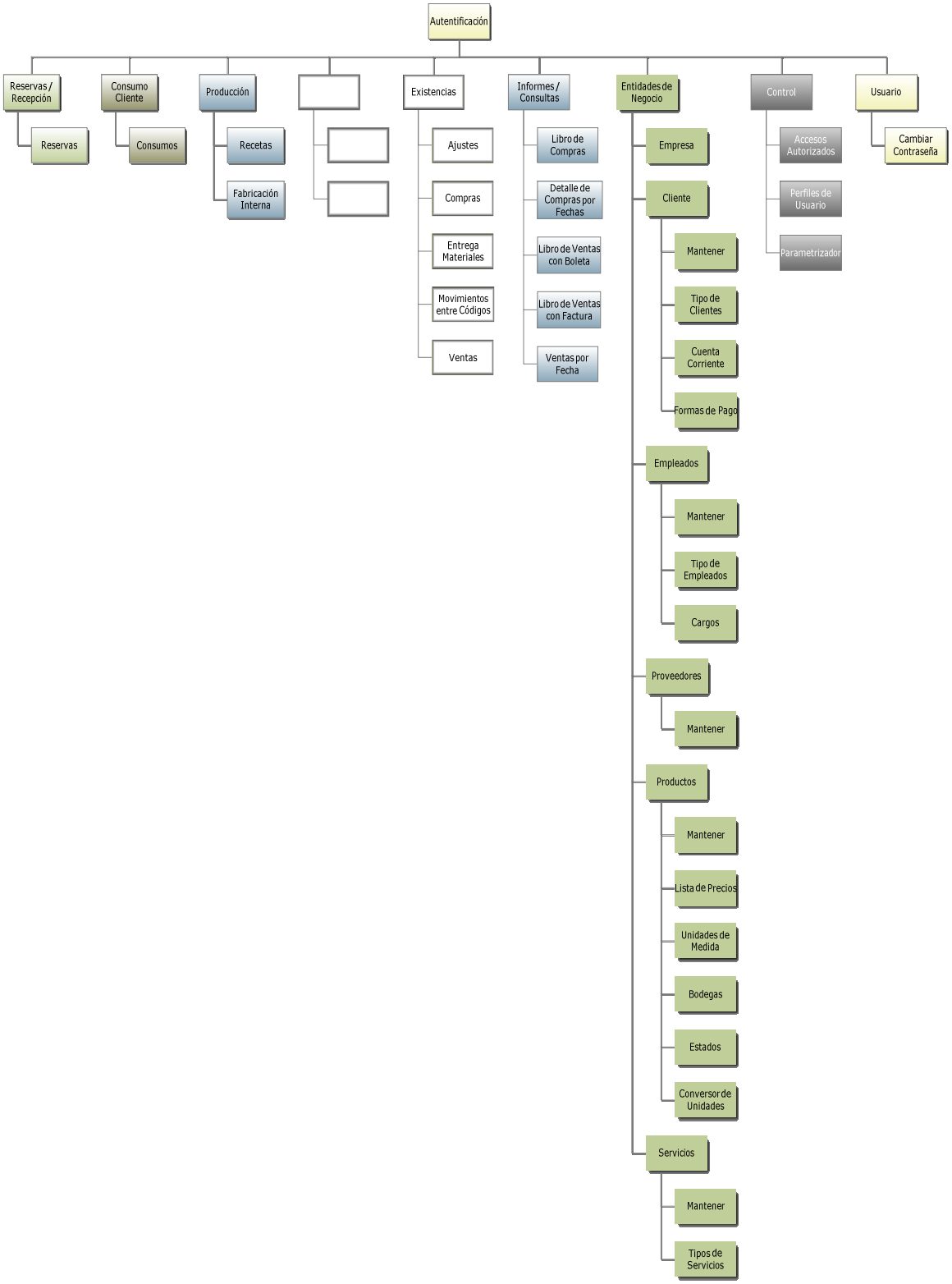
```
/* Index: UNIDADES_MEDIDA_CONVERTIDA_PK                          */
/*==============================================================*/
create unique index UNIDADES_MEDIDA_CONVERTIDA_PK on
UNIDADES_MEDIDA_CONVERTIDA (
ID_CONVERSION_UM
);

/*==============================================================*/
/* Index: ES_EQUIVALENTE_FK                                     */
/*==============================================================*/
create  index ES_EQUIVALENTE_FK on UNIDADES_MEDIDA_CONVERTIDA (
CODIGO_UNIDAD_MEDIDA
);

/*==============================================================*/
/* Table: UNIDAD_MEDIDA                                         */
/*==============================================================*/
create table UNIDAD_MEDIDA (
   CODIGO_UNIDAD_MEDIDA SERIAL               not null,
   NOMBRE_UNIDAD_MEDIDA VARCHAR(30)          null,
   DESCRIPCION_UNIDAD_MEDIDA VARCHAR(30)          null,
   constraint PK_UNIDAD_MEDIDA primary key (CODIGO_UNIDAD_MEDIDA)
);

/*==============================================================*/
/* Index: UNIDAD_MEDIDA_PK                                      */
/*==============================================================*/
create unique index UNIDAD_MEDIDA_PK on UNIDAD_MEDIDA (
CODIGO_UNIDAD_MEDIDA
);

alter table ACCESOS_AUTORIZADOS
   add constraint FK_ACCESOS__ES_ACEPTA_PROGRAMA foreign key
(ID_PROGRAMA)
      references PROGRAMAS (ID_PROGRAMA)
      on delete restrict on update restrict;

alter table ACCESOS_AUTORIZADOS
   add constraint FK_ACCESOS__ES_AUTORI_PERFIL_U foreign key
(CODIGO_PERFIL_USUARIO)
      references PERFIL_USUARIO (CODIGO_PERFIL_USUARIO)
      on delete restrict on update restrict;

alter table CABECERA_TRANSACCION
   add constraint FK_CABECERA_COMPRA_CLIENTE foreign key (CODIGO_CLIENTE)
      references CLIENTE (CODIGO_CLIENTE)
      on delete restrict on update restrict;

alter table CABECERA_TRANSACCION
   add constraint FK_CABECERA_ES_ENUMER_TIPO_DOC foreign key
(ID_DOCUMENTO)
      references TIPO_DOCUMENTO (ID_DOCUMENTO)
      on delete restrict on update restrict;

alter table CABECERA_TRANSACCION
   add constraint FK_CABECERA_ES_GENERA_FORMA_PA foreign key
```

```
(CODIGO_FORMA_PAGO)
      references FORMA_PAGO (CODIGO_FORMA_PAGO)
      on delete restrict on update restrict;

alter table CABECERA_TRANSACCION
   add constraint FK_CABECERA_PROVEE_PROVEEDO foreign key (ID_PROVEEDOR)
      references PROVEEDOR (ID_PROVEEDOR)
      on delete restrict on update restrict;

alter table CABECERA_TRANSACCION
   add constraint FK_CABECERA_SE_LLEVA__EMPRESA foreign key (ID_EMPRESA)
      references EMPRESA (ID_EMPRESA)
      on delete restrict on update restrict;

alter table CABECERA_TRANSACCION
   add constraint FK_CABECERA_VENDE_EMPLEADO foreign key (ID_EMPLEADO)
      references EMPLEADO (ID_EMPLEADO)
      on delete restrict on update restrict;

alter table CAJA
   add constraint FK_CAJA_GENERADA__TIPO_MOV foreign key
(CODIGO_MOVIMIENTO)
      references TIPO_MOVIMIENTO (CODIGO_MOVIMIENTO)
      on delete restrict on update restrict;

alter table CAJA
   add constraint FK_CAJA_REGISTRA__EMPLEADO foreign key (ID_EMPLEADO)
      references EMPLEADO (ID_EMPLEADO)
      on delete restrict on update restrict;

alter table CLIENTE
   add constraint FK_CLIENTE_TIENE_TIPO_CLI foreign key
(CODIGO_TIPO_CLIENTE)
      references TIPO_CLIENTE (CODIGO_TIPO_CLIENTE)
      on delete restrict on update restrict;

alter table CUENTA_CORRIENTE
   add constraint FK_CUENTA_C_AUTORIZAD_CLIENTE foreign key
(CODIGO_CLIENTE)
      references CLIENTE (CODIGO_CLIENTE)
      on delete restrict on update restrict;

alter table CUENTA_CORRIENTE
   add constraint FK_CUENTA_C_ESTA_DADA_FORMA_PA foreign key
(CODIGO_FORMA_PAGO)
      references FORMA_PAGO (CODIGO_FORMA_PAGO)
      on delete restrict on update restrict;

alter table DETALLE_TRANSACCION
   add constraint FK_DETALLE__ES_CABECE_CABECERA foreign key
(NUMERO_CABECERA)
      references CABECERA_TRANSACCION (NUMERO_CABECERA)
      on delete restrict on update restrict;

alter table DETALLE_TRANSACCION
   add constraint FK_DETALLE__ES_NEGOCI_PRODUCTO foreign key
```

```
(CODIGO_PRODUCTO)
      references PRODUCTO (CODIGO_PRODUCTO)
      on delete restrict on update restrict;

alter table DETALLE_TRANSACCION
   add constraint FK_DETALLE__SE_IDENTI_TIPO_TRA foreign key
(ID_TIPO_TRANSACCION)
      references TIPO_TRANSACCION (ID_TIPO_TRANSACCION)
      on delete restrict on update restrict;

alter table EMPLEADO
   add constraint FK_EMPLEADO_ES_TIPO_EMP foreign key
(CODIGO_TIPO_EMPLEADO)
      references TIPO_EMPLEADO (CODIGO_TIPO_EMPLEADO)
      on delete restrict on update restrict;

alter table EMPLEADO
   add constraint FK_EMPLEADO_ES_LOGUEA_PERFIL_U foreign key
(CODIGO_PERFIL_USUARIO)
      references PERFIL_USUARIO (CODIGO_PERFIL_USUARIO)
      on delete restrict on update restrict;

alter table EMPLEADO
   add constraint FK_EMPLEADO_OCUPA_CARGO foreign key (CODIGO_CARGO)
      references CARGO (CODIGO_CARGO)
      on delete restrict on update restrict;

alter table PARAMETROS
   add constraint FK_PARAMETR_UTILIZA_PROGRAMA foreign key (ID_PROGRAMA)
      references PROGRAMAS (ID_PROGRAMA)
      on delete restrict on update restrict;

alter table PRODUCTO
   add constraint FK_PRODUCTO_CONSTA_SERVICIO foreign key
(CODIGO_SERVICIO)
      references SERVICIO (CODIGO_SERVICIO)
      on delete restrict on update restrict;

alter table PRODUCTO
   add constraint FK_PRODUCTO_POSEE_ESTADO_P foreign key
(CODIGO_ESTADO_PRODUCTO)
      references ESTADO_PRODUCTO (CODIGO_ESTADO_PRODUCTO)
      on delete restrict on update restrict;

alter table PRODUCTO
   add constraint FK_PRODUCTO_SE_MIDE_UNIDAD_M foreign key
(CODIGO_UNIDAD_MEDIDA)
      references UNIDAD_MEDIDA (CODIGO_UNIDAD_MEDIDA)
      on delete restrict on update restrict;

alter table RECETA
   add constraint FK_RECETA_ES_CONTEN_PRODUCTO foreign key
(CODIGO_PRODUCTO)
      references PRODUCTO (CODIGO_PRODUCTO)
      on delete restrict on update restrict;
```

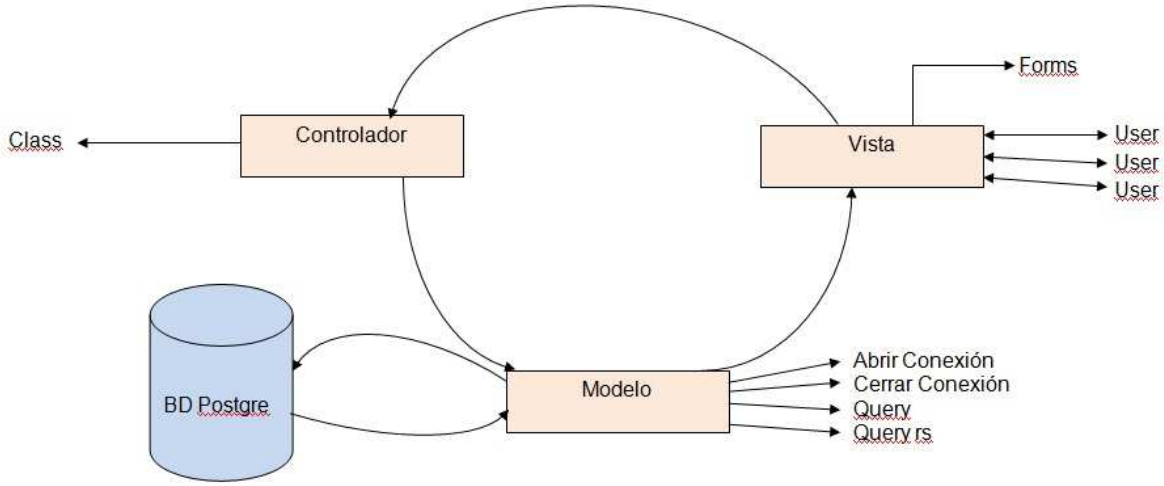
```
alter table SALDO_MATERIALES
   add constraint FK_SALDO_MA_ES_ALMACE_PRODUCTO foreign key
(CODIGO_PRODUCTO)
      references PRODUCTO (CODIGO_PRODUCTO)
      on delete restrict on update restrict;

alter table SALDO_MATERIALES
   add constraint FK_SALDO_MA_MANEJA_BODEGA foreign key (CODIGO_BODEGA)
      references BODEGA (CODIGO_BODEGA)
      on delete restrict on update restrict;

alter table SERVICIO
   add constraint FK_SERVICIO_ES_DIVIDI_TIPO_SER foreign key
(CODIGO_TIPO_SERVICIO)
      references TIPO_SERVICIO (CODIGO_TIPO_SERVICIO)
      on delete restrict on update restrict;

alter table UNIDADES_MEDIDA_CONVERTIDA
   add constraint FK_UNIDADES_ES_EQUIVA_UNIDAD_M foreign key
(CODIGO_UNIDAD_MEDIDA)
      references UNIDAD_MEDIDA (CODIGO_UNIDAD_MEDIDA)
      on delete restrict on update restrict;
```

8.3 Menú de Navegación de sistema



Capítulo 9: Construcción

9.1 Modelo Vista – Controlador del Sistema



En este esquema podemos observar que en Vista se cargan los formularios y es donde los usuarios interactúan con el sistema. La información ingresada por el usuario es manejada por el controlador, el cual contiene las clases con sus respectivos atributos y métodos. Posteriormente esta información es pasada al modelo, el cual se encarga de la conexión y realizar las consultas correspondientes con la base de datos. Finalmente la información cargada es transferida a Vista para entregar al usuario las acciones y/o información requerida. Es importante mencionar que cada vez que se realiza un ingreso, edición, eliminación o consulta de información, el Modelo abre la conexión para realizar la acción requerida y una vez terminada la acción inmediatamente cierra la conexión con la base de datos.

9.2 Conversión

Cuando una empresa adquiera un nuevo sistema, es necesario saber si ya se contaba con una versión anterior del sistema, un sistema antiguo similar al nuevo o no se contaba con un sistema. Las 2 primeras opciones requieren de una fase de conversión, en donde los 2 sistemas trabajan a la par, esto para conocer y probar el nuevo sistema sin dejar de operar en el sistema antiguo. La fase de conversión se realiza principalmente para la detección de errores en el nuevo sistema. La duración de la fase de conversión se debe estipular dentro de cada empresa. Generalmente, sistemas como este requieren de una fase de conversión de entre 1 y 3 meses. La fase de conversión implica un cambio de escenario desde una forma de trabajo a otra.

La conversión de Procesos trata de:

- Aumentar el grado de automatización de funciones.
- Cubrir nuevas funciones.
- Suprimir funcionalidades.

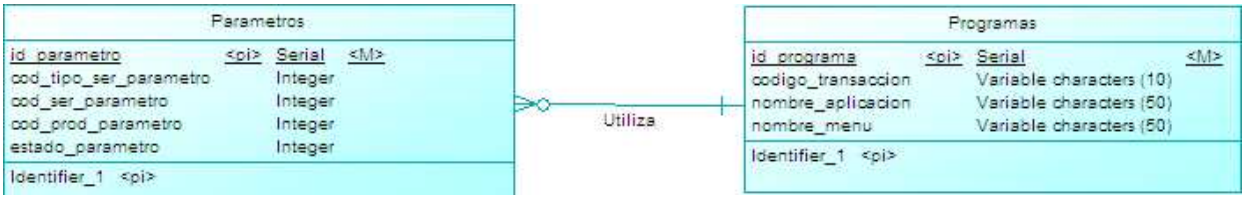
Se presentan repercusión en el cambio:

- En procedimientos administrativos.
- En procedimientos computacionales.
- En operación de sistemas.
- En archivos.
- En funciones relacionadas.

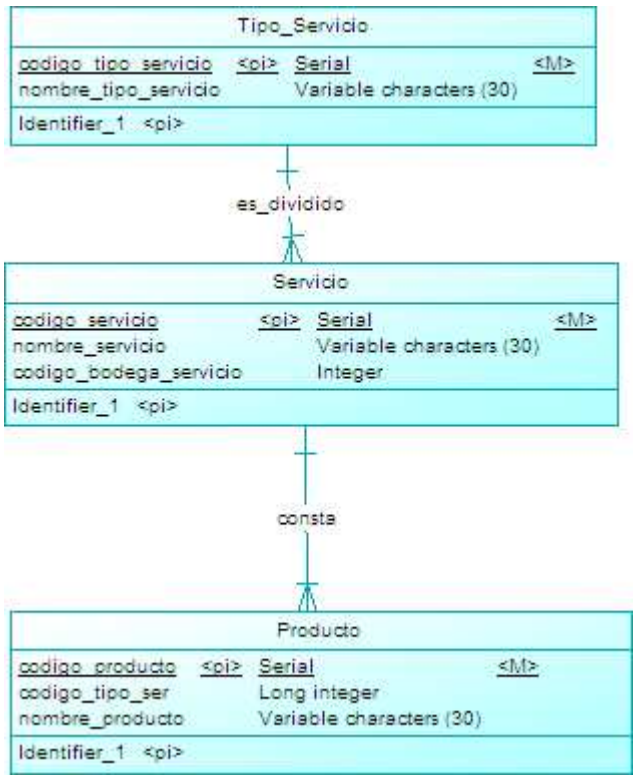
9.3 Parametrización en el Sistema

1.- Parametrización de combo box.

La implementación lógica de la parametrización en la base de datos, contempla 2 tablas: “Programa” y “Parámetros”, y la relación entre ellas de la forma “uno a muchos”. El esquema es el siguiente:



También contempla el esquema jerárquico de Tipo de servicio, Servicio y Producto:



Se puede apreciar que la tabla parámetros contiene el tipo de servicio, servicios y productos, que podrán ser accedidos por un programa en particular. En definitiva, dichos datos en conjunto podrían definirse como “parámetros” del programa.

Se identifican 3 posibles casos en la estructura de los parámetros:

1.- Sólo se ingresa tipo de servicio: en este caso se muestran todos los productos, de los servicios asociados al tipo de servicio.

2.- Se ingresa tipo de servicio y servicio: se muestran los productos de dicho servicio asociado al tipo de servicio.

3.- Se ingresa tipo de servicio, servicio y producto: se muestra el producto específico del servicio asociado al tipo de servicio.

La implementación a nivel de código se centró principalmente en el filtrado de combo box. Esto permite mayor simpleza y entendimiento de esta funcionalidad. Se identifican 2 variaciones en la parametrización de combo box:

a) Filtro de un combo box (Productos):

Algunos programas (Forms) del sistema contienen un solo combo box parametrizado (Ej: Cuenta Corriente). En este caso se creó una función “Cargar Parámetros”, que toma como referencia el id_programa (de la tabla “Programa”), y obtiene los parámetros asociados en la tabla “Parámetros”.

De acuerdo a la estructura de los parámetros (descrita anteriormente), se cargan los productos en el combo box.

b) Filtro de tres combo box (Tipo servicio, servicio y producto):

Se centra en el evento de seleccionar cada combo box de la jerarquía hasta llegar al producto. También existe una función similar a la del caso anterior, que en primera instancia carga los tipos de servicio (definidos en la tabla “Parámetros”, para el programa determinado).

Al seleccionar un ítem del combo box “Tipo Servicio”, se cargan automáticamente los servicios asociados en el combo box contiguo, es decir, “Servicio”.

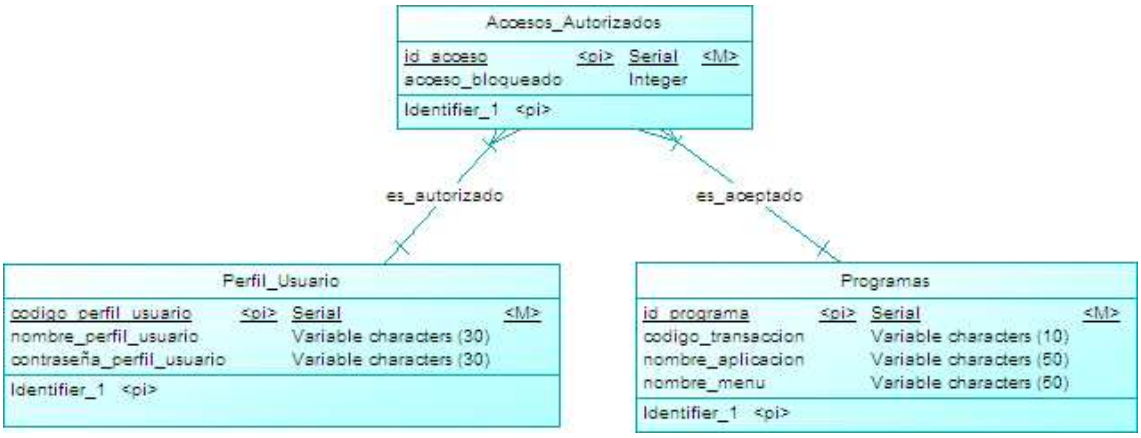
De igual forma, al seleccionar un ítem del combo box “Servicios”, se cargan automáticamente los productos en su combo box correspondiente.

Todo lo anterior se debe hacer respetando la información que se encuentra en la tabla parámetros.

2.- Accesos Autorizados.

Otra forma de parametrización implementada es el control de accesos autorizados, que permite definir los formularios e ítems de la barra de menú, que serán visibles al usuario.

A nivel lógico, en la base de datos, se tienen las siguientes tablas, con sus respectivas relaciones:

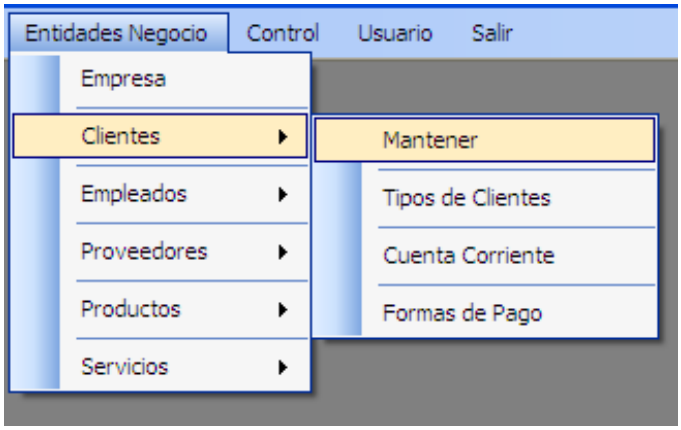


Como se puede ver, la tabla “Accesos Autorizados” actúa como tabla intermedia, que contendrá los “Programas” visibles para un “Perfil de usuario”.

En la implementación a nivel de código pone como restricción, en primera instancia que todos los ítems del menú tengan su correspondiente en la tabla “Programas”. Además, que el atributo “nombre_menu” de la tabla “Programas” sea equivalente con el nombre del ítem perteneciente a la barra menú del sistema. Además se requiere

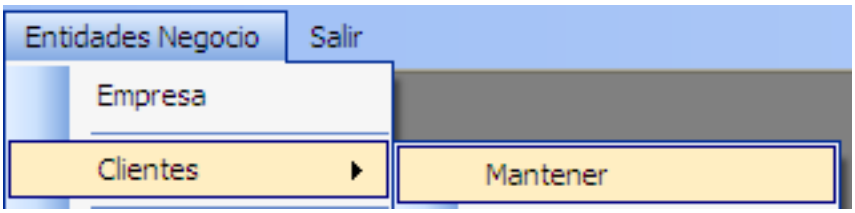
Es importante destacar, que el nombre de dicho ítem de menú no necesariamente posee un nombre igual al texto que proyecta.

Similar al otro tipo de parametrización, “Accesos Autorizados” funciona de manera jerárquica. Esto se explica con el siguiente ejemplo:



Si se quiere dar a un usuario la visibilidad sobre el menú “Empresa”, y “Mantener”, el primer paso es agregar en la tabla accesos autorizados el programa “EntidadesNegocio” (el de más alto nivel). Luego se debe ingresar el programa “Empresa”. El siguiente paso es ingresar el programa “Clientes”. Finalmente, se debe incorporar el programa “Mantener”.

Con todo esto, la visibilidad para el usuario quedaría:



Capítulo 10: Pruebas del sistema

Para realizar las pruebas se considerarán diversas situaciones, las cuales se nombran a continuación:

1. Realizar una reserva en un hotel, de una habitación con una determinada fecha, la cual esté ocupada para tal fecha.

   Posibles salidas:

   - Reserva realizada con éxito. Esto significa que la funcionalidad es errónea.
   - Reserva arroja mensaje de error, ya que la habitación está ocupada. Esto significa que la funcionalidad es correcta.

2. Realizar una recepción sin reserva en una habitación que ya esté ocupada.

   Posibles salidas:

   - Recepción realizada con éxito. Lo cual significaría que el proceso está funcionando de forma errónea.
   - Recepción arroja mensaje de error, ya que la pieza está ocupada. Lo cual significaría que el proceso está funcionando de de forma correcta.

3. Realizar una reserva en un hotel, de una habitación con una determinada fecha, la cual esté desocupada para tal fecha.

   Posibles salidas:

   - Reserva realizada con éxito. Esto significa que la funcionalidad es correcta.

- Reserva arroja mensaje de error, ya que la habitación está ocupada. Esto significa que la funcionalidad es errónea.

4. Realizar una recepción sin reserva en una habitación que esté disponible.

   Posibles salidas:

   - Recepción realizada con éxito. Lo cual significaría que el proceso está funcionando de forma correcta.
   - Recepción arroja mensaje de error, ya que la pieza está ocupada. Lo cual significaría que el proceso está funcionando de de forma errónea.

5. Registrar un consumo de un cliente, en una habitación en la cual no hay nadie alojado.

   Posibles salidas:

   - Consumo registrado con éxito. Esto quiere decir que el proceso está funcionando de mala manera.
   - Consumo arroja mensaje de error, ya que no hay nadie alojado en la habitación. Esto quiere decir que el proceso está funcionando de forma correcta.

6. Registrar un consumo de un cliente, en una habitación ocupada.

   Posibles salidas:

   - Consumo registrado con éxito. Esto quiere decir que el proceso está funcionando de forma correcta.

- Consumo arroja mensaje de error, ya que no hay nadie alojado en la habitación. Esto quiere decir que el proceso está funcionando de mala manera.

7. Ingresar en el programa compras, parámetros que consideren solo el tipo de servicio, como por ejemplo, hotelero.

   Posibles salidas:

   - En compras se cargan todos los servicios asociados al tipo de servicio hotelero y todos los productos asociados a cada servicio. Esto indicaría que el proceso está funcionando de buena manera.
   - En compras se cargan todos los tipos de servicio, todos los servicios y todos los productos. Esto indicaría que el proceso está funcionando de forma errónea.
   - En compras se cargan los servicios asociados al tipo de servicio hotelero, pero no se cargan los productos. Esto quiere decir que el proceso no está trabajando de forma satisfactoria.
   - En compras no se cargan ni los servicios, ni los productos. Esto indicaría que el proceso está funcionando de mala manera.

8. Ingresar en el programa compras, parámetros que consideren el tipo de servicio hotelero y un servicio, como por ejemplo, habitación.

   Posibles salidas:

   - En compras se cargan todos los productos asociados al tipo de servicio hotelero y el servicio habitación. Esto significaría que el proceso funciona correctamente.
   - En compras se cargan todos los tipos de servicio, todos los servicios y todos los productos. Esto indicaría que el proceso está funcionando de forma errónea.
   - En compras se cargan los servicios asociados al tipo de servicio hotelero, pero no se cargan los productos. Esto quiere decir que el proceso no está trabajando de forma satisfactoria.
   - En compras no se cargan ni los servicios, ni los productos. Esto indicaría que el proceso está funcionando de mala manera.

9. Ingresar en el programa compras, parámetros que consideren el tipo de servicio hotelero, el servicio habitación, y un producto, como por ejemplo, habitación2A.

   Posibles salidas:

   - En compras se carga el producto asociado al servicio habitación y a su vez asociado al tipo de servicio hotelero. Esto indicaría que el proceso está funcionando de buena manera.

- En compras no se carga el producto asociado al servicio habitación y a su vez asociado al tipo de servicio hotelero. Esto indica que el proceso no está funcionando de buena manera.
- En compras se cargan múltiples productos. Esto significa que el proceso está funcionando de forma errónea.

10. En accesos autorizados, deseamos autorizar al usuario, con perfil ejecutivo, la posibilidad de registrar compras.

    Posibles salidas:

    - El ejecutivo tiene la posibilidad de registrar compras. Esto indica que el proceso está funcionando de buena manera.
    - El ejecutivo está privado de la posibilidad de registrar compras. Esto indicaría que el proceso está funcionando de mala manera.

Bibliografía

Controladores Lógicos, Manuel Álvarez Pulido. Tercera Edición. Marcombo Boixareu Editores. 1998.

Reutilización, Parametrización y Patrones de Diseño, José Carlo Tudela. Primera Edición. 2007.

Ingeniería de Software, Juan Carlo Granja Álvarez. Primera Edición. 1995.

Patrones de Diseño, Elementos de Software orientado a objetos reutilizables, Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides. Primera Edición. Addison Wesley. 1995.

Selecting a Development approach, Office of information Services. 2005.

Crystal Report Para Visual Studio.Net, Blanco Luis Miguel. 2003.

PostgreSQL Reference Manual, ***The PostgreSQL Global Development Group. Primera Edición. 2007.***

PostgreSQL, Korry Douglas, Susan Douglas. Segunda Edición. Editorial Apress. 2005.

Anexos

Diccionario de Datos (DFD)

Nombre Flujo	**1. Libro de Compras**
Origen	Administrar Empresa Turismo
Tipo Origen	Proceso
Destino	Servicio de Impuestos Internos
Tipo Destino	Entidad
Descripción	Libro de compras que entrega toda empresa que se dedica a comprar insumos a proveedores para realizar su negocio. Exigidos por ley, los cuales se entregan al Servicio de Impuestos Internos.

Nombre Flujo	**2. Libro de Ventas**
Origen	Administrar Empresa Turismo
Tipo Origen	Proceso
Destino	Servicio de Impuestos Internos
Tipo Destino	Entidad
Descripción	Libro de ventas que entrega toda empresa que se dedica a realizar arriendos o ventas. Exigidos por ley, los cuales se entregan al Servicio de Impuestos Internos.

Nombre Flujo	3. Comprobante de Pago
Origen	Servicio de Impuestos Internos
Tipo Origen	Entidad
Destino	Administrar Empresa Turismo
Tipo Destino	Proceso
Descripción	Comprobante de Pago que entrega el Servicio de Impuestos Internos al momento de realizar los pagos correspondientes por ley.

Nombre Flujo	4. Petición Producto a Reservar
Origen	Clientes
Tipo Origen	Entidad
Destino	Administrar Empresa Turismo
Tipo Destino	Proceso
Descripción	El cliente informa que desea reservar un producto a disposición.

Nombre Flujo	5. Resolución Petición Producto a Reservar
Origen	Administrar Empresa Turismo
Tipo Origen	Proceso
Destino	Clientes
Tipo Destino	Entidad
Descripción	Se le informa al cliente la resolución de su petición.

Nombre Flujo	6. Solicitud de Venta
Origen	Clientes Administrar
Tipo Origen	Entidad
Destino	Empresa Turismo
Tipo Destino	Proceso
Descripción	El Cliente pide comprar un producto, el cual será vendido por la empresa.

Nombre Flujo	7. Producto Vendido
Origen	Administrar Empresa Turismo
Tipo Origen	Proceso
Destino	Clientes
Tipo Destino	Entidad
Descripción	La empresa le hace entrega al Cliente del producto señalado.

Nombre Flujo	8. Valor Total Servicio Prestado
Origen	Administrar Empresa Turismo
Tipo Origen	Proceso
Destino	Clientes
Tipo Destino	Entidad
Descripción	Se le informa al cliente la cantidad total de dinero que debe pagar después de haber realizado compras y/o arriendos.

Nombre Flujo	9. Forma de Pago
Origen	Clientes
Tipo Origen	Entidad
Destino	Administrar Empresa Turismo
Tipo Destino	Proceso
Descripción	EL cliente puede pagar de múltiples formas, ya sea, tarjetas de débito, cheque, tarjetas de crédito o bien en efectivo. Si la forma de pago se realiza con cheque, este debe ser entregado a la empresa.

Nombre Flujo	10. Boleta o Factura
Origen	Administrar Empresa Turismo
Tipo Origen	Proceso
Destino	Clientes
Tipo Destino	Entidad
Descripción	La empresa entrega una boleta o factura al cliente de una venta o un arriendo efectuado.

Nombre Flujo	11.Solicitud de Cotización de Insumos
Origen	Administrar Empresa Turismo
Tipo Origen	Proceso
Destino	Proveedores
Tipo Destino	Entidad
Descripción	La empresa realiza múltiples cotizaciones a empresas externas para así seleccionar la mejor opción disponible.

Nombre Flujo	12.Resultado Cotización
Origen	Proveedores
Tipo Origen	Entidad
Destino	Administrar Empresa Turismo
Tipo Destino	Proceso
Descripción	Los proveedores envían los resultados de sus cotizaciones a la empresa.

Nombre Flujo	13.Orden de Compra
Origen	Administrar Empresa Turismo
Tipo Origen	Proceso
Destino	Proveedores
Tipo Destino	Entidad
Descripción	La empresa envía la orden de compra para anunciar los productos que desea adquirir a través de sus proveedores.

Nombre Flujo	14. Documento de Pago
Origen	Administrar Empresa Turismo
Tipo Origen	Proceso
Destino	Proveedores
Tipo Destino	Entidad
Descripción	La empresa realiza el pago de sus productos adquiridos a través de sus proveedores, mediante un documento de pago.

Nombre Flujo	15.Factura
Origen	Proveedores
Tipo Origen	Entidad
Destino	Administrar Empresa Turismo
Tipo Destino	Proceso
Descripción	Los proveedores entregan una factura a la empresa compradora al momento de realizar el pago, para así certificar la compra.

Nombre Flujo	16.Guía de Despacho
Origen	Proveedores
Tipo Origen	Entidad
Destino	Administrar Empresa Turismo
Tipo Destino	Proceso
Descripción	Los proveedores confeccionan una guía de despacho para enviar los suministros adquiridos por la empresa.

Nombre Flujo	17.Datos Cliente Nuevo
Origen	Reservar
Tipo Origen	Proceso
Destino	Cliente
Tipo Destino	Archivo
Descripción	Se registran los datos de un nuevo cliente en archivo.

Nombre Flujo	18.Datos Clientes Históricos
Origen	Cliente
Tipo Origen	Archivo
Destino	Reservar
Tipo Destino	Proceso
Descripción	Se cargan los datos de un cliente histórico para poder realizar descuentos o facilitar el proceso de reserva.

Nombre Flujo	19.Fechas
Origen	Calendario
Tipo Origen	Archivo
Destino	Reservar
Tipo Destino	Proceso
Descripción	Se cargan las fechas correspondientes al calendario para poder realizar una reserva.

Nombre Flujo	20.Datos Productos
Origen	Productos
Tipo Origen	Archivo
Destino	Reservar
Tipo Destino	Proceso
Descripción	Se carga la información correspondiente a los productos que ofrece la empresa, para poder generar una reserva.

Nombre Flujo	21.Reservas con o sin Rebajas
Origen	Reservar
Tipo Origen	Proceso
Destino	Transacción
Tipo Destino	Archivo
Descripción	La información generada en una reserva es guardada en un archivo que maneja la información de transacciones.

Nombre Flujo	22.Datos Producto a Vender
Origen	Producto
Tipo Origen	Archivo
Destino	Vender
Tipo Destino	Proceso
Descripción	Se carga la información de los productos para poder generar una venta.

Nombre Flujo	23.Datos Cliente
Origen	Cliente
Tipo Origen	Archivo
Destino	Vender
Tipo Destino	Proceso
Descripción	Se carga la información de los clientes para poder generar una venta.

Nombre Flujo	24.Nuevos Datos Clientes
Origen	Vender
Tipo Origen	Proceso
Destino	Clientes
Tipo Destino	Archivo
Descripción	Se registran los datos de los clientes que no sean históricos, para facilitar en el futuro la reserva o venta de productos.

Nombre Flujo	25.Orden de Producción
Origen	Vender
Tipo Origen	Proceso
Destino	Elaborar Producto
Tipo Destino	Proceso
Descripción	Al momento de que un cliente pide un producto para su consumo, este producto puede tener la posibilidad de ser generado en ese mismo momento, por lo que se debe dar una orden de elaboración de un producto, generalmente es ocupado cuando un cliente pide comida a la habitación.

Nombre Flujo	26.Producto Elaborado
Origen	Elaborar Producto
Tipo Origen	Proceso
Destino	Vender
Tipo Destino	Proceso
Descripción	Un producto elaborado es transferido al proceso de venta, para posteriormente registrar la información de esta.

Nombre Flujo	27.Datos Elaboración de Producto
Origen	Recetas
Tipo Origen	Archivo
Destino	Elaborar Producto
Tipo Destino	Proceso
Descripción	Se carga la información de las recetas para elaborar un producto determinado.

Nombre Flujo	28.Cantidad Requerida de Insumos
Origen	Bodega
Tipo Origen	Archivo
Destino	Elaborar Producto
Tipo Destino	Proceso
Descripción	Se carga la información de los insumos necesarios para la elaboración de un producto.

Nombre Flujo	29. Datos Insumos a Vender
Origen	Bodega
Tipo Origen	Archivo
Destino	Vender
Tipo Destino	Proceso
Descripción	Se carga la información de los insumos para poder generar una venta.

Nombre Flujo	30. Datos Insumos
Origen	Bodega
Tipo Origen	Archivo
Destino	Comprar Insumos
Tipo Destino	Proceso
Descripción	Se carga la información de los insumos necesarios a ser adquiridos por la empresa.

Nombre Flujo	31. Datos Nuevos Insumos
Origen	Comprar Insumos
Tipo Origen	Proceso
Destino	Bodega
Tipo Destino	Archivo
Descripción	Se registran los insumos adquiridos que no estaban dentro de la base de datos.

Nombre Flujo	32.Datos Servicio Prestado
Origen	Vender
Tipo Origen	Proceso
Destino	Cuenta Corriente
Tipo Destino	Archivo
Descripción	La información del servicio prestado al cliente es registrada en la cuenta corriente.

Nombre Flujo	33.Cargo Crédito
Origen	Financiar
Tipo Origen	Proceso
Destino	Cuenta Corriente
Tipo Destino	Archivo
Descripción	El carga la información de crédito de un cliente en la cuenta corriente.

Nombre Flujo	34.Saldo Deuda
Origen	Cuenta Corriente
Tipo Origen	Archivo
Destino	Financiar
Tipo Destino	Proceso
Descripción	Se carga la información de deuda de un cliente, generalmente para que el cliente realice el pago final.

Nombre Flujo	35.Ingreso
Origen	Financiar
Tipo Origen	Proceso
Destino	Caja
Tipo Destino	Archivo
Descripción	Se registra el ingreso de un cliente, generalmente se ocupa cuando el cliente arrienda una habitación.

Nombre Flujo	36.Datos Entrega realizada
Origen	Vender
Tipo Origen	Proceso
Destino	Financiar
Tipo Destino	Proceso
Descripción	Se trasladan los datos de una venta al proceso de financiar.

Nombre Flujo	37.Datos Transacción
Origen	Transacción
Tipo Origen	Archivo
Destino	Financiar
Tipo Destino	Proceso
Descripción	Se cargan los datos de la transacción para poder financiar esta.

Nombre Flujo	38.Reservas Realizadas
Origen	Transacción
Tipo Origen	Archivo
Destino	Vender
Tipo Destino	Proceso
Descripción	Se carga la información de las reservas realizadas anteriormente para poder generar la venta.

Nombre Flujo	39.Datos Cliente a Financiar
Origen	Cliente
Tipo Origen	Archivo
Destino	Financiar
Tipo Destino	Proceso
Descripción	Se cargan los datos de los clientes para que puedan financiar los movmientos que han realizado dentro de la empresa.

Nombre Flujo	40.Datos a Informar
Origen	Transacción
Tipo Origen	Archivo
Destino	Generar Informes
Tipo Destino	Proceso
Descripción	Se carga la información de las transacciones realizadas en un determinado tiempo.

Nombre Flujo	41.Aviso Sobre-Stock
Origen	Comprar Insumos
Tipo Origen	Proceso
Destino	---
Tipo Destino	---
Descripción	El proceso de compra de insumos avisa a una entidad interna (operario del sistema) cuando se están adquiriendo demasiados insumos, por lo que se podría generar un sobre-stock de insumos.

Nombre Flujo	42.Reportes Generados
Origen	Generar Informes
Tipo Origen	Proceso
Destino	---
Tipo Destino	---
Descripción	Se cargan los informes generados por el sistema, que son mostrados a una entidad interna (operario del sistema) para poder apreciar la información estadística del sistema.

Nombre Flujo	43.Datos Insumos con Saldo Igual o Inferior al Nivel de Reordenación
Origen	Comparar Saldo Nivel de Insumos v/s Niveles
Tipo Origen	Proceso
Destino	Pedir y Cotizar Insumos
Tipo Destino	Proceso
Descripción	Se transfiere la información de los niveles de stock, para poder dar aviso que los niveles de stock están a un nivel de cautela.

Nombre Flujo	44. Datos Insumos con Saldo inferior al Mínimo
Origen	Comparar Saldo Nivel de Insumos v/s Niveles
Tipo Origen	Proceso
Destino	Pedir y Cotizar Insumos
Tipo Destino	Proceso
Descripción	Se transfiere la información de los niveles de stock, para cuando faltan insumos poder dar aviso de esta situación, para posteriormente poder realizar la compra de insumos.

Nombre Flujo	45. Datos Producto y Fecha Solicitados
Origen	Seleccionar Producto y Fecha a Reservar
Tipo Origen	Proceso
Destino	Verificar Disponibilidad y Decidir
Tipo Destino	Proceso
Descripción	Se transfieren los datos del producto y fecha para poder decidir si la reserva puede ser exitosa o no.

Diccionario de Datos MER

1.- Accesos Autorizados.

Descripción: relaciona los programas que podrán acceder los diferentes perfiles de usuario.

Atributos:

Atributo	Tipo de Dato	Descripción
id_acceso	Serial	Clave primaria, autoincremental.
acceso_bloqueado	Integer	Indica los programas que pueden ser accedidos por un perfil de usuario.

Llaves Foráneas:

Atributo	Tipo de Dato	Descripción
id_programa	Integer	Clave primaria de la relación “Programa”
codigo_perfil_usuario	Integer	Clave primaria de la relación “Perfil de Usuario”

2.- Bodega.

Descripción: registra las bodegas donde se almacenarán los distintos productos.

Atributos:

Atributo	Tipo de Dato	Descripción
codigo_bodega	Serial	Clave primaria, autoincremental.
nombre_bodega	Character varying(20)	Nombre de la bodega.

3.- Cabecera Transacción.

Descripción: registra los datos generales de una transacción.

Atributos:

Atributo	Tipo de Dato	Descripción
numero_cabecera	Serial	Clave primaria, autoincremental.
glosa_cabecera	Character varying(50),	Breve descripción de la cabecera.
rut_empresa_cabecera	Character varying(20)	Rut de la empresa donde se genera la transacción.
rut_cliente_cabecera	Character varying(20)	Rut cliente.involucrado en la transacción.
nombre_completo_cliente	Character varying(50)	Nombre cliente involucrado en la transacción.
rut_empleado_cabecera	Character varying(20)	Rut empleado involucrado en la transacción.

nombre_completo_empleado	Character varying(50)	Nombre empleado involucrado en la transacción.
rut_proveedor_cabecera	Character varying(20)	Rut proveedor involucrado en la transacción.
nombre_completo_proveedor	Character varying(50)	Nombre proveedor involucrado en la transacción.
nombre_tipo_documento	Character varying(30)	Tipo documento generado en la transacción.
numero_documento	Bigint	Número del documento generado en la transacción.
fecha_documento	Date	Fecha del documento generado en la transacción.

nombre_forma_pago_cabecera	**Character varying(30)**	**Forma de pago en que se lleva a cabo la transacción.**
estado_cabecera	Integer	Utilizado para la eliminación lógica de una cabecera de transacción. 1: Activo; 0: Inactivo.

Llaves Foráneas:

Atributo	Tipo de Dato	Descripción
id_proveedor	Integer	Clave primaria de la relación “Proveedor”.
codigo_forma_pago	Integer	Clave primaria de la relación “Forma de Pago”.
id_documento	Integer	Clave primaria de la relación “Tipo de Documento”.
codigo_cliente	Integer	Clave primaria de la relación “Cliente”.
id_empresa	Integer	Clave primaria de la relación “Empresa”.
id_empleado	Integer	Clave primaria de la relación “Empleado”.

4.- Caja.

Descripción: registra el flujo de dinero generado por un determinado tipo de movimiento.

Atributos:

Atributo	Tipo de Dato	Descripción
codigo_flujo_caja	Serial	Clave primaria, autoincremental.
fecha_flujo_caja	Date	Fecha en que se lleva a cabo el movimiento de caja.
hora_flujo_caja	Time	Hora en que se lleva a cabo el movimiento de caja.
monto_caja	BigInt	Monto total a registrar.
glosa_caja	**Character varying(50)**	**Detalle del flujo de caja generado.**
estado_caja	Integer	Utilizado para la eliminación lógica de un movimiento de caja. 1: Activo; 0: Inactivo.

Llaves Foráneas:

Atributo	Tipo de Dato	Descripción
codigo_movimiento	Integer	Clave primaria de la relación “Tipo_Movimiento”
id_empleado	Integer	Clave primaria de la relación “Empleado”.

5.- Cargo.

Descripción: registra los diferentes cargos de empleado.

Atributos:

Atributo	Tipo de Dato	Descripción
codigo_cargo	Serial	Clave primaria, autoincremental.
nombre_cargo	Character varying(30)	Nombre del cargo empleado.

6.- Cliente.

Descripción: registra la información de los clientes manejados por el sistema.

Atributos:

Atributo	Tipo de Dato	Descripción
codigo_cliente	Serial	Clave primaria, autoincremental.
rut_cliente	Character varying(20)	Rut único cliente.
nombre_cliente	Character varying(50)	Nombre cliente.
apellido_cliente	Character varying(50)	Apellido cliente.
direccion_cliente	Character varying(30)	Dirección cliente.

comuna_cliente	**Character varying(30)**	**Comuna cliente.**
ciudad_cliente	Character varying(30)	Ciudad cliente.
pais_cliente	Character varying(30)	País cliente.
telefono_cliente	Character varying(30)	Teléfono cliente (fijo o celular).
pasaporte_cliente	Character varying(30)	Pasaporte cliente.
email_cliente	Character varying(30)	Dirección email cliente.
nacionalidad_cliente	Character varying(30)	Nacionalidad cliente.
giro_cliente	Character varying(50)	Giro cliente.
fecha_registro_cliente	Date	Fecha en que el cliente fue registrado.
autorizado_cuenta_corriente	Integer	1: Autorizado; 2: No autorizado

estado_cliente	Integer	Utilizado para la eliminación lógica. 1: Activo; 0: Inactivo.

Llaves Foráneas:

Atributo	Tipo de Dato	Descripción
codigo_tipo_cliente	Integer	Clave primaria de la relación "Tipo_Cliente"

7.- Cuenta Corriente.

Descripción: registra el flujo de dinero generado por un cliente autorizado para usar cuenta corriente.

Atributos:

Atributo	Tipo de Dato	Descripción
codigo_cuenta_corriente	Serial	Clave primaria, autoincremental.
codigo_producto_abono	Integer	Codigo del producto al cual se abona (cargado directamente de la tabla "Productos").
fecha_mov_cta_cte	Date	Fecha en que se lleva a cabo el movimiento de cuenta corriente.
hora_mov_cta_cte	Time	Hora en que se lleva a cabo el movimiento de cuenta corriente.
codigo_mov_cta_cte	Integer	Codigo del movimiento generado (cargado

		directamente de la tabla “Tipo_Movimiento”).
monto_mov_cta_cte	BigInt	Monto total generado.
glosa_mov_cta_cte	Character varying(50)	Detalle del movimiento de cuenta corriente generado.
estado_cuenta_corriente	Integer	Utilizado para la eliminación lógica. 1: Activo; 0: Inactivo.

Llaves Foráneas:

Atributo	Tipo de Dato	Descripción
codigo_forma_pago	Integer	Clave primaria de la relación “Forma_Pago”
codigo_cliente	Integer	Clave primaria de la relación “Cliente”.

8.- Detalle Transacción.

Descripción: registra el detalle de las transacciones sobre productos, generadas por el sistema (Ej: Reservas, Anulaciones, Compras, Ventas, etc.).

Atributos:

Atributo	Tipo de Dato	Descripción
id_detalle_transaccion	Serial	Clave primaria, autoincremental.

id_tipo_ser_transaccion	Integer	Id del tipo de servicio al que está asociado el producto transado.
id_ser_transaccion	Integer	Id del servicio al que está asociado el producto transado.
cod_unidad_medida	Integer	Código de la unidad de medida del producto transado.
cantidad	Double	Cantidad a transar del producto.
precio_unitario	Double	Precio de venta unitario del producto transado.
costo_unitario	Double	Costo unitario del producto transado.
porcentaje_rebaja	BigInt	Porcentaje de rebaja que se quiere aplicar sobre el producto transado.
valor_rebaja bigint	BigInt	Monto de rebaja que se quiere aplicar sobre el producto transado.

valor_sin_rebaja	BigInt	Valor del producto sin rebaja.
valor_total_transaccion	Double	Cantidad * Precio Unitario
cod_bodega	**Integer**	**Codigo de la bodega en la que se rebaja el producto.**
fecha_inicio_transaccion date,	Date	Fecha en que se inicia la transacción.
hora_inicio_transaccion time without time zone,	Time	Hora en que se inicia la transacción.
fecha_termino_transaccion	Date	Fecha en que se termina la transacción.
hora_termino_transaccion	Time	Hora en que se termina la transacción.
fecha_inicio	Date	Fecha de inicio de la reserva.
hora_inicio	Time	Hora de inicio de la reserva.

fecha_termino	Date	Fecha de termino de la reserva.
hora_termino	Time	Hora de término de la reserva.
numero_personas	BigInt	Número de personas implicadas en la reserva.
cantidad_adultos	Integer	Cantidad de adultos implicados en la reserva.
cantidad_ninos	Integer	Cantidad de niños implicados en la reserva.
estado_transaccion	Integer	Utilizado para la eliminación lógica. 1: Activo; 0: Inactivo.

Llaves Foráneas:

Atributo	Tipo de Dato	Descripción
id_tipo_transaccion	Integer	Clave primaria de la relación “Tipo_Transaccion”
numero_cabecera	Integer	Clave primaria de la relación “Cabecera_Transacción”.
codigo_producto	Integer	Clave primaria de la relación “Productos”.

9.- Empleado.

Descripción: registra la información de los empleados manejados por el sistema.

Atributos:

Atributo	Tipo de Dato	Descripción
id_empleado	Serial	Clave primaria, autoincremental.
rut_empleado	Character varying(20)	Rut único empleado.
nombre_empleado	Character varying(50)	Nombre empleado.
apellido_empleado	Character varying(50)	Apellido empleado.
direccion_empleado	Character varying(30)	Dirección empleado.
comuna_empleado	Character varying(30)	Comuna empleado.
telefono_empleado	Character varying(30)	Teléfono empleado.
email_empleado	Character varying(30)	Email empleado.
fecha_ingreso_empleado	Date	Fecha registro empleado.

sueldo_bruto_empleado	BigInt	Sueldo bruto del empleado.
estado_empleado	Integer	Utilizado para la eliminación lógica. 1: Activo; 0: Inactivo.

Llaves Foráneas:

Atributo	Tipo de Dato	Descripción
codigo_perfil_usuario	Integer	Clave primaria de la relación "Perfil_Usuario"
codigo_tipo_empleado	Integer	Clave primaria de la relación "Tipo_Empleado".
codigo_cargo	Integer	Clave primaria de la relación "Cargos".

10.- Empresa.

Descripción: registra la información de la empresa en que funciona el sistema.

Atributos:

Atributo	Tipo de Dato	Descripción
id_empresa	Serial	Clave primaria, autoincremental.
rut_empresa	Character varying(20)	Rut único empresa.

nombre_empresa	Character varying(30)	Nombre empresa.
giro_empresa	Character varying(30)	Giro empresa.
telefono_empresa	Character varying(30)	Telefono empresa.
fax_empresa	Character varying(30)	Fax Empresa.
email_empresa	Character varying(30)	Email empresa.
direccion_empresa	Character varying(30)	Dirección empresa
comuna_empresa	Character varying(30)	Comuna empresa.
region_empresa	Character varying(30)	Región empresa.
ciudad_empresa	Character varying(30)	Ciudad empresa.
pais_empresa	Character varying(20)	País empresa.
codigo_postal_empresa	Character varying(30)	Código postal empresa.
rut_rep_legal	Character varying(20)	Rut representante legal empresa.

nombre_rep_legal	Character varying(50)	Nombre representante legal empresa.
direccion_rep_legal	Character varying(30)	Dirección representante legal empresa.
cargo_rep_legal	Character varying(30)	Cargo representante legal empresa.

condicion_pago	**Character varying(30)**	**Condición de pago empresa.**
desc_factura	Character varying(50)	Descripción de factura empresa.
nombre_sistema	Character varying(30)	Nombre del sistema.
porcentaje_iva	Integer	Porcentaje de iva seguido por la empresa.

11.- Estado Producto.

Descripción: registra los diferentes estados por los que puede pasar un producto (Ej: Disponible, Agotado, Activo, etc.).

Atributos:

Atributo	Tipo de Dato	Descripción
codigo_estado_producto	Serial	Clave primaria, autoincremental.
nombre_estado_producto	Character varying(20)	Nombre del estado producto.

12.- Forma de Pago.

Descripción: registra las formas de pago que se registrarán en alguna transacción o cuenta corriente.

Atributos:

Atributo	Tipo de Dato	Descripción
codigo_forma_pago	Serial	Clave primaria, autoincremental.
nombre_forma_pago	Character varying(30)	Nombre de la forma de pago.

13.- Parámetros.

Descripción: registra los parámetros de cada programa, es decir, los tipos de servicio, servicios y productos asociados al programa.

Atributos:

Atributo	Tipo de Dato	Descripción
id_parametro	Serial	Clave primaria, autoincremental.
cod_tipo_ser_parametro	Integer	Código del tipo de servicio asociado a un producto.
cod_ser_parametro	Integer	Código del servicio asociado a un producto.
cod_prod_parametro	Integer	Código del producto.

estado_parametro	Integer	Utilizado para la eliminación lógica. 1: Activo; 0: Inactivo.

Llaves Foráneas:

Atributo	Tipo de Dato	Descripción
id_programa	Integer	Clave primaria de la relación “Programas”

14.- Perfiles de Usuario.

Descripción: registra los perfiles de cada usuario del sistema.

Atributos:

Atributo	Tipo de Dato	Descripción
codigo_perfil_usuario	Serial	Clave primaria, autoincremental.
nombre_perfil_usuario	Character varying(30),	Nombre del perfil de usuario.
contrasena_perfil_usuario	Character varying(30),	Contraseña usuario.

15.- Producto.

Descripción: registra la información los productos que maneja el sistema.

Atributos:

Atributo	Tipo de Dato	Descripción
codigo_producto	Serial	Clave primaria, autoincremental.
codigo_producto_empresa	Integer	Código del producto asignado por la empresa.
codigo_tipo_ser	Integer	Código del tipo de servicio asociado a un producto.
nombre_producto	Character varying(30)	Nombre del producto.
descripcion_producto	Character varying(30)	Breve descripción del producto.
activacion_niveles	Integer	Indica si se activan los niveles de stock, y se da aviso al usuario. 1: Si; 0: No
stock_maximo	Double	Stock máximo que puede tener un producto.
stock_minimo	Double	Stock mínimo que puede tener un producto.
nivel_reordenacion	Double	Nivel intermedio entre el stock mínimo y máximo.

saldo_actual	Double	Saldo total del producto (Suma de todas las bodegas).
precio_costo	Double ,	Precio costo del producto.
precio_promedio	Double	Precio promedio del producto, en base al costo actual y anterior, además de las cantidades anteriores y actuales.
precio_venta	Double	Precio de venta producto.
precio_acuerdo	Double	Precio acuerdo producto

contenedor	**Integer**	**Indica si el producto contiene o no otros ingredientes.** **1: Si; 0: No.**
es_ingrediente	Integer	Indica si el producto es o no ingrediente 1: Si; 0: No.
impuesto_adicional	Integer	Indica el porcentaje de impuesto adicional sobre un producto.
sobre_neto	Integer	Indica si el impuesto adicional se calcula sobre el neto.

		1: Si; 0: No
estado_producto	Integer	Utilizado para la eliminación lógica. 1: Activo; 0: Inactivo.

Llaves Foráneas:

Atributo	Tipo de Dato	Descripción
codigo_unidad_medida	Integer	Clave primaria de la relación “Unidad Medida”
codigo_estado_producto	Integer	Clave primaria de la relación “Estado Producto”
codigo_servicio	Integer	Clave primaria de la relación “Servicios”

16.- Programas.

Descripción: registra todos los programas que gestiona el sistema.

Atributos:

Atributo	Tipo de Dato	Descripción
id_programa	Serial	Clave primaria, autoincremental.
codigo_transaccion	Character varying(10)	Código de la transacción (programa) definida por el administrador.
nombre_aplicacion	Character varying(50)	Nombre del programa.

nombre_menu	Character varying(50)	Nombre del programa en el menú del sistema.

17.- Proveedor.

Descripción: registra los proveedores que maneja el sistema.

Atributos:

Atributo	Tipo de Dato	Descripción
id_proveedor	Serial	Clave primaria, autoincremental.
rut_proveedor	Character varying(13)	Rut único proveedor.
nombre_proveedor	Character varying(50)	Nombre proveedor.
apellido_proveedor	Character varying(50)	Apellido proveedor.
direccion_proveedor	Character varying(30)	Dirección proveedor.
comuna_proveedor	Character varying(30)	Comuna proveedor.
ciudad_proveedor	Character varying(30)	Ciudad proveedor.
pais_proveedor	Character varying(30)	País proveedor.
telefono_proveedor	Character varying(30)	Teléfono proveedor

email_proveedor	**Character varying(30)**	**Email proveedor.**
nacionalidad_proveedor	Character varying(30)	Nacionalidad proveedor.
giro_proveedor	Character varying(30)	Giro proveedor.
fecha_registro_proveedor	Date	Fecha de registro del proveedor en el sistema.
estado_proveedor	Integer	Utilizado para la eliminación lógica. 1: Activo; 0: Inactivo.

18.- Receta.

Descripción: registra los ingredientes necesarios en la fabricación de un producto.

Atributos:

Atributo	Tipo de Dato	Descripción
codigo_receta	Serial	Clave primaria, autoincremental.
codigo_producto_ingrediente	Integer	Código del producto definido como ingrediente.

cantidad_ingrediente	Double	Cantidad necesaria del ingrediente.
unidad_medida_ingrediente	Character varying(30)	Nombre de la unidad de medida del ingrediente.

Llaves Foráneas:

Atributo	Tipo de Dato	Descripción
codigo_producto	Integer	Clave primaria de la relación “Productos”.

19.- Saldo Materiales.

Descripción: Detalla el saldo y la ubicación de un producto en bodega.

Atributos:

Atributo	Tipo de Dato	Descripción
id_saldo_material	Serial	Clave primaria, autoincremental.
pasillo_bodega	BigInt	Pasillo de la bodega en que se ubica el producto.
fila_bodega	BigInt	Fila de la bodega en que se ubica el producto.
columna_bodega	Character varying(30)	Columna de la bodega en que se ubica el producto.
saldo_material	Double	Saldo en bodega del producto.

Llaves Foráneas:

Atributo	Tipo de Dato	Descripción
codigo_producto	Integer	Clave primaria de la relación "Producto".
codigo_bodega	Integer	Clave primaria de la relación "Bodega".

20.- Servicio.

Descripción: Registra los servicios que gestiona el sistema, asociados a un tipo de servicio.

Atributos:

Atributo	Tipo de Dato	Descripción
codigo_servicio	Serial	Clave primaria, autoincremental.
nombre_servicio	Character varying(30)	Nombre del servicio ofrecido.
codigo_bodega	Integer	Código de la bodega asociada a un servicio.

Llaves Foráneas:

Atributo	Tipo de Dato	Descripción
codigo_tipo_servicio	Integer	Clave primaria de la relación "Tipo Servicio".

21.- Tipo Cliente.

Descripción: Registra los tipos de cliente manejados por el sistema.

Atributos:

Atributo	Tipo de Dato	Descripción
codigo_tipo_cliente	Serial	Clave primaria, autoincremental.
nombre_tipo_cliente	Character varying(30)	Nombre del tipo de cliente.

22.- Tipo Documento.

Descripción: Registra los tipos de documento manejados por el sistema (Ej: Boleta, Factura, Orden de Compra, etc.).

Atributos:

Atributo	Tipo de Dato	Descripción
id_documento	Serial	Clave primaria, autoincremental.
nombre_documento	Character varying(30)	Nombre del tipo de documento.

23.- Tipo Empleado.

Descripción: Registra los tipos de empleado manejados por el sistema (Ej: Partime, Tiempo Completo, etc.).

Atributos:

Atributo	Tipo de Dato	Descripción
codigo_tipo_empleado	Serial	Clave primaria, autoincremental.
nombre_tipo_empleado	Character varying(50)	Nombre del tipo de empleado.

24.- Tipo Movimiento.

Descripción: Registra los tipos de movimiento manejados por el sistema (Ej: Efectivo, Crédito, etc.).

Atributos:

Atributo	Tipo de Dato	Descripción
codigo_movimiento	Serial	Clave primaria, autoincremental.
nombre_movimiento	Character varying(30)	Nombre del movimiento.
descripcion_movimiento	Character varying(50)	Descripción del movimiento.

25.- Tipo Servicio.

Descripción: Registra los tipos de servicio que provee el sistema (Ej: Hotelero, Stock, etc.).

Atributos:

Atributo	Tipo de Dato	Descripción
codigo_tipo_servicio	Serial	Clave primaria, autoincremental.
nombre_tipo_servicio	Character varying(30)	Nombre del tipo de servicio.

26.- Tipo Transacción.

Descripción: Registra los tipos de transacción que se generan en el sistema (Ej: Reservas, Compras, Ventas, etc.).

Atributos:

Atributo	Tipo de Dato	Descripción
id_tipo_transaccion	Serial	Clave primaria, autoincremental.
codigo_tipo_transaccion	Character varying(20)	Código del tipo de transacción generado por el administrador.
nombre_tipo_transaccion	Character varying(50)	Nombre del tipo de transacción.

27.- Unidad Medida.

Descripción: Registra las unidades asociadas a los productos (Ej: Diario, Mensual, Kilogramos, Gramos, etc.).

Atributos:

Atributo	Tipo de Dato	Descripción
codigo_unidad_medida	Serial	Clave primaria, autoincremental.
nombre_unidad_medida	Character varying(30)	Nombre de la unidad de medida.
descripcion_unidad_medida	Character varying(30)	Descripción de la unidad de medida.

28.- Unidad Medida Convertida.

Descripción: Registra las equivalencias entre las unidades de medida, y su factor de conversión (Ej: Mensual – Diario, Kilogramo – Gramo, Docena – Unidad, etc.).

Atributos:

Atributo	Tipo de Dato	Descripción
id_conversion_um	Serial	Clave primaria, autoincremental.
um_equivalente	Integer	Código de la unidad de medida equivalente.

factor_conversion	Double	Factor para convertir la unidad de medida.

Llaves Foráneas:

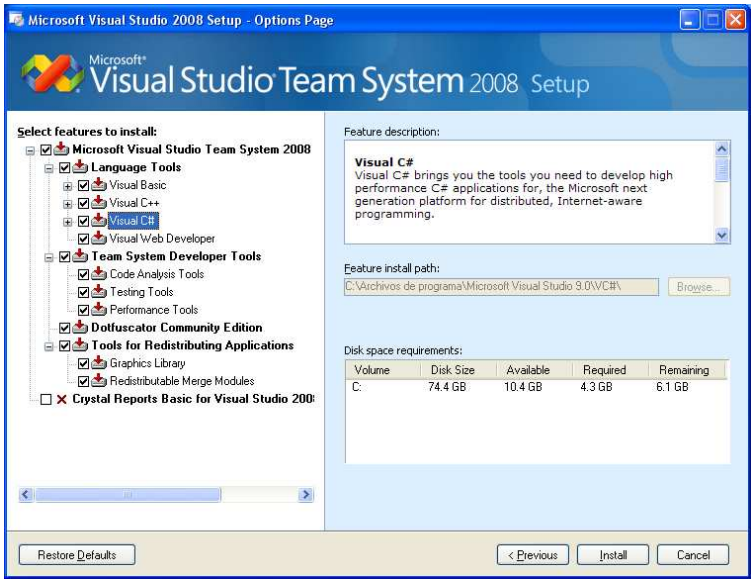
Atributo	Tipo de Dato	Descripción
codigo_unidad_medida	Integer	Clave primaria de la relación "Unidad Medida".

Guía de Instalación Visual Basic 2008

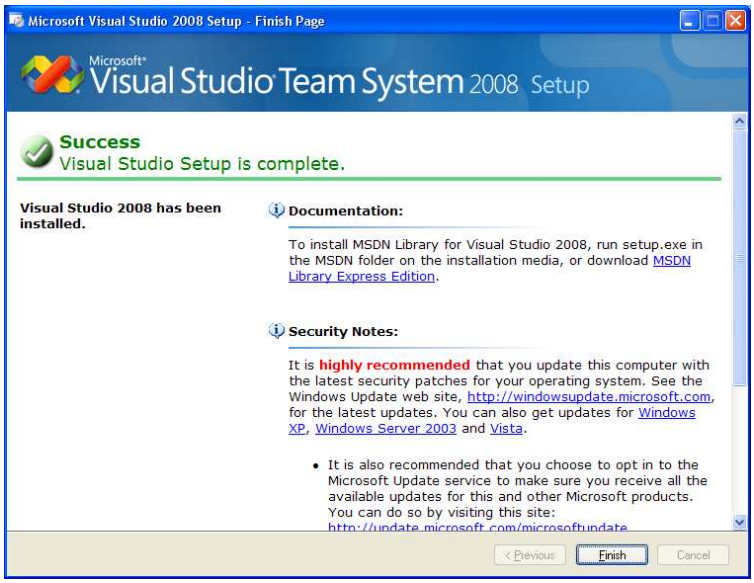
A continuación se detallan los pasos a seguir para instalar de forma correcta el Visual Studio 2008.

1. Al momento de introducir el cd/dvd de instalación de Visual Studio 2008, aparecerá la siguiente imagen. Para iniciar la instalación pulsar en "Next".

2. A continuación marcamos la opción de “Crystal Report Basic for Visual Studio 2008” junto a todas las demás y pulsamos “Install”:

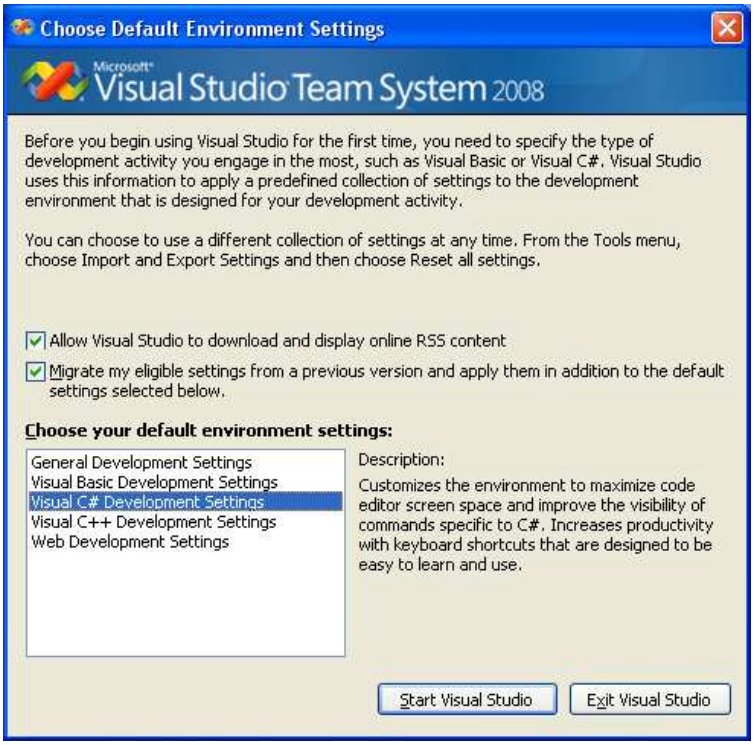


3. Posteriormente, debe aparecer la siguiente imagen, la cual menciona que el proceso de instalación finalizó correctamente:



4. Presionamos “Finish” y el proceso de instalación nos pedirá reiniciar el sistema. Pulsamos “Restart Now”. El ordenador reiniciará el sistema.

5. Luego de reiniciar, abrimos Visual Studio 2008, en donde aparecerá la siguiente imagen:



6. Marcamos la opción “General Development Settings” y presionamos en “Start Visual Studio” para finalizar completamente el proceso de instalación.

Documentación Casos de Uso

Actores

Actor	**Usuario Interno**
Caso de Uso	- Entidad de Negocio - Recepción - Reservas - Consumo Clientes - Movimiento Caja - Informes - Compras - Recetas - Autentificación
Tipo	Primario
Descripción	Este actor es el que interactúa diariamente con el sistema. Realiza múltiples tareas que se describen a continuación: - Opera datos que corresponden a la información de la empresa, los clientes, los empleados, los proveedores, los productos y

	servicios. - Realiza la recepción de clientes. - Maneja las reservas de productos. - Registra los consumos que realizan los clientes. - Efectúa el movimiento en caja que se va generando a lo largo del día. - Tiene la posibilidad de observar los informes (reportes) generados por el sistema. - Genera o da aviso de las compras que se deben realizar en la empresa. - Registra recetas de productos incluidos en la empresa. - Realiza una autentificación para poder ingresar al sistema.

Actor	**Administrador Técnico**
Caso de Uso	- Autentificación - Parámetros Sistema - Perfiles de Usuario
Tipo	Primario
Descripción	- Realiza una autentificación para poder ingresar al sistema y obtener los permisos de administrador. - Determina la parametrización que

	se ocupa dentro del sistema. - Tiene la posibilidad de crear, editar y eliminar cuentas de usuario dentro del sistema y dar los privilegios correspondientes.

Casos de Uso

Caso de Uso	**Entidad de Negocio**
Actores	Usuario interno
Propósito	Darle al usuario la posibilidad de manejar la información relevante a la empresa, de clientes, empleados, proveedores, productos y servicios.
Precondiciones	Ninguna

Caso de Uso	**Recepción**
Actores	Usuario interno
Propósito	Darle al usuario la posibilidad de realizar la recepción de clientes.
Precondiciones	Puede haberse registrado previamente una reserva para esa recepción.

Caso de Uso	**Reservas**
Actores	Usuario interno
Propósito	Darle al usuario la posibilidad de registrar o anular reservas.
Precondiciones	Ninguna

Caso de Uso	**Consumo Clientes**
Actores	Usuario interno
Propósito	Darle al usuario la posibilidad de registrar los consumos que realizan los clientes dentro de la empresa.
Precondiciones	La bodega debe contener elementos para poder ser pedidos por el usuario y el producto a ser pedido debe estar incluido en la parametrización del sistema.

Caso de Uso	**Bodega**
Actores	Ninguno
Propósito	Darle al caso de uso “Consumo clientes” información relevante a si el producto se encuentra o no en bodega.
Precondiciones	Ninguna

Caso de Uso	**Movimiento Caja**
Actores	Usuario interno
Propósito	Darle al usuario la posibilidad de manejar la información relevante al movimiento diario de caja que se realiza dentro de la empresa
Precondiciones	Debe haber un previo consumo de clientes para que se puedan generar movimientos en caja.

Caso de Uso	**Informes**
Actores	Usuario interno
Propósito	Darle al usuario la posibilidad de generar reportes estadísticos de hechos ocurridos dentro del sistema (base de datos).
Precondiciones	Debe haber un cierto movimiento en caja o haber realizado compras.

Caso de Uso	**Compras**
Actores	Usuario interno
Propósito	Darle al usuario la posibilidad de manejar la información relevante a las compras que se deben realizar para poseer un cierto nivel de stock de los productos.
Precondiciones	Debe haber un cierto stock dentro de la empresa.

Caso de Uso	**Stock**
Actores	Ninguno
Propósito	Darle al caso de uso “Compras” información relevante a los niveles de stock de la empresa, ya que si el stock es bajo debe dar aviso de compra.
Precondiciones	Ninguna

Caso de Uso	**Recetas**
Actores	Usuario interno
Propósito	Darle al usuario la posibilidad de manejar la información relevante a recetas de productos que se pueden confeccionar dentro de la misma empresa.
Precondiciones	Debe haber un cierto stock dentro de la empresa y el producto debe pertenecer a la parametrización del sistema.

Caso de Uso	**Autentificación**
Actores	Usuario interno, Administrador Técnico
Propósito	Darle al usuario y administrador la posibilidad de ingresar al sistema.
Precondiciones	Ninguna

Caso de Uso	**Parámetros Sistema**
Actores	Administrador Técnico
Propósito	Darle al administrador la posibilidad de manejar aquellos campos los cuales pueden o no ser incluidos dentro del sistema.
Precondiciones	Ninguna

Caso de Uso	**Perfiles de Usuario**
Actores	Administrador Técnico
Propósito	Darle al administrador la posibilidad de manejar las cuentas de usuario y dar ciertos privilegios.
Precondiciones	Ninguna