ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE TECNOLOGÍA

DISEÑO E IMPLEMENTACIÓN DEL SISTEMA DE FACTURACIÓN Y VENTAS PARA EL RESTAURANTE METRO CAFE

PROYECTO PREVIO A LA OBTENCIÓN DEL TÍTULO DE TECNÓLOGO EN ANÁLISIS DE SISTEMAS INFORMÁTICOS

RAÚL DAVID GUIJARRO GARCIA
OSMANY FABIAN AGUILAR ROSILLO

DIRECTOR: ING. EDGAR CHICAISA

Quito, Marzo 2006

DECLARACIÓN

Nosotros, Raúl David Guijarro García, y Osmani Fabian Aguilar, declaramos bajo juramento que el trabajo aquí descrito es de nuestra autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que hemos consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedemos nuestros derechos de propiedad intelectual correspondiente a este trabajo, a la Escuela Politécnica Nacional, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.

Raúl David Guijarro García.

Osmani Fabian Aguilar R.

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por los señores OSMANY FABIAN AGUILAR ROSILLO Y RAUL DAVID GUIJARRO GARCIA, bajo mi supervisión.

Ing. EDGAR CHICAIZA
DIRECTOR DE PROYECTO

AGRADECIMIENTO

Me parece un tópico agradecer a quienes me
Apoyaron para culminar mi tesis,
pero no puedo pasar por alto el sacrificio de
mis padres que me dieron mucha fuerza
para que termine mi carrera.

Gracias de verdad.

Dedicado a Fabiana… mi hija, donde quiera que esté.

Osmany Fabian

AGRADECIMIENTO

A mis padres, que me han apoyado desde muy lejos,
Con sus consejos para poder lograr mis objetivos.

A mis hermanos, que de una u otra forma han aportado
Para que yo pueda terminar mis estudios universitarios

Raúl David

DEDICATORIA

A mis padres, por el inmenso sacrificio
que realizaron, para hoy poder verme
cumplir una de mis metas.

A mis hermanos en especial a DINA por el apoyo incondicional
brindado en los momentos que mas los necesite.

Y a todos los amigos, profesores, que colaboraron de una u otra
forma para que este sueño se haga realidad.

Raúl David

PRESENTACIÓN

A lo largo de algunos cursos que hemos recibido en nuestra carrera, el desarrollar sistemas ha desempeñado un papel importante en la formación de nuestra profesión, basta citar algunas materias en las cuales se han desarrollado un sin número de aplicaciones sean estas pequeñas o grandes las que han logrado despertar en nosotros un interés de seguir alcanzando conocimientos informáticos.

Este proyecto que hemos desarrollado, está basado en un análisis y diseño orientado a objetos.

El sistema abarca: Administración del personal, administración de adquisiciones, administración de ventas y facturas, administración de productos, administración de proveedores, emisión de reportes y estadísticas.
Para su desarrollo se empleó una arquitectura cliente / servidor de dos capas, con Visual Basic 6.0 como front end y SQL Server 2000 como back end.
Se trabajó con base en la interfaz estándar de Windows.

RESUMEN

En el primer capitulo se detalla los múltiples problemas que le ocasiona la administración del restaurante de forma manual, de llevar un control de empleados, generar reportes de productos, control de productos dañados. En este capitulo también planteamos los objetivos que cubriremos al desarrollar un sistema que permita resolver los problemas antes mencionados.

En el transcurso de este trabajo repasaremos conceptos que son importantes para el análisis del sistema. Recordaremos todo sobre la ingeniería de software que no es más que un enfoque sistemático del desarrollo, operación, y mantenimiento, para lograr soluciones a los problemas de desarrollo de software, es decir, lo cual permite elaborar consistentemente productos correctos y utilizables.

Hablaremos de los conceptos sobre cliente/servidor y describiremos las herramientas que usamos para la construcción de este proyecto

En el tercer capítulo hablaremos sobre la metodología utilizada en nuestro proyecto; para el análisis lo hacemos con OMT y el modelo lo hacemos con el Lenguaje Unificado de Modelado (UML), el cual es independiente del lenguaje de programación utilizado.

En el cuarto capitulo hablaremos de las conclusiones y recomendaciones que hemos logrado obtener con la realización de nuestra tesis, conclusiones acerca de todo lo que hemos utilizado para realizar nuestro proyecto.

CONTENIDO

CAPITULO I

INTRODUCCIÓN

1.1 ÁMBITO

El restaurante Metro Café inició sus actividades el 11 de octubre del 2003, está cumpliendo dos años de funcionamiento, se caracteriza por entregar un buen servicio y producto de buena calidad a sus clientes, su capacidad es para 200 personas y está ubicada en uno de los sectores más comerciales de Quito, a este restaurante lo integran un grupo de personas muy entusiasmadas en sus labores; existen tres áreas que son: cocina, servicio y personal administrativo. Este restaurante no lo cierran nunca, atiende las 24 horas, por esa razón los horarios de trabajo se dividen en una forma adecuada, para que el personal no sufra cansancio. La cocina la conforman 10 personas; servicio esta conformado por 8 personas y 3 para personal administrativo.

1.1.1 PLANTEAMIENTO DEL PROBLEMA

El restaurante Metro café es un Restaurante que no lleva un control sobre las compras a proveedores, ventas a los clientes, y los inventarios diarios, de una forma ordenada.

El control sobre las compras a proveedores se lo realiza en forma manual y desorganizada y causa la pérdida de productos.

Al momento de realizar compras, hay veces que se devuelve el producto y eso nunca se registra.

Las ventas a los clientes son bajo un sistema muy viejo y lento, lo cual produce un tráfico de empleados al momento de ingresar la ordenes, además no tiene todas las opciones del menú, esto causa disgustos a los clientes.

En ventas se realiza anulaciones y devoluciones debido a producto mal ingresado o por devoluciones del cliente; y, su registro es ineficiente.

No se lleva un control de todo el producto que está caducado, en mal estado, y mal preparado.

Los productos tienden a dañarse por no saber controlar y se los da de baja, por lo tanto se lleva un control inadecuado.

El control de la llegada y salida del personal se lo realiza de forma manual, para luego contar el número de horas de trabajo.

1.1.2 JUSTIFICACIÓN

La gran demanda que tiene el Restaurante Metro Café, en la adquisición y venta de productos, ha hecho que se tenga que llevar de una manera desorganizada los inventarios y la falta de producto es notable, por que al momento de realizar las ventas varios productos no existen, se quiere realizar una distribución adecuada de productos en bodega para que no existan sobrantes ni faltantes.

Este sistema se realizará gracias a las herramientas de programación visual que se han desarrollado con el fin de prestar servicios confiables y que responden a nuestras necesidades, esta distribución de productos, facturación, Adquisición, Ventas, Reportes, se organizará mediante la herramienta Visual Basic y el apoyo de una base de datos SQL, con la cual almacenaremos datos necesarios para esta organización, gracias a las tecnologías avanzadas de hardware, este sistema funcionará en modo cliente / servidor y se adaptará fácilmente a un computador con características básicas para su funcionamiento.

Los reportes necesarios para inventarios los sacaremos exactos y adecuados para controlar productos que faltan y sobran, para no recargarnos de productos que no salen y abastecer bien las bodegas.

El control de horas de entrada y salida se registrará cada empleado para que conste como día trabajado

La gerencia de Metro café, al ver que todo se hace de una forma incorrecta, con deficiencia en los procesos, sabiendo que estos se los puede optimizar al usar un sistema mejor estructurado y más rápido, y al ver que se lo puede realizar en un tiempo inferior a lo que se esta realizando, ahora necesita de un mejor sistema que le preste más servicios.

La facturación se la realiza por medio de un sistema viejo y pondremos en funcionamiento detalles que hagan más eficaz el proceso de ingreso de productos.

Los gerentes no han buscado un software nuevo y no se han dado cuenta hasta ahora que necesitan de un sistema completo pero al momento de ofrecerles nuestro sistema y las ventajas que promete, han decidido trabajar con nuestras propuestas.

Todo lo expuesto anteriormente justifica desarrollar un sistema computarizado para el control de producto y facturación del restaurante Metro Café, para tener una información a tiempo y justificada, para así poder tomar las decisiones más acertadas en el tiempo conveniente.

1.2 OBJETIVOS

1.2.1 OBJETIVO GENERAL

Desarrollar e implementar un sistema de facturación y ventas para ayudar en el mejoramiento del servicio y que el proceso de Administración sea más sencillo.

1.2.2 OBJETIVOS ESPECÍFICOS

- Llevar a cabo un módulo de adquisiciones con la cual controlaremos todos los productos que ingresan a bodega organizando cada uno de ellos y verificando fechas de caducidad, calidad y estado del producto y tendremos un control más específico de las compras.
- Establecer un módulo para controlar las ventas de los productos, para que al momento de ingresar estén dispuestos todos los productos y generar facturas certificadas por el SRI, para cada venta realizada.
- Desarrollar un módulo para administrar los productos en mal estado y productos que se devuelvan, para tener en stock de bodega solo productos que sirvan.
- Implantar un módulo para administrar a los proveedores haciendo pedidos del producto que se necesita y verificando en bodega el stock, para así poder realizar pedidos con anticipación.
- Desarrollar un módulo para crear los diferentes perfiles de usuarios, con lo cual se le proveerá uno de estos a los empleados, para que les permita a cada uno de ellos ingresar al sistema con su nombre de usuario, para que se registren a la hora de entrada y salida.
- Crear un módulo de generación de reportes, en lo cual se tendría diferentes clases de consultas, de acuerdo a la información que necesite el usuario, estos reportes serán diseñados a la vez para imprimir.

1.3 ALCANCE Y LIMITACIONES

1.3.1 ALCANCE

Lo que queremos alcanzar con este sistema de software es contribuir sustancialmente a un mejor desenvolvimiento del restaurante metro café, entregando un software con opciones fáciles de manejo para los empleados y así solucionando problemas de servicio, de control de producto, inventarios, y de proveedores; para con esto optimizar el tiempo de prestación de servicios a los clientes.

1.3.2 LIMITACIONES

El proyecto se encuentra limitado a los procesos de administración del personal, control de inventario, facturación , y generar reportes además este proyecto esta desarrollado para adaptarse a la forma de trabajo que lleva el restaurante metro café, si se lo quisiera instalar para otro restaurante se tendría que hacer ciertas modificaciones para que se adapte a las necesidades requeridas.

1.4 PRESUPUESTO (cuadro 1.1)

COSTO APROXIMADO PARA EL DESARROLLO DEL SISTEMA

RECURSOS HUMANOS		**VALOR**	
	Número Horas	**Valor x Horas**	**Costo Total**
Analista	1400	$ 5.00	$ 7000.00
Desarrolladores	384	$ 2.00	$ 768.00
Software 1 Licencia Rational Rose 2000			$ 450.00
1 Licencia SQL Server 2000			$ 410.00
1 Licencia Visual Basic 6.0			$ 519.00
Hardware Computadora Pentium IV Case JS ATX 80 GB disco duro 512 MB en RAM Procesador Intel 3.0 GHZ Monitor, teclado , Mouse			$ 740
Gastos Varios Hojas de papel Bonn Diskettes y Cds Cartuchos de Impresora			$ 150
Costo aproximado del proyecto			$ 6989.00

CUADRO 1.1

CAPITULO II

MARCO TEÓRICO

2.1 INGENIERÍA DEL SOFTWARE

La Ingeniería del software es una disciplina o área de la Informática o Ciencias de la Computación, que ofrece métodos y técnicas para desarrollar y mantener software de calidad que resuelven problemas de todo tipo. Hoy día, es cada vez mas frecuente la consideración de la Ingeniería del Software como una nueva área de la Ingeniería, y el Ingeniero del Software comienza a ser una profesión implantada en el mundo laboral internacional, con derechos, deberes y responsabilidades que cumplir, junto a una, ya, reconocida consideración social en el mundo empresarial y, por suerte, para esas personas con brillante futuro.

El proceso de ingeniería de software se define como "un conjunto de etapas parcialmente ordenadas con la intención de lograr un objetivo, en este caso, la obtención de un producto de software de calidad". El proceso de desarrollo de software "es aquel en que las necesidades del usuario son traducidas en requerimientos de software, estos requerimientos transformados en diseño y el diseño implementado en código, el código es probado, documentado y certificado para su uso operativo". Concretamente "define quién está haciendo qué, cuándo hacerlo y cómo alcanzar un cierto objetivo.

2.1.1 Diseño Conceptual

En esta etapa se debe construir un esquema de la información que se usa en la empresa, independientemente de cualquier consideración física. A este esquema se le denomina esquema conceptual. Al construir el esquema, los diseñadores descubren la semántica (significado) de los datos de la empresa: encuentran entidades, atributos y relaciones. El objetivo es comprender:

- La perspectiva que cada usuario tiene de los datos.
- La naturaleza de los datos, independientemente de su representación física.
- El uso de los datos a través de las áreas de aplicación.

El esquema conceptual se puede utilizar para que el diseñador transmita a la empresa lo que ha entendido sobre la información que ésta maneja. Para ello, ambas partes deben estar familiarizadas con la notación utilizada en el esquema entidad relación. [1]

2.1.2 Diseño Lógico

El diseño lógico es el proceso de construir un esquema de la información que utiliza la empresa, basándose en un modelo de base de datos específico, independiente del SGBD concreto que se vaya a utilizar y de cualquier otra consideración física. [2]

El esquema lógico es una fuente de información para el diseño físico. Además, juega un papel importante durante la etapa de mantenimiento del sistema, ya que permite que los futuros cambios que se realicen sobre los programas de aplicación o sobre los datos, se representen correctamente en la base de datos.

Tanto el diseño conceptual, como el diseño lógico, son procesos iterativos, tienen un punto de inicio y se van refinando continuamente. Ambos se deben

[1] http://www3.uji.es/~mmarques/f47/apun/node46.html
[2] http://www.itlp.edu.mx/publica/tutoriales/analisis/24.htm

ver como un proceso de aprendizaje en el que el diseñador va comprendiendo el funcionamiento de la empresa y el significado de los datos que maneja. El diseño conceptual y el diseño lógico son etapas clave para conseguir un sistema que funcione correctamente. Si el esquema no es una representación fiel de la empresa, será difícil, sino imposible, definir todas las vistas de usuario (esquemas externos), o mantener la integridad de la base de datos. También puede ser difícil definir la implementación física o el mantener unas prestaciones aceptables del sistema. Además, hay que tener en cuenta que la capacidad de ajustarse a futuros cambios es un sello que identifica a los buenos diseños de bases de datos. Por todo esto, es fundamental dedicar el tiempo y las energías necesarias para producir el mejor esquema que sea posible.

2.2 ARQUITECTURA CLIENTE SERVIDOR

El concepto de cliente / servidor proporciona una forma eficiente de utilizar todos estos recursos de máquina, de tal forma que la seguridad y fiabilidad que proporcionan los entornos mainframe se traspasa a la red de área local. A esto hay que añadir la ventaja de la potencia y simplicidad de los ordenadores personales. [3]

La arquitectura cliente / servidor es un modelo para el desarrollo de sistemas de información en el que las transacciones se dividen en procesos independientes que cooperan entre sí para intercambiar información, servicios o recursos. Se denomina cliente al proceso que inicia el diálogo o solicita los recursos y servidor al proceso que responde a las solicitudes.

En este modelo las aplicaciones se dividen de forma que el servidor contiene la parte que debe ser compartida por varios usuarios, y en el cliente permanece sólo lo particular de cada usuario.

[3] http://www.monografias.com/trabajos24/arquitectura-cliente-servidor/arquitectura-cliente-servidor.shtml#element

2.2.1 CARACTERÍSTICAS DE LA ARQUITECTURA CLIENTE SERVIDOR

En el modelo CLIENTE / SERVIDOR podemos encontrar las siguientes características:

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

2. Las funciones de Cliente y Servidor pueden estar en plataformas separadas, o en la misma plataforma.

3. Un servidor da servicio a múltiples clientes en forma concurrente.

4. 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.

5. 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.

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.

El servidor presenta a todos sus clientes una interfase única y bien definida.

El cliente no necesita conocer la lógica del servidor, sólo su interfase externa.

El cliente no depende de la ubicación física del servidor, ni del tipo de equipo físico en el que se encuentra, ni de su sistema operativo.

Los cambios en el servidor implican pocos o ningún cambio en el cliente.

2.2.2 ELEMENTOS DE LA ARQUITECTURA CLIENTE / SERVIDOR[4]

En esta aproximación, y con el objetivo de definir y delimitar el modelo de referencia de una arquitectura Cliente / servidor, debemos identificar los componentes que permitan articular dicha arquitectura, considerando que toda aplicación de un sistema de información está caracterizada por tres componentes básicos:

- Captación de Información
- Procesos
- Almacenamiento de la Información

2.3 HERRAMIENTAS DE SOPORTE.

2.3.1 SQL SERVER 2000

El Lenguaje de Consulta Estructurado (Structured Query Language) es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones sobre las mismas. incorpora características del álgebra y el cálculo relacional permitiendo lanzar consultas con el fin de recuperar información de interés de una base de datos, de una forma sencilla.

Los orígenes del SQL están ligados a los de las bases de datos relaciónales. En 1970 Codd propone el modelo relacional y asociado a éste un sublenguaje de acceso a los datos, basado en el cálculo de predicados. Basándose en estas ideas los laboratorios de IBM definen el lenguaje SEQUEL (Structured

[4] http://www.monografias.com/trabajos24/arquitectura-cliente-servidor/arquitectura-cliente-servidor.shtml#element

English QUEry Language) que más tarde sería ampliamente implementado por el SGBD experimental System R, desarrollado en 1977 también por IBM. Sin embargo, fue Oracle quien lo introdujo por primera vez en 1979 en un programa comercial.

El SEQUEL terminaría siendo el predecesor de SQL, siendo éste una versión evolucionada del primero. El SQL pasa a ser el lenguaje por excelencia de los diversos SGBD relacionales surgidos en los años siguientes y es por fin estandarizado en 1986 por el ANSI, dando lugar a la primera versión estándar de este lenguaje, el SQL-86 o SQL1. Al año siguiente este estándar es también adoptado por la ISO.

El SQL es un lenguaje de acceso a bases de datos que explota la flexibilidad y potencia de los sistemas relacionales permitiendo gran variedad de operaciones sobre los mismos.

Es un lenguaje declarativo de alto nivel o de no procedimiento, que gracias a su fuerte base teórica y su orientación al manejo de conjuntos de registros, y no a registros individuales, permite una alta productividad en codificación. De esta forma una sola sentencia puede equivaler a uno o más programas que utilizasen un lenguaje de bajo nivel orientado a registro.

2.3.2 RATIONAL ROSE 2000

La rational rose es una Lenguaje orientado a objetos Unificada que Modela el instrumento de diseño de software intencionado para el modelado visual y la construcción de aplicaciones de software de nivel de la empresa. Del modo igual a un director teatral se obstruye de un juego, un diseñador de software usa el Rose Rational para modelar lo diferentes diagramas del modelo espiral.

Dos rasgos populares de Rational rose es su capacidad de proporcionar el desarrollo iterativo y la ingeniería de ida y vuelta. El Rational rose permite a diseñadores aprovechar el desarrollo iterativo (a veces llamaban el desarrollo evolutivo) porque el uso nuevo puede ser creado por etapas con la salida de

una iteración que se hace la entrada al siguiente (próximo). Entonces, como el revelador comienza a entender cómo los componentes actúan recíprocamente y hace modificaciones en el diseño, el Rational puede realizar lo que llaman " la ingeniería de ida y vuelta".

2.3.3 VISUAL BASIC V 6.

Visual Basic es un lenguaje de programación desarrollado por Microsoft Visual Basic es un lenguaje visual que desciende del lenguaje de programación Basic. Su primera versión fue presentada en 1991 con la intención de simplificar la programación utilizando un ambiente de desarrollo completamente gráfico que facilitara la creación de interfaces gráficas y en cierta medida también la programación misma.

Es un lenguaje de fácil aprendizaje pensado tanto para programadores principiantes como expertos, guiado por eventos, y centrado en un motor de formularios poderoso que facilita el rápido desarrollo de aplicaciones gráficas. Su principal innovación, que luego, fue adoptada por otros lenguajes, fue el uso de un tipo de dll, llamado inicialmente vbx y posteriormente ocx, que permiten contener toda la funcionalidad de un control y facilitar su rápida incorporación a los formularios.

Su sintaxis, derivada del antiguo BASIC, ha sido ampliada con el tiempo al agregarse las características típicas de los lenguajes estructurados modernos. Se ha agregado una implementación limitada de la programación orientada a objetos (los propios formularios y controles son objetos), que, sin embargo, no admite ni polimorfismos ni herencia No requiere de manejo de punteros y posee un manejo muy sencillo de cadenas de caracteres. Posee varias bibliotecas para manejo de bases de datos, pudiendo conectar con cualquier base de datos a través de ODBC (Informix, DBase, Access, MySQL, SQL Server, etc) a través de ADO

Es utilizado principalmente para aplicaciones de gestión de empresas, debido a la rapidez con la que puede hacerse un programa que utilice una base de datos sencilla, además de la abundancia de programadores en este lenguaje.

2.3.4 SEGATE CRYSTAL REPORTS PROFESSIONAL 7.0

La herramienta de Windows más poderosa para elaborar informes, que usted puede obtener hoy en día. Además de ser muy fácil de usar no tiene ningún problema para trabajar con Windows xp y complementa con visual Basic 6.0 dándonos una vista previa de los reportes antes de ser impresos. Además no tiene conflictos con SQL 2000 y con la ayuda de este programa hemos conseguido mejorar la estética de nuestro sistema

2.3.5 HELP CREATOR 1.0

Con este programa podemos crear las ayudas de nuestro proyecto de una forma muy fácil, nos permite incluir imágenes en los archivos de ayuda de programa, de la misma forma que los programas de Windows lo tienen; se puede crear ayudas interactivos para así poder facilitar el funcionamiento del software a usuario final y con este tipo de ayuda henos conseguido que nuestro sistema tenga una interfaz similar a los programas de Windows.

CAPITULO III

ASPECTOS METODOLÓGICOS

3.1 PARADIGMA INCREMENTAL ESPIRAL[5]

El modelo espiral para la ingeniería de software ha sido desarrollado para cubrir las mejores características tanto del ciclo de vida clásico, como de la creación de prototipos, añadiendo al mismo tiempo un nuevo elemento: el análisis de riesgo. El modelo representado mediante la espiral de la figura 2.4, define cuatro actividades principales:

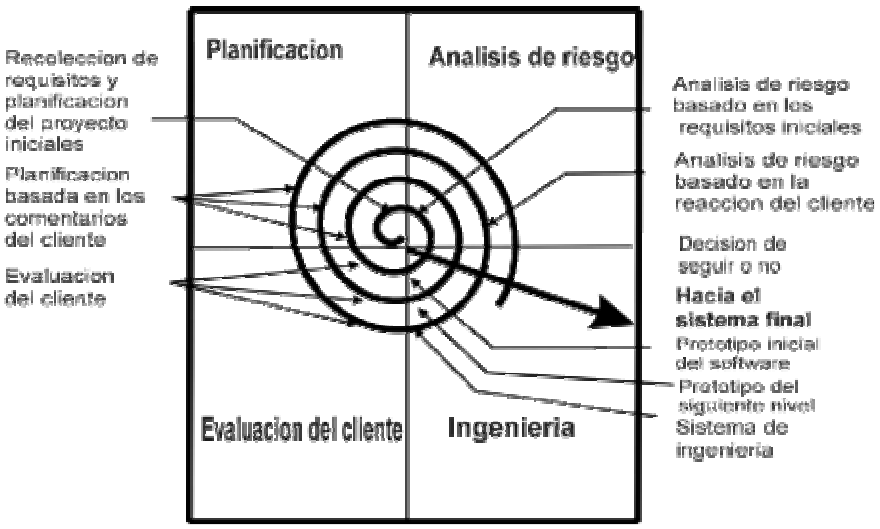


Figura 2.4

Fuente: http://www.monografias.com/trabajos6/meto/meto.shtml

Durante la primera vuelta, alrededor de la espiral se definen los objetivos, las alternativas y las restricciones, y se analizan e identifican los riesgos. Si el análisis de riesgo indica que hay una incertidumbre en los requisitos, se puede

[5] http://www.monografias.com/trabajos6/meto/meto.shtml

usar la creación de prototipos en el cuadrante de ingeniería para dar asistencia, tanto al encargado de desarrollo como al cliente.

El cliente evalúa el trabajo de ingeniería (cuadrante de evaluación de cliente) y sugiere modificaciones. Sobre la base de los comentarios del cliente se produce la siguiente fase de planificación y de análisis de riesgo. En cada bucle alrededor de la espiral, la culminación del análisis de riesgo resulta en una decisión de "seguir o no seguir".

Con cada iteración alrededor de la espiral (comenzando en el centro y siguiendo hacia el exterior), se construyen sucesivas versiones del software, cada vez más completa y, al final, al propio sistema operacional.

El paradigma del modelo en espiral para la ingeniería de software es actualmente el enfoque más realista para el desarrollo de software y de sistemas a gran escala. Utiliza un enfoque evolutivo para la ingeniería de software, permitiendo al desarrollador y al cliente entender y reaccionar a los riesgos en cada nivel evolutivo. Utiliza la creación de prototipos como un mecanismo de reducción de riesgo, pero, lo que es más importante permite a quien lo desarrolla aplicar el enfoque de creación de prototipos en cualquier etapa de la evolución de prototipos.[6]

3.1.1 Etapas del Ciclo

Planificación: determinación de objetivos, alternativas y restricciones.

Análisis de riesgo: análisis de alternativas e identificación/resolución de riesgos.

Ingeniería: desarrollo del producto del "siguiente nivel",

Evaluación del cliente: Valorización de los resultados de la ingeniería.

[6] http://www.itlp.edu.mx/publica/tutoriales/analisis/24.htm

3.2 METODOLOGÍA

La metodología hace referencia a un conjunto de reglas y procedimientos que permiten ejecutar en forma lógica una serie de pasos para obtener un resultado determinado, indica qué tareas, en qué orden, cuándo ejecutarlas, cómo ejecutarlas, cómo controlar el avance de esas tareas y cuáles son las herramientas que deben utilizar en esa tarea.

Una metodología es una especificación de los pasos a seguir durante el ciclo de vida del desarrollo de sistemas que incluye:

- Tareas paso a paso por cada fase.
- Funciones individuales y en grupo desempeñadas en cada tarea.
- Productos resultantes.

3.2.1 METODOLOGÍA OMT

La metodología OMT (Object Modeling Technique) fue creada por James Rumbaugh y Michael Blaha en 1991, mientras James dirigía un equipo de investigación de los laboratorios General Electric.

OMT es una de las metodologías de análisis y diseño orientadas a objetos, eficientes que existen en la actualidad. La gran virtud que aporta esta metodología es su carácter de abierta (no propietaria), que le permite ser de dominio público y , en consecuencia, sobrevivir con enorme vitalidad. Esto facilita su evolución para acoplarse a todas las necesidades actuales y futuras de la ingeniería de software.

La característica fundamental de OMT es tener en cuenta todos los principios del paradigma orientado a objetos y refuerza particularmente la necesidad de modelos consistentes de análisis y diseño del negocio como paso previo y fundamental al desarrollo de la programación y la definición de la base de datos.

3.2.1.1 Etapas de la metodología OMT

3.2.1.1.1 Análisis.

El analista construye un modelo del dominio del problema, mostrando sus propiedades más importantes. El modelo de análisis es una abstracción resumida y precisa de lo que debe hacer el sistema deseado y no de la forma en que se hará. Los elementos del modelo deben ser conceptos del dominio de aplicación y no conceptos informáticos, tales como estructuras de datos. Un buen modelo debe poder ser entendido y criticado por expertos en el dominio del problema que no tengan conocimientos informáticos.

3.2.1.1.2 Diseño del sistema.

El diseñador del sistema toma decisiones de alto nivel sobre la arquitectura del mismo. Durante esta fase el sistema se organiza en subsistemas basándose tanto en la estructura del análisis como en la arquitectura propuesta. Se selecciona una estrategia para afrontar el problema.

3.2.1.1.3 Diseño de objetos.

El diseñador de objetos construye un modelo de diseño basándose en el modelo de análisis, pero incorporando detalles de implementación. El diseño de objetos se centra en las estructuras de datos y algoritmos que son necesarios para implementar cada clase. OMT describe la forma en que el diseño puede ser implementado en distintos lenguajes (orientados y no orientados a objetos, bases de datos, etc.).

3.2.1.1.4 Implementación.

Las clases de objetos y relaciones desarrolladas durante el análisis de objetos se traducen finalmente a una implementación concreta. Durante la fase de implementación es importante tener en cuenta los principios de la ingeniería de software, de forma que la correspondencia con el diseño sea directa y el sistema implementado sea flexible y extensible. No tiene sentido que utilicemos AOO y DOO de forma que potenciemos la reutilización de código y la

correspondencia entre el dominio del problema y el sistema informático, si luego perdemos todas estas ventajas con una implementación de mala calidad.

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

3.2.1.1.5 Modelo de Objetos

Describe la estructura estática de los objetos del sistema (identidad, relaciones con otros objetos, atributos y operaciones). El modelo de objetos proporciona el entorno esencial en el cual se pueden situar el modelo dinámico y el modelo funcional. El objetivo es capturar aquellos conceptos del mundo real que sean importantes para la aplicación. Se representa mediante diagramas de objetos.

Esta es la parte principal de la Técnica para el modelado, ya que se fundamenta en la teoría de OO. La definición clara de las entidades que intervienen en el sistema es un paso inicial necesario para poder definir qué transformaciones ocurren en ellas y cuándo se producen estas transformaciones. Esta forma de pensar es inherente al paradigma de OO donde las clases y su jerarquía determinan el sistema.
Los diagramas de objetos permiten representar gráficamente los objetos, las clases y sus relaciones mediante dos tipos de diagramas:

- los diagramas de clases
- los diagramas de casos concretos (instancias).

Los diagramas de clases describen las clases que componen el sistema y que permitirán la creación de casos concretos, los diagramas de casos concretos describen la manera en que los objetos del sistema se relacionan y los casos concretos que existen en el sistema de cada clase. En los diagramas que componen este modelo se pueden representar los siguientes elementos del sistema: objetos y clases, atributos, operaciones, y relaciones o asociaciones.

3.2.1.1.6 Modelo Dinámico

Los aspectos del sistema que están relacionados con el tiempo y con los cambios constituyen el modelo dinámico.
Los conceptos más importantes del modelado dinámico son los sucesos, que representan estímulos externos, y los estados, que representan los valores de los objetos. El diagrama de estados va a representar los sucesos y los estados que se dan en el sistema.
El modelo de objetos describe las posibles tramas de objetos, atributos y enlaces que pueden existir en un sistema. Los valores de los atributos y de los enlaces mantenidos por un objeto son lo que se denomina su estado. A lo largo del tiempo, los objetos se estimulan unos a otros, dando lugar a una serie de cambios en sus estados.

Describe los aspectos de un sistema, los sucesos que marcan los cambios, además muestra la secuencias de sucesos, estados que definen el contexto para los sucesos, aquel aspecto de un sistema que describe las secuencias de operaciones que se producen sin tener en cuenta lo que hagan las operaciones, aquello a lo que afecten o la forma en que están implementadas. Se representa gráficamente mediante diagramas de estado.

Los pasos a seguir en el modelo dinámico son:

- Se preparan escenarios de secuencias típicas de interacción.
- Se identifican sucesos que actúen entre objetos.
- Se prepara un seguimiento de sucesos para cada escenario.
- Se construyen diagramas de estado
- Se comparan los sucesos intercambiados entre objetos para verificar la congruencia.

3.2.1.1.7 Modelo funcional[7]

Describe las transformaciones de valores de datos (funciones, correspondencias, restricciones y dependencias funcionales) que ocurren dentro del sistema. Captura lo que hace el sistema, independientemente de cuando se haga o de la forma en que se haga. Se representa mediante diagramas de flujo de datos

El modelo funcional muestra la forma en que se derivan los valores producidos en un cálculo a partir de los valores introducidos, sin tener en cuenta el orden en el cual se calculan los valores. Consta de múltiples diagramas de flujo de datos, que muestran el flujo de valores desde las entradas externas, a través de las operaciones y almacenes internos de datos hasta las salidas externas. También incluyen restricciones entre valores dentro del modelo de objetos. Los diagramas de flujo de datos no muestran el control ni tampoco información acerca de la estructura de los objetos; todo esto pertenece a los modelos dinámico y de objetos.

Los pasos del modelo funcional son:

- Identificación de los valores de entrada y de salida
- Construcción de diagramas de flujo de datos que muestren las dependencias funcionales.
- Descripción de las funciones.
- Identificación de restricciones.
- Especificación de los criterios de optimización.

[7] *RUMBAUGH, J; Modelado y Diseño Orientado a Objetos. Metodología OMT*

3.2.2. Lenguaje Unificado de Modelado UML

El Lenguaje de Modelamiento Unificado (UML - Unified Modeling Language) es un lenguaje gráfico para visualizar, especificar y documentar cada una de las partes que comprende el desarrollo de software. UML entrega una forma de modelar cosas conceptuales como lo son procesos de negocio y funciones de sistema, además de cosas concretas como lo son escribir clases en un lenguaje determinado, esquemas de base de datos y componentes de software que se pueden volver a usar.

Los diagramas de UML se pueden dividir en estáticos, dinámicos, y funcional.

3.2.2.1 Diagramas estáticos:

3.2.2.1.1 Diagramas de casos de uso

El diagrama de casos de uso representa la forma en cómo un Cliente (Actor) opera con el sistema en desarrollo, además de la forma, tipo y orden en cómo los elementos interactúan (operaciones o casos de uso).

Un diagrama de Casos de Uso muestra las distintas operaciones que se esperan de una aplicación o sistema y cómo se relaciona con su entorno (usuarios u otras aplicaciones).

Un diagrama de casos de uso consta de los siguientes elementos:

Actor

Casos de Uso

Relaciones de Uso

3.2.2.1.1 Elementos

3.2.2.1.1.1 **Actor**: (figura 3.1)

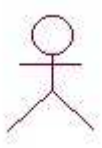


Figura 3.1

Una definición previa, es que un Actor es un rol que un usuario juega con respecto al sistema. Es importante destacar el uso de la palabra rol**,** pues con esto se especifica que un Actor no necesariamente representa a una persona en particular, sino más bien la labor que realiza frente al sistema.
Como ejemplo a la definición anterior, tenemos el caso de un sistema de ventas en que el rol de Vendedor con respecto al sistema puede ser realizado por un Vendedor o bien por el Jefe de Local.

3.2.2.1.1 .2 **Caso de Uso**: (figura 3.2)

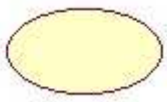


Figura 3.2

Es una operación/tarea específica que se realiza tras una orden de algún agente externo, sea desde una petición de un actor o bien desde la invocación desde otro caso de uso.

3.2.2.1.2 Relaciones:

3.2.2.1.2.1 Asociación: (figura 3.3)

Figura 3.3

Es el tipo de relación más básica que indica la invocación desde un actor o caso de uso a otra operación (caso de uso). Dicha relación se denota con una flecha simple

3.2.2.1.2 Diagramas de Clases

El Diagrama de Clases es el diagrama principal para el análisis y diseño. Un diagrama de clases presenta las clases del sistema con sus relaciones estructurales y de herencia. La definición de clase incluye definiciones para atributos y operaciones. El modelo de casos de uso aporta información para establecer las clases, objetos, atributos y operaciones. El mundo real puede ser visto desde abstracciones diferentes (subjetividad).

Un diagrama de clases sirve para visualizar las relaciones entre las clases que involucran el sistema, las cuales pueden ser asociativas, de herencia, y de uso.

Un diagrama de clases esta compuesto por los siguientes elementos:
Clase: atributos, métodos y visibilidad.
Relaciones: Herencia, Composición, Agregación, Asociación y Uso

3.2.2.2 Diagramas dinámicos:

3.2.2.1.1 Diagrama de secuencia

El Diagrama de Secuencia es más adecuado para observar la perspectiva cronológica de las interacciones, muestra la secuencia explícita de mensajes y son mejores para especificaciones de tiempo real y para escenarios complejos.

Dicho diagrama puede ser obtenido desde el diagrama de Casos de Uso.

3.2.2.1.2 Diagrama de colaboración

El Diagrama de Colaboración ofrece una mejor visión espacial mostrando los enlaces de comunicación entre objetos, muestra las relaciones entre objetos y son mejores para comprender todos los efectos que tiene un objeto y para el diseño de procedimientos. El diagrama de Colaboración puede obtenerse automáticamente a partir del correspondiente diagrama de Secuencia (o viceversa).

Es una descripción de una colección de objetos que interactúan para implementar un cierto comportamiento dentro de un contexto. Describe una sociedad de objetos cooperantes unidos para realizar un cierto propósito. Una colaboración contiene ranuras que son rellenadas por los objetos y enlaces en tiempo de ejecución. Una ranura de colaboración se llama Rol porque describe el propósito de un objeto o un enlace dentro de la colaboración.

3.2.2.3 Diagramas funcionales:

3.2.2.3.1 Diagrama de Actividades

El Diagrama de Actividad es una especialización del Diagrama de Estado, organizado respecto de las acciones y usado para especificar:
Un método
Un caso de uso

Un proceso de negocio (Workflow)

Un diagrama de actividades es provechoso para entender el comportamiento de alto nivel de la ejecución de un sistema, sin profundizar en los detalles internos de los mensajes. Los parámetros de entrada y salida de una acción se pueden mostrar usando las relaciones de flujo que conectan la acción y un estado de flujo de objeto.

Un diagrama de actividades puede contener bifurcaciones, así como divisiones de control en hilos concurrentes. los hilos concurrentes representan actividades que se pueden realizar concurrentemente por los diversos objetos o personas.

Las actividades concurrentes se pueden realizar simultáneamente o en cualquier orden. Un diagrama de actividades es como un organigrama tradicional, excepto que permite el control de concurrencia además del control secuencial.

3.2.2.3.2 Diagrama de Estados.

Muestra el conjunto de estados por los cuales pasa un objeto durante su vida en una aplicación, junto con los cambios que permiten pasar de un estado a otro Identifica un periodo de tiempo del objeto (no instantáneo) en el cual el objeto esta esperando alguna operación, tiene cierto estado característico o puede recibir cierto tipo de estímulos. Se representa mediante un rectángulo con los bordes redondeados, que puede tener tres compartimientos: uno para el nombre, otro para el valor característico de los atributos del objeto en ese estado y otro para las acciones que se realizan al entrar, salir o estar en un estado (entry, exit o do), respectivamente Se marcan también los estados iniciales y finales mediante los símbolos  y  , respectivamente.

3.2.3 TABLA DE ASPECTOS METODOLOGICOS

TABLA 3.1

<table>
<tr><td>PARADIGMA
ESPIRAL
INCREMENTAL</td><td>METODOLOGIA OMT</td><td colspan="2">LENGUAJE UNIFICADO DE MODELADO</td></tr>
<tr><td rowspan="7">Análisis

Diseño</td><td rowspan="2"></td><td colspan="2">Identificación de Actores</td></tr>
<tr><td colspan="2">Diagrama de Casos de Uso</td></tr>
<tr><td rowspan="3">Modelo Estático</td><td colspan="2">Diagrama de Clases</td></tr>
<tr><td colspan="2">Diccionario de Clases</td></tr>
<tr><td colspan="2">Diagrama de Objetos</td></tr>
<tr><td rowspan="2">Modelo Dinámico</td><td rowspan="2">Diagrama de Interacción:</td><td>Secuencia</td></tr>
<tr><td>Colaboración</td></tr>
<tr><td rowspan="2">Modelo Funcional</td><td colspan="2">Diagrama de Actividades</td></tr>
<tr><td colspan="2">Diagrama de Estados</td></tr>
<tr><td>Construcción</td><td></td><td colspan="2">Racional Rose 2000
SQL Server 2000
Visual Basic 6.0
Cliente / Servidor</td></tr>
<tr><td>Pruebas</td><td>Pruebas funcionales</td><td colspan="2">Descripción de la prueba
Procedimiento de la prueba</td></tr>
<tr><td>Mantenimiento</td><td></td><td colspan="2"></td></tr>
</table>

El análisis y diseño del sistema se encuentra detallado en el anexo A (en el manual técnico)

CAPITULO IV

CONCLUSIONES Y RECOMENDACIONES

4.1 CONCLUSIONES

- Una vez concluido con el desarrollo del sistema podemos llegar a la conclusión que la metodología OMT es sumamente necesaria para poder ir identificando los diferentes problemas que se van descubriendo en el desarrollo de sistema.
- Otra de las conclusiones importantes que se tomará en cuenta es que, para poder realizar un sistema se necesita de información que debe ser proporcionada por las personas que trabajan en la empresa, en nuestra caso tuvimos que interactuar con algunas de las personas que trabajan en el restaurante Metro Café, para poder saber cuáles son las dificultades que ellos tenían en las diferentes labores diarias.
- El paradigma espiral incremental permitió que el usuario evalúe el desarrollo del sistema en todas sus etapas y con esto logramos entregar un producto beta para que el usuario del sistema lo pruebe y nos diga cuáles son las cosas que hay que modificar, hasta llegar así a un producto que satisfaga sus necesidades.

4.2 RECOMENDACIONES

- Es recomendable que la persona que va a estar a cargo del sistema tenga conocimientos de computación y que sepa el manejo del sistema y pueda utilizar las ayudas que permitirán el mejor manejo.
- El administrador del sistema deberá trabajar con contraseña personal para que los empleados no puedan acceder a información de interés por los supervisores.
- Se recomienda antes del uso del sistema, que todo el personal que va a manipular el mismo, haya revisado el manual de usuario para que no exista ningún tipo de complicación con el funcionamiento del sistema Metro.

BIBLIOGRAFIA

- PRESSMAN, R; Ingeniería de Software, un enfoque práctico, Quinta Edición 2000.
- RUMBAUGH, J; Modelado y Diseño Orientado a Objetos. Metodología OMT; Santa Fe, Bogota; Prentice Hall 1997.
- AVILA. C; Modelado con UML. Principios y Aplicaciones; Perú, Primera Edición 2001.

SITIOS WEB DE REFERENCIA.

http://www.itlp.edu.mx/publica/tutoriales/analisis/24.htm
http://fciencias.ens.uabc.mx/~metprog2/cap6.htm
http://www.creangel.com/uml/componente.php
http://www.cs.ualberta.ca/~pfiguero/soo/uml/casos_uso01.html
http://www.dcc.uchile.cl/~psalinas/uml/casosuso.htl
http://www.mcc.unam.mx/~cursos/Objetos/Omt/omt.html
http://www.monografias.com/
http://www.monografias.com/trabajos6/meto/meto.shtml
Microsoft SQL Server – Características
http://www.microsoft.com/latam/sql/evaluation/features/default.asp
Microsoft Visual Basic 6_0
http://www.microsoft.com/catalog/display.asp?subid=45&site=730
http://www.monografias.com/trabajos24/arquitectura-cliente-servidor/arquitectura-cliente-servidor.shtml#element

GLOSARIO DE TERMINOS

Metodología.- Se trata de construir un modelo de un dominio de aplicación ampliando detalles para que se implementen durante el diseño de un sistema.

SQL2000.- Structured Query Language es un lenguaje declarativo de acceso a bases de datos relacionales estas permiten especificar diversos tipos de operaciones sobre las mismas.

Racional Rose.- La rational rose es una Lenguaje orientado a objetos Unificada que Modela el instrumento de diseño de software intencionado para el modelado visual y la construcción de aplicaciones de software de nivel de la empresa.

Cliente / servidor.- es un modelo para el desarrollo de sistemas de información en el que las transacciones se dividen en procesos independientes que cooperan entre sí para intercambiar información, servicios o recursos.

Help Creator 1.0.- Este sirve para crear ayudas al sistema

Interfaz.- Es necesario de dos sistemas para intercambiar comunicación.

Diagrama casos de uso.- Representa la forma en como un Cliente opera con el sistema en desarrollo, además de la forma, tipo y orden en como los elementos interactúan.

Relación de asociación.- Indica la invocación desde un actor o caso de uso a otra operación.

Diagrama de clases.-El Diagrama de Clases es el diagrama principal para el análisis y diseño. Un diagrama de clases presenta las clases del sistema con sus relaciones estructurales y de herencia.

Diccionario de Clases.- Es una descripción de los procesos que encontramos en el diagrama de clases.

Diagrama de Actividades.- Es una especialización del Diagrama de Estado, organizado respecto de las acciones.

Paradigma.- Es un esquema de control de avance de un proyecto de desarrollo de sistemas.

Usuario.- una persona que manipula el sistema.

Administrador.-Es la persona que se encarga de la funcionalidad del sistema en el momento de que personas ajenas no pueden ingresar.

Empleado.- Es la persona que ingresa las ventas y saca facturas.

Proveedor.- Es la persona que nos entrega el producto

ANEXO A

SISTEMA DE FACTURACIÓN Y VENTAS PARA EL RESTAURANTE METRO CAFE

MANUAL TÉCNICO

Marzo 2006

CONTENIDO

ANÁLISIS

MODELO ESTÁTICO

1.- MODELO ESTÁTICO

1.1 Descripción de actores

Administrador.- Es quien se encarga de administrar y controlar personal, adquisiciones, ventas e inventario.

*Empleado***.-** Es quien se encarga de registrar hora de entrada y salida, además ingresa el producto que se va a vender.

Proveedores.- Es quien se encarga de entregar productos con sus respectiva factura al administrador.

1.2 Descripción de casos de uso

Administrar producto.- En este proceso el administrador se encarga de revisar y hacer consultas del producto, para saber si está en buen estado o sino dar de baja. El administrador consulta reportes de compra, venta, productos para imprimirlos y saber sobre la existencia de los mismos.

*Administrar personal***.-** El empleado se encarga de ingresar la hora de entrada y de salida, en la cual se Irán acumulando las horas de trabajo, a éstas sólo podrá acceder el administrador para revisarlas.

*Administrar proveedores***.-** El administrador registra los datos del proveedor y le entrega una lista de pedidos, el administrador recibe una lista de productos y entrega una orden de pago, el proveedor se encarga de entregar la factura con sus respectivos detalles al administrador.

*Administrar adquisiciones***.-** El administrador se encarga de almacenar una lista de pedidos y luego procede a registrarlas y le entregan los productos

adquiridos y registra aquellas adquisiciones, verifica las condiciones del producto y actualiza su existencia.

Administrar ventas.- El empleado registra el detalle de los pedidos destinados para la venta, registra el tipo de venta e imprime una nota de venta, el empleado solicita los datos del cliente para registrarlos y emitir una factura valida por el SRI. El administrador revisa cada una de las ventas y si existen errores procede a la anulación de ventas mal realizadas y actualiza las facturas.

1.3 Diagramas de casos de uso

CASO DE USO DEL MÓDULO ADMINISTRAR PRODUCTO.

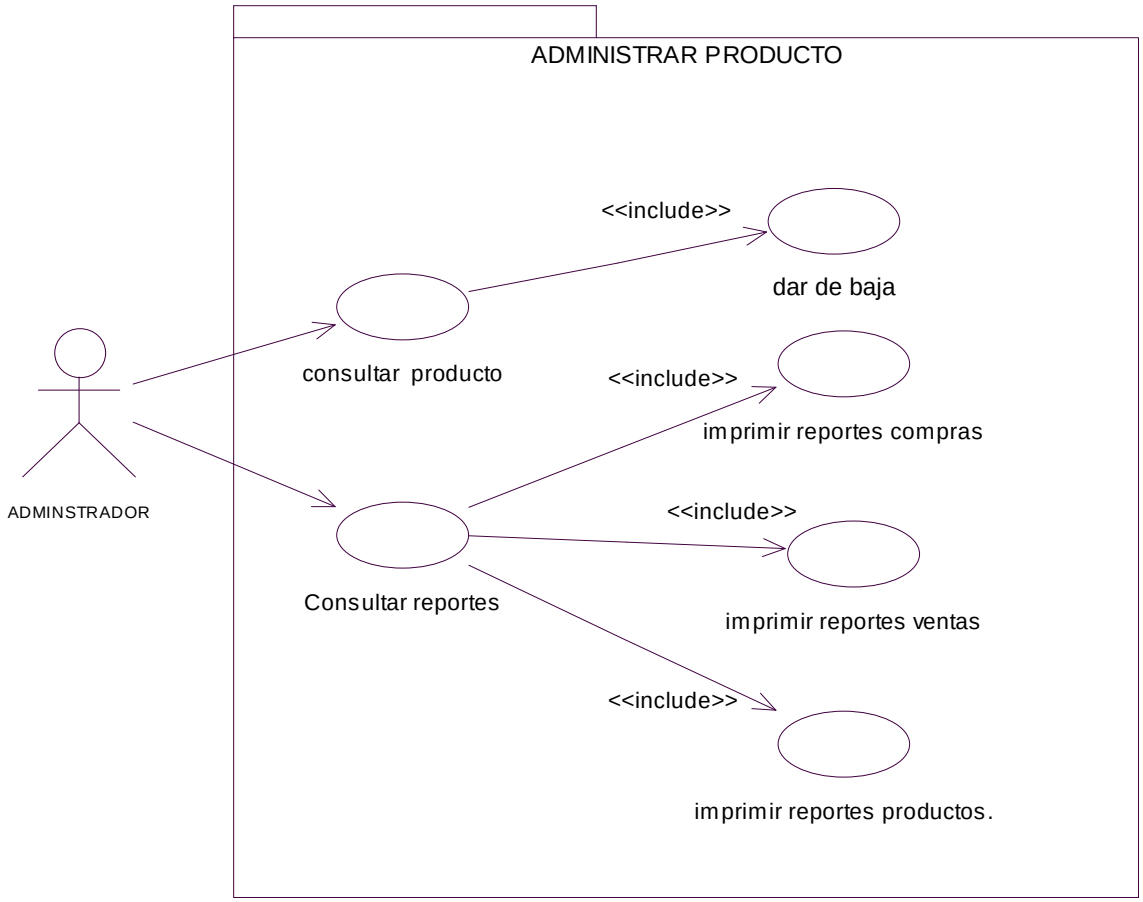


FIGURA 4.1

CASO DE USO DEL MÓDULO ADMINISTRAR PERSONAL

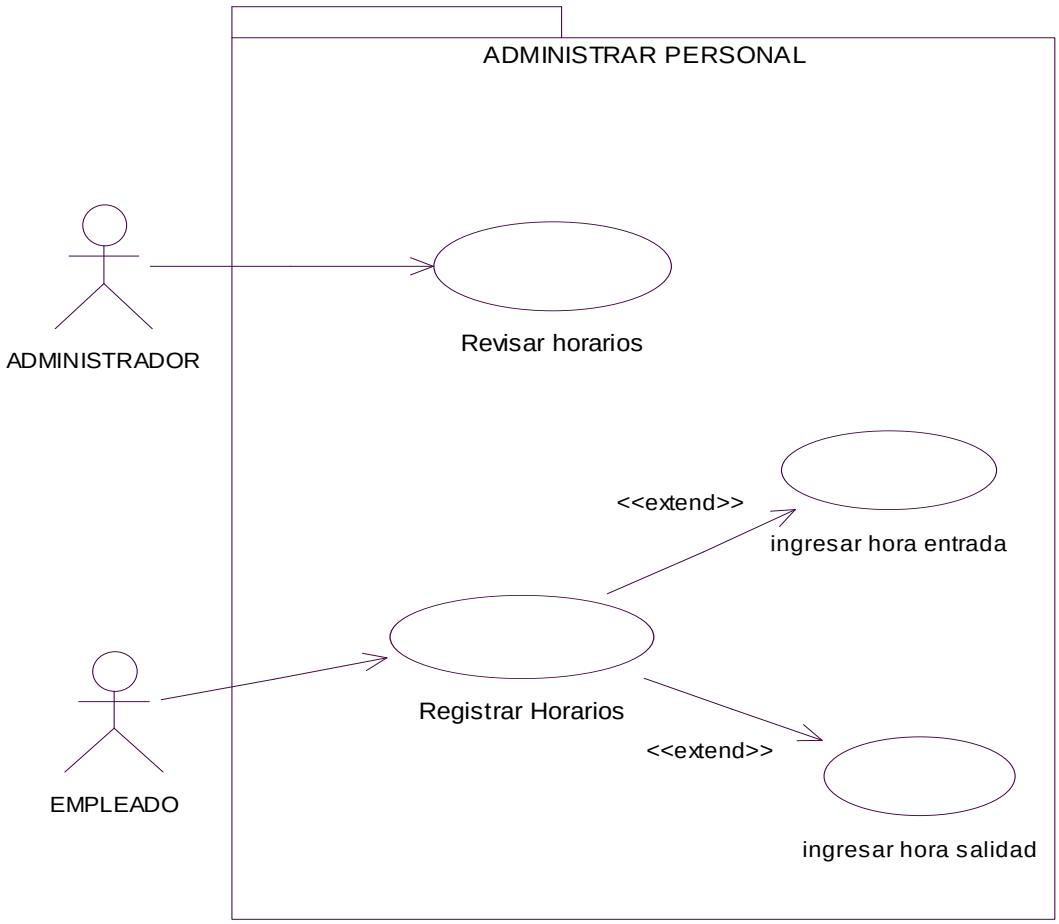


FIGURA 4.2

CASO DE USO DEL MÓDULO ADMINISTRAR PROVEEDORES.

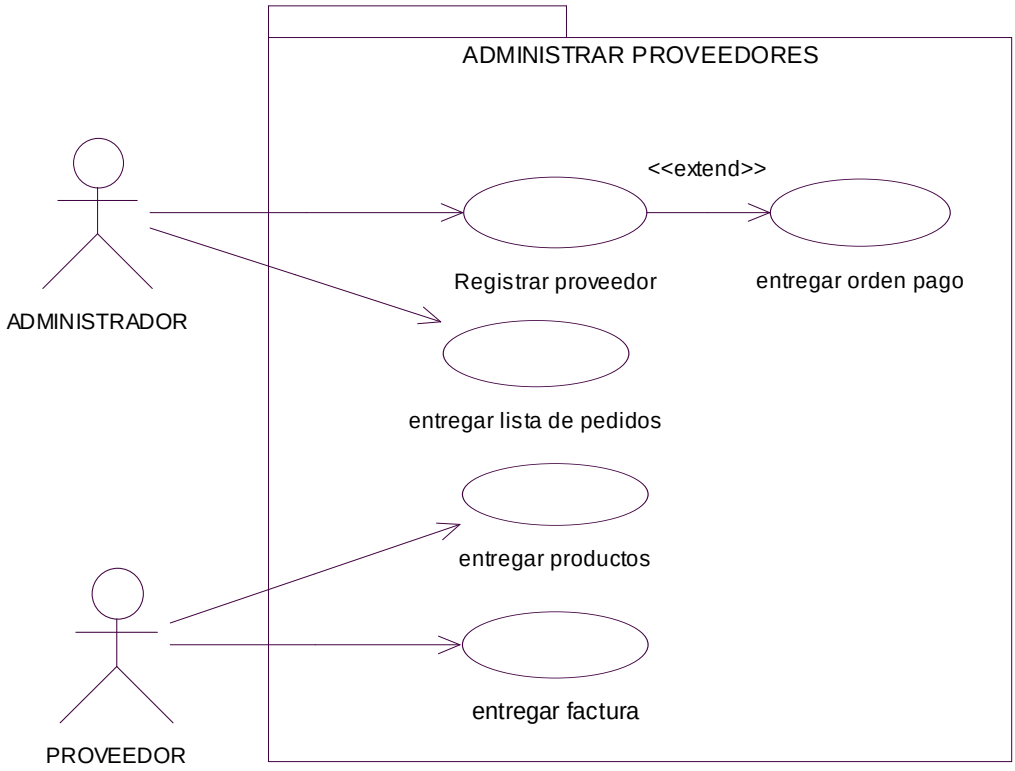


FIGURA 4.3

CASO DE USO DEL MÓDULO ADMINISTRAR ADQUISICIONES.

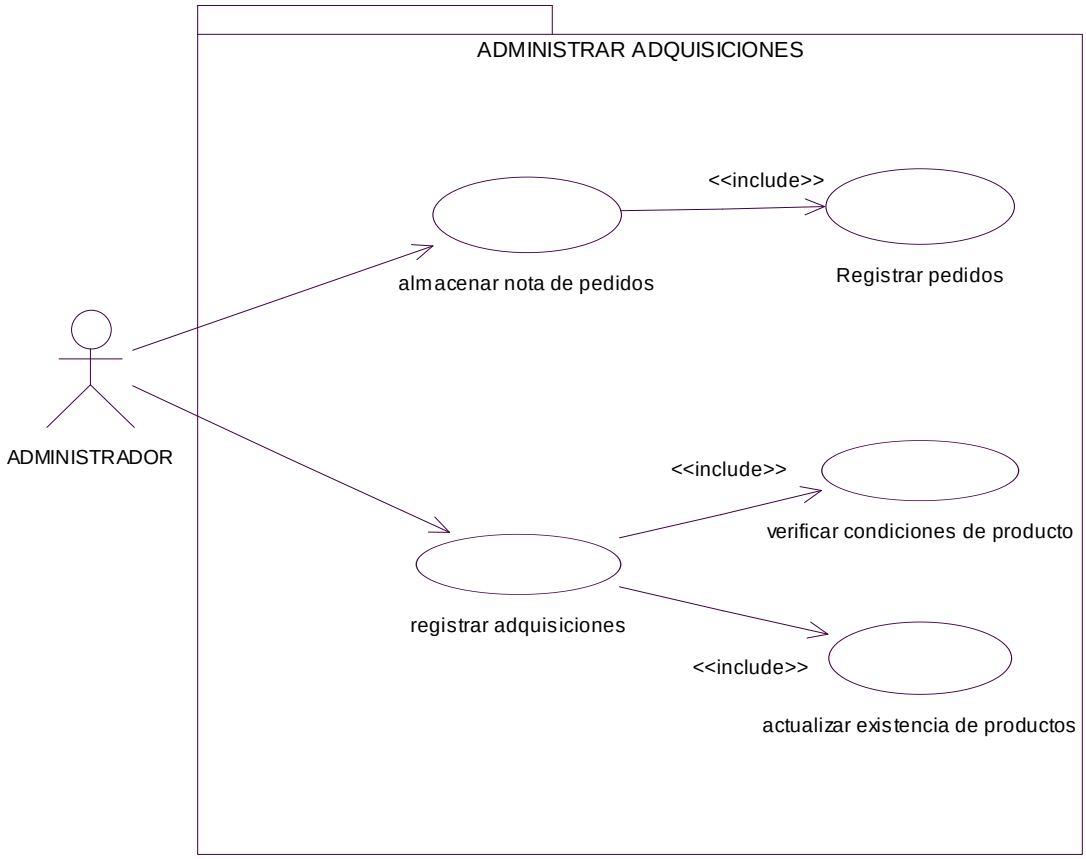


FIGURA 4.4

CASO DE USO DEL MÓDULO ADMINISTRAR VENTAS.

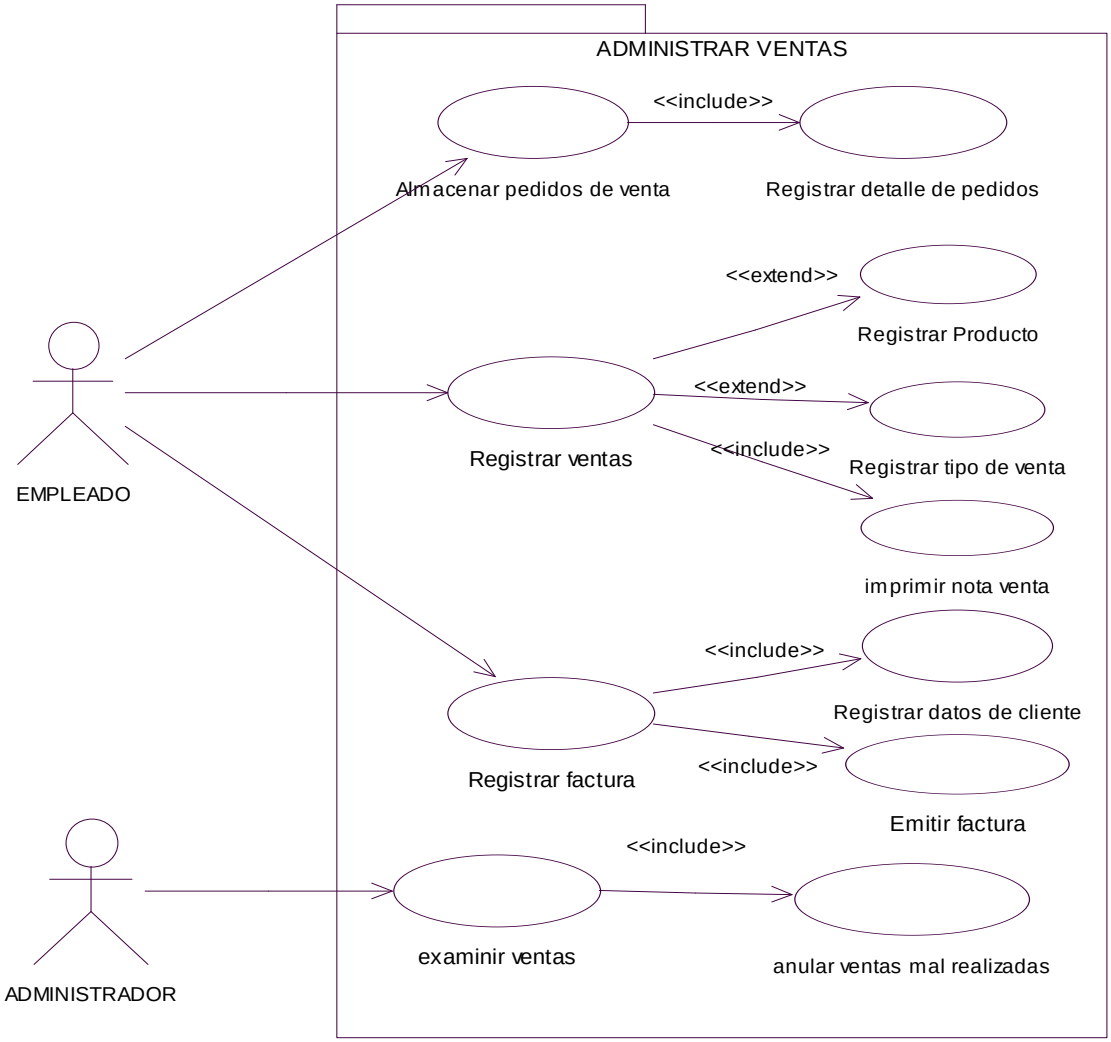
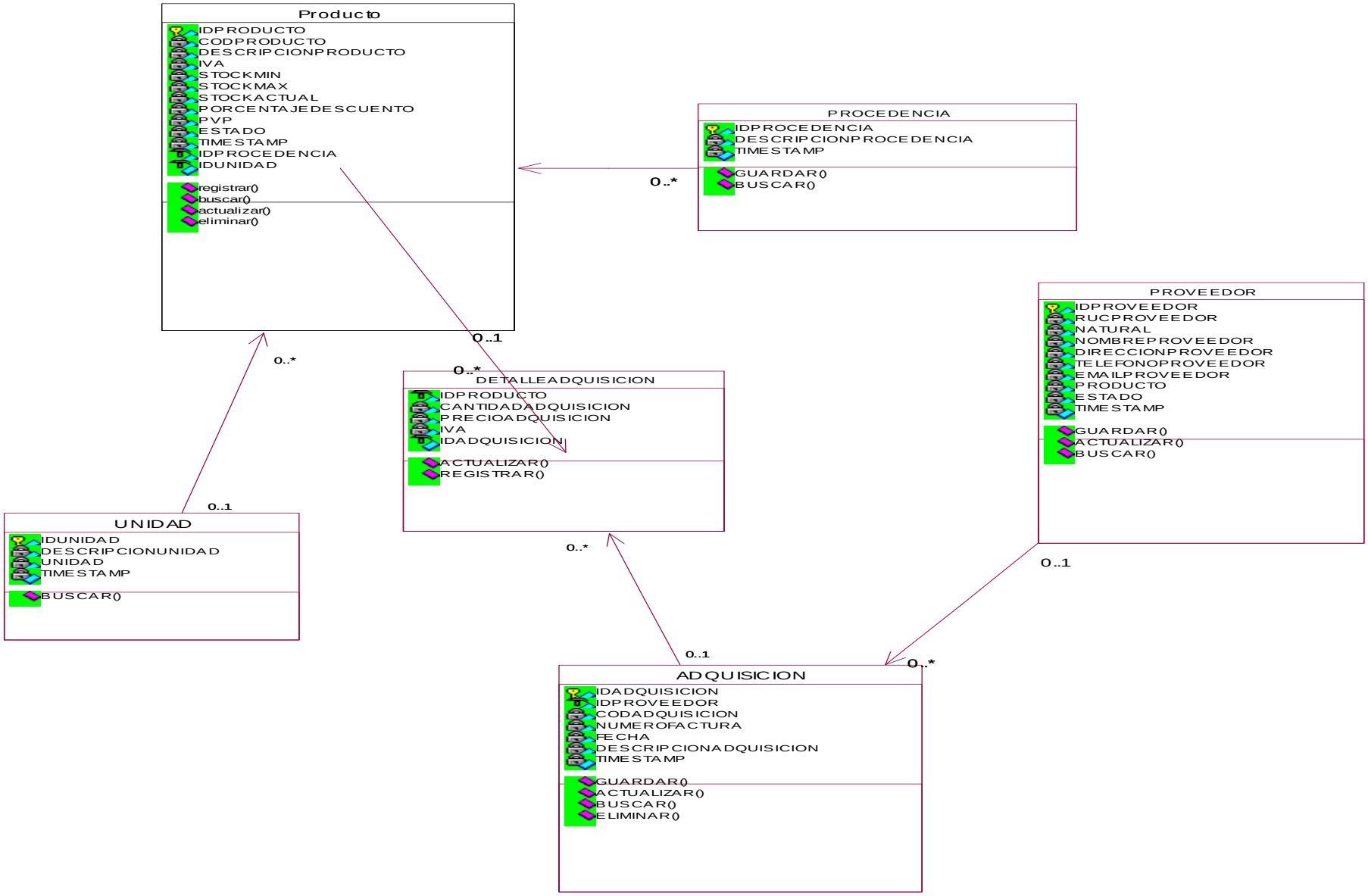
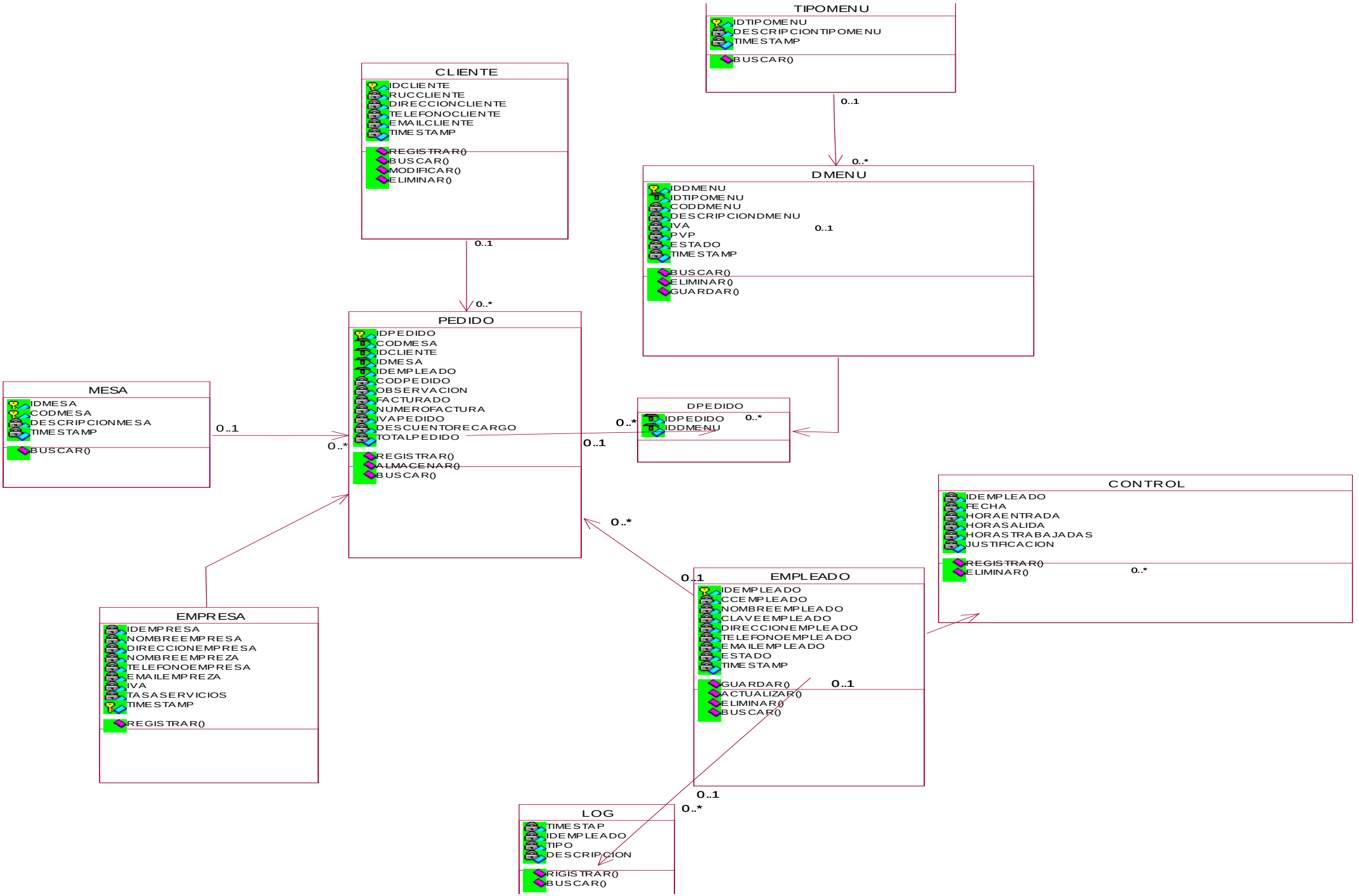


FIGURA 4.5

1.4 Diagrama de Clases

DIAGRAMA DE CLASES DEL SISTEMA DEL RESTAURANTE METRO CAFE


Producto
IDPRODUCTO
CODPRODUCTO
DESCRIPCIONPRODUCTO
IVA
STOCKMIN
STOCKMAX
STOCKACTUAL
PORCENTAJEDESCUENTO
PVP
ESTADO
TIMESTAMP
IDPROCEDENCIA
IDUNIDAD
registrar()
buscar()
actualizar()
eliminar()
PROCEDENCIA
IDPROCEDENCIA
DESCRIPCIONPROCEDENCIA
TIMESTAMP
GUARDAR()
BUSCAR()
0..*
0..1
0..*
0..1
DETALLEADQUISICION
IDPRODUCTO
CANTIDADADQUISICION
PRECIOADQUISICION
IVA
IDADQUISICION
ACTUALIZAR()
REGISTRAR()
PROVEEDOR
IDPROVEEDOR
RUCPROVEEDOR
NATURAL
NOMBREPROVEEDOR
DIRECCIONPROVEEDOR
TELEFONOPROVEEDOR
EMAILPROVEEDOR
PRODUCTO
ESTADO
TIMESTAMP
GUARDAR()
ACTUALIZAR()
BUSCAR()
0..1
UNIDAD
IDUNIDAD
DESCRIPCIONUNIDAD
UNIDAD
TIMESTAMP
BUSCAR()
0..*
0..1
0..1
0..*
ADQUISICION
IDADQUISICION
IDPROVEEDOR
CODADQUISICION
NUMEROFACTURA
FECHA
DESCRIPCIONADQUISICION
TIMESTAMP
GUARDAR()
ACTUALIZAR()
BUSCAR()
ELIMINAR()


TIPOMENU
IDTIPOMENU
DESCRIPCIONTIPOMENU
TIMESTAMP
BUSCAR()
0..1
0..*
CLIENTE
IDCLIENTE
RUCCLIENTE
DIRECCIONCLIENTE
TELEFONOCLIENTE
EMAILCLIENTE
TIMESTAMP
REGISTRAR()
BUSCAR()
MODIFICAR()
ELIMINAR()
DMENU
IDDMENU
IDTIPOMENU
CODDMENU
DESCRIPCIONDMENU
IVA
PVP
ESTADO
TIMESTAMP
BUSCAR()
ELIMINAR()
GUARDAR()
0..1
0..*
PEDIDO
IDPEDIDO
CODMESA
IDCLIENTE
IDMESA
IDEMPLEADO
CODPEDIDO
OBSERVACION
FACTURADO
NUMEROFACTURA
IVAPEDIDO
DESCUENTORECARGO
TOTALPEDIDO
REGISTRAR()
ALMACENAR()
BUSCAR()
MESA
IDMESA
CODMESA
DESCRIPCIONMESA
TIMESTAMP
BUSCAR()
0..1
0..*
DPEDIDO
IDPEDIDO
IDDMENU
0..*
0..*
0..1
CONTROL
IDEMPLEADO
FECHA
HORAENTRADA
HORASALIDA
HORASTRABAJADAS
JUSTIFICACION
REGISTRAR()
ELIMINAR()
0..*
EMPLEADO
IDEMPLEADO
CCEMPLEADO
NOMBREEMPLEADO
CLAVEEMPLEADO
DIRECCIONEMPLEADO
TELEFONOEMPLEADO
EMAILEMPLEADO
ESTADO
TIMESTAMP
GUARDAR()
ACTUALIZAR()
ELIMINAR()
BUSCAR()
0..1
0..1
0..*
EMPRESA
IDEMPRESA
NOMBREEMPRESA
DIRECCIONEMPRESA
NOMBREEMPREZA
TELEFONOEMPRESA
EMAILEMPREZA
IVA
TASASERVICIOS
TIMESTAMP
REGISTRAR()
LOG
TIMESTAP
IDEMPLEADO
TIPO
DESCRIPCION
RIGISTRAR()
BUSCAR()

1.5 Diccionario de Clases

Tabla pedido.- En esta tabla registramos todo lo que se refiere al pedido del cliente, además se registra el número de factura, descuentos, la fecha y hora en que se hizo el pedido.

CAMPO	DESCRIPCION
IDPEDIDO	Numero para identificar el pedido
CODPEDIDO	Código único del pedido
OBSERVACION	Observación al momento de hacer el pedido
FACTURADO	Venta realizada
NUMERO FACTURA	Numero único de la factura
IVAPEDIDO	Es el porcentaje del 12% de iva
DESCUENTORECARGO	Campo si existe un algún descuento
TOTALPEDIDO	Total final a pagar
FECHA	Fecha en la que se hizo el pedido
HORA	Hora en la que se hizo el pedido

Tabla mesa.- en esta tabla registramos el número de mesa en la cual vamos a realizar la venta.

CAMPO	DESCRIPCION
IDMESA	Número propio de la mesa y servirá para su identificación.
CODMESA	Es un código que se asigna para la relación con los pedidos.

DESCRIPCIONMESA	Información de lo que contiene la mesa.
TIMESTAMP	Marca la versión de las filas de la tabla.

Tabla Empleado.- Es la que se encarga de registrar toda la información del empleado.

CAMPO	DESCRIPCION
IDEMPLEADO	Número único que se le asigna al empleado.
CCEMPLEADO	Código del empleado que sirve para ingresar a revisar el estado de la mesa.
NOMBREEMPLEADO	Identificación del empleado
CLAVEEMPLEADO	Clave para ingresar.
DIRECCIONEMPLEADO	Datos de dirección.
TELEFONOEMPLEADO	Datos de teléfono.
EMAILEMPLEADO	Datos de correo electrónico.
ESTADO	Datos de estado.
TIMESTAMP	Marca la versión de las filas de la tabla.

Tabla Control.- controla la hora de entrada y salida de los empleados.

CAMPO	DESCRIPCION
FECHA	Fecha actual.
HORAENTRADA	Hora que ingresa el trabajador.
HORASALIDA	Hora de egreso del trabajador.
HORASTRABAJADAS	Horas acumuladas de trabajo en un mes.
JUSTIFICACION	Modificación de horarios.

Tabla Tipo Menu.- consta el menú para la venta que posee la empresa.

CAMPO	DESCRIPCION
IDTIPOMENU	Menú que posee la empresa.
DESCRIPTIONTIPOMENU	Datos del menú descritos.
TIMESTAMP	Marca la versión de las filas de la tabla.

Tabla Log.- guarda el log de transacciones.

CAMPO	DESCRIPCION
TIMESATMP	Marca la versión de las filas de la tabla.
TIPO	Tipo de transacción a realizar.
DESCRIPCION	Describe la transacción que se realizó

Tabla Empresa.- registra los datos de la empresa y sus actividades.

CAMPO	DESCRIPCION
IDEMPRESA	Código de la empresa.
NOMBREEMPRESA	Nombre de la empresa.
DIRECCIONEMPRESA	Dirección de la empresa.
TELEFONOEMPRESA	Teléfono de la empresa.
EMAILEMPRESA	Correo de la empresa.
IVA	Impuestos de el iva.
TASASERVICIOS	Impuestos de servicio.
TIMESTAMP	Marca la versión de las filas de la tabla.

Tabla Cliente.- Registra los datos del cliente al cual le realizamos la venta.

CAMPO	DESCRIPCION
IDCLIENTE	Código de cliente.
RUCCLIENTE	Identidad del cliente
NOMBRECLIENTE	Nombre del cliente.
DIRECCIONCLIENTE	Dirección del cliente.
TELEFONOCLIENTE	Teléfono del cliente
EMAILCLIENTE	Correo de cliente.
TIMESTAMP	Marca la versión de las filas de la tabla.

Tabla Unidad.- transforma las unidades a las necesidades del trabajador.

CAMPO	DESCRIPCION
IDUNIDAD	Identificación de la unidad de conversión.
DESCRIPCIONUNIDAD	Descripción de las unidades.
UNIDAD	Unidades de conversión.
DESCRIPCIONCONVESION	Descripción de las unidades de descripción.
TIMESTAMP	Marca la versión de las filas de la tabla.

Tabla DMenu.- Esta tabla muestra el menú que ofrece le empresa con los impuestos.

CAMPO	DESCRIPCION
IDDMENU	Identificación del menú.
CODDMENU	Código del menú.
DESCRIPCIONDMENU	Descripción del menú.
IVA	Impuesto al valor agregado.
PVP	Precio de venta al público.
ESTADO	Estado de venta.
TIMESTAMP	Marca la versión de las filas de la tabla.

Tabla Adquisición.- Describe las compras que se realizan.

CAMPO	DESCRIPCION
IDADQUISICION	Código para las compras.
CODADQUISICION	Código para cada compra.
NUNEROFACTURA	Número de la factura.
FECHA	Fecha de compra.
DESCRIPCIONADQUISICION	Descripción de compra.
TIMESTAMP	Marca la versión de las filas de la tabla.

Tabla Producto.- Describe el producto que adquirimos, y observamos el número de artículos que tenemos a disposición, desechamos producto y realizamos descuentos si es necesario.

CAMPO	DESCRIPCION
IDPRODUCTO	Identificación del producto.
CODPRODUCTO	Código del producto.
DESCRIPCIONPRODUCTO	Descripción del producto.

IVA	Impuestos del producto.
STOCKMIN	Número mínimo de artículos que debemos tener.
STOCKACTUAL	Número actual de artículos
PORCENTAJEDESCUENTO	Descuentos a realizar si son necesarios.
PVP	Precio de venta al público
ESTADO	Estado del producto.
TIMESTAMP	Marca la versión de las filas de la tabla.

Tabla Procedencia.- verificamos la procedencia del producto.

CAMPO	DESCRIPCION
IDPROCEDENCIA	Código de procedencia del producto.
DESCRIPCIONPROCEDENCIA	Descripción de la procedencia del producto.
TIMESTAMP	Marca la versión de las filas de la tabla.

Tabla Detalle Adquisición.- controlamos la cantidad que vamos a adquirir del producto y el precio.

CAMPO	DESCRIPCION
CANTIDADADQUISICION	Cantidad a comprar.
PRECIOADQUISICION	Precio de producto a comprar.
IVA	Impuestos a pagar.

Tabla Proveedor.- Administramos el proveedor y registramos los datos.

CAMPO	DESCRIPCION
IDPROVEEDOR	Código del proveedor.
RUCPROVEEDOR	Identidad del proveedor.
NATURAL	Marca la versión de las filas de la tabla.
NOMBREPROVEEDOR	Nombre del proveedor.
DIRECCIONPROVEEDOR	Dirección de proveedor.
TELEFONOPROVEEDOR	Teléfono de proveedor.
EMAILPROVEEDOR	Correo de proveedor.
PRODUCTO	Producto que nos entrega.
ESTADO	Estado del producto
TIMESTAMP	Marca la versión de las filas de la tabla.

DISEÑO

MODELO DINÁMICO

2 MODELO DINÁMICO

2.1 Diagramas de Interacción

2.1.1 Diagrama de Secuencia

DIAGRAMA DE SECUENCIA DEL MÓDULO ADMINISTRAR PRODUCTO.

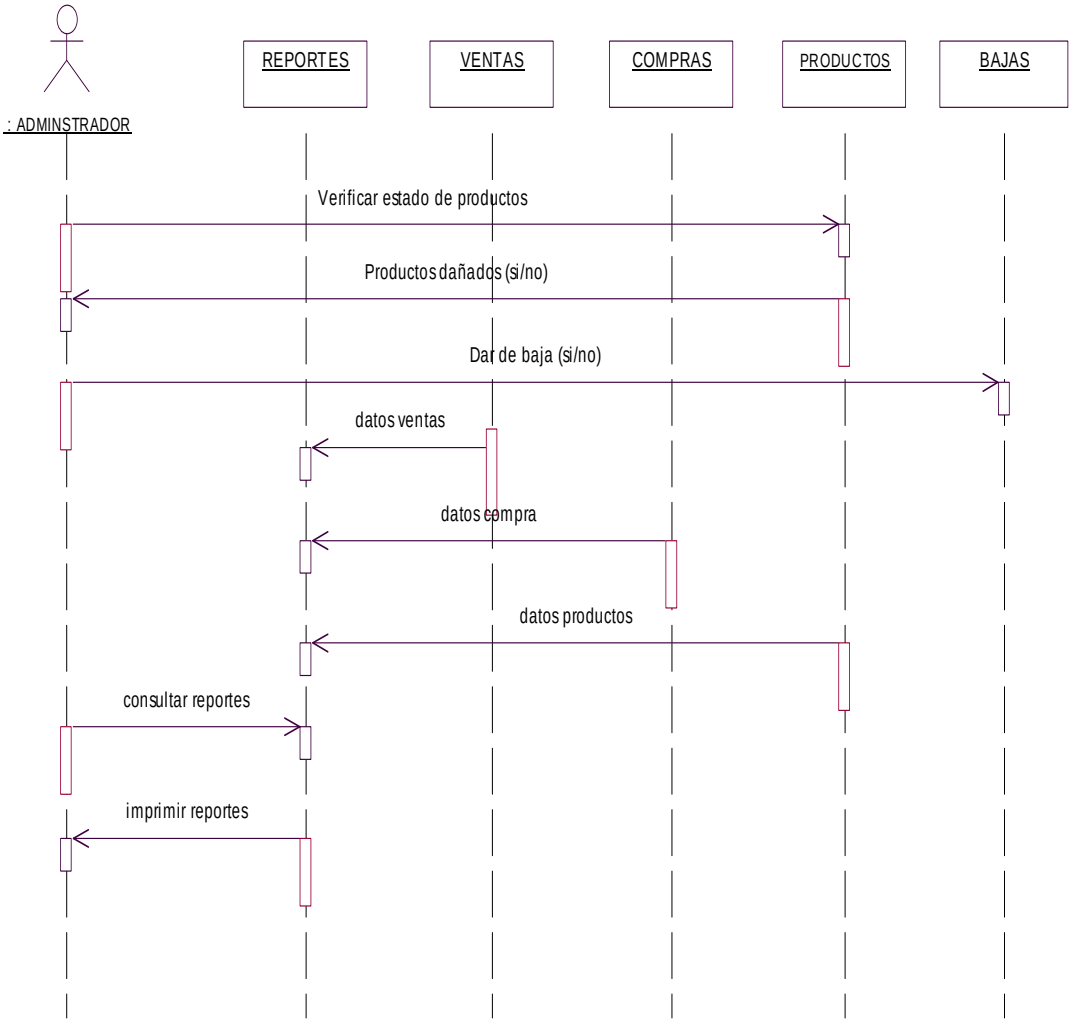


FIGURA 4.10

DIAGRAMA DE SECUENCIA ADMINISTRAR PERSONAL.

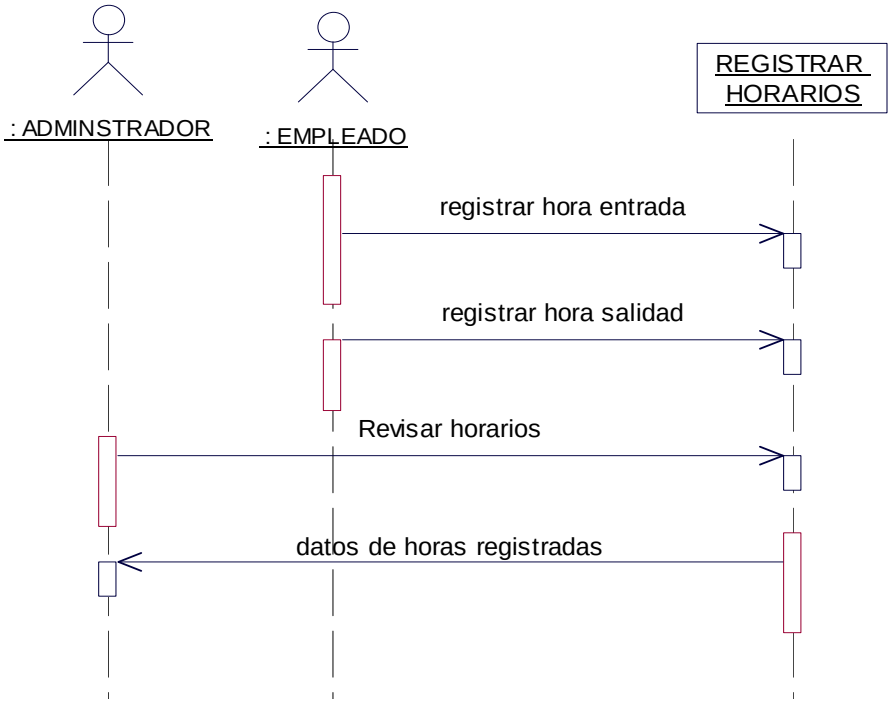


FIGURA 4.11

DIAGRAMA DE SECUENCIA DE ADMINISTRAR PROVEEDORES.

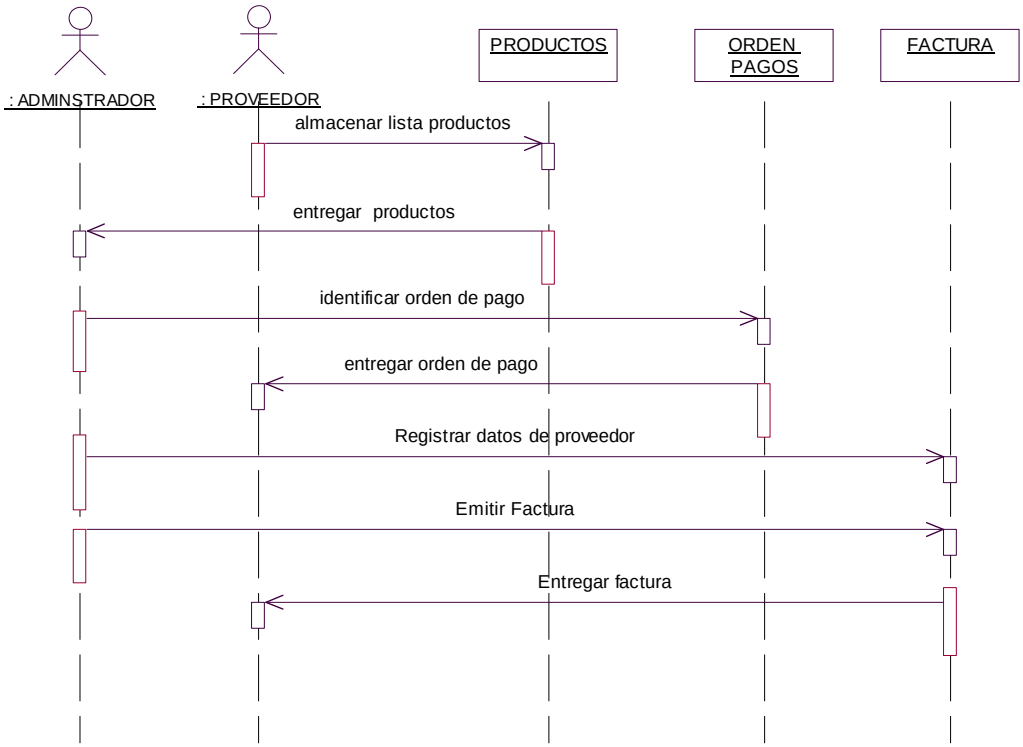


FIGURA 4.12

DIAGRAMA DE SECUENCIA ADMINISTRAR ADQUISICIONES.

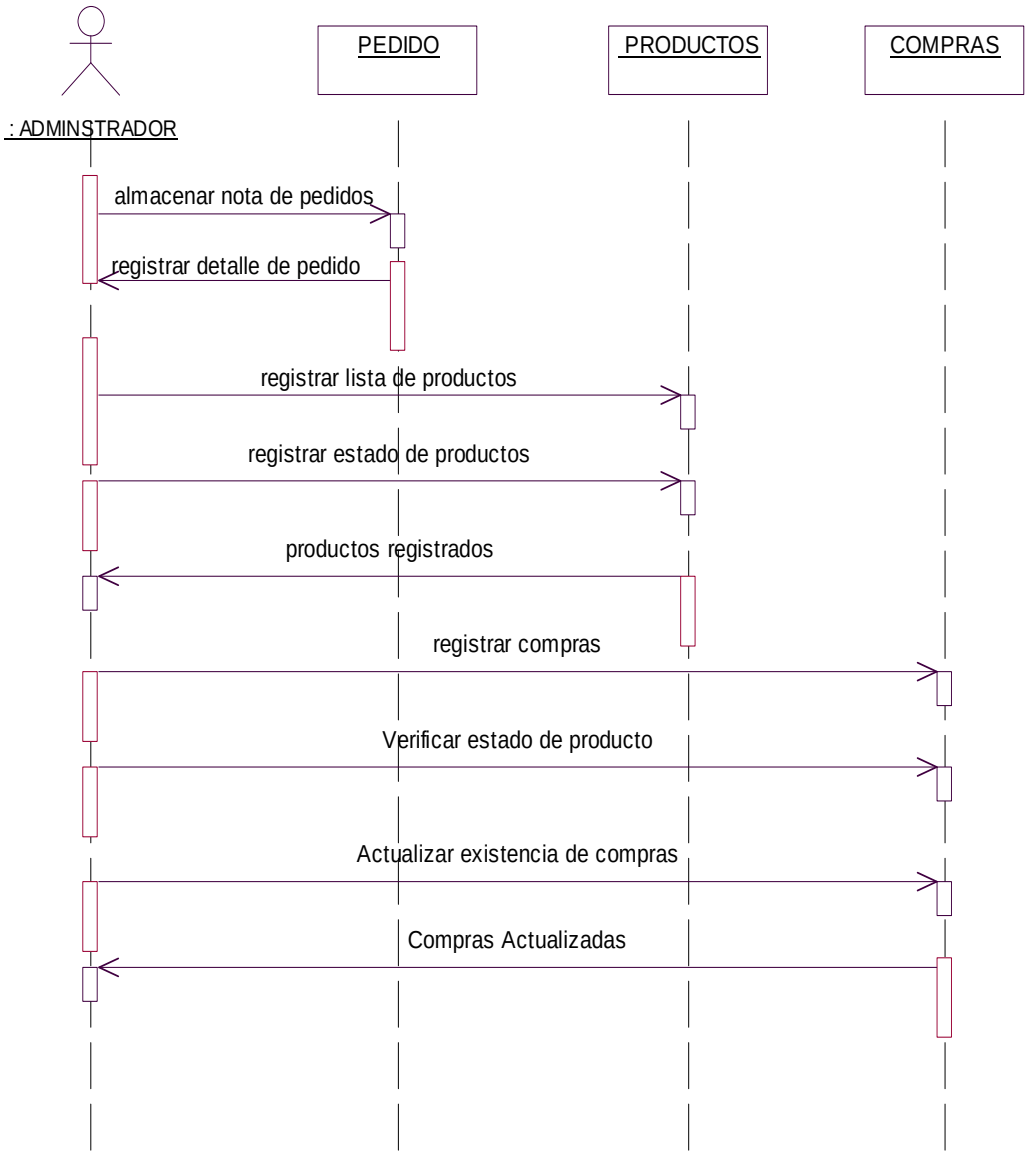


FIGURA 4.13

DIAGRAMA DE SECUENCIA DEL MÓDULO ADMINISTRAR VENTAS.

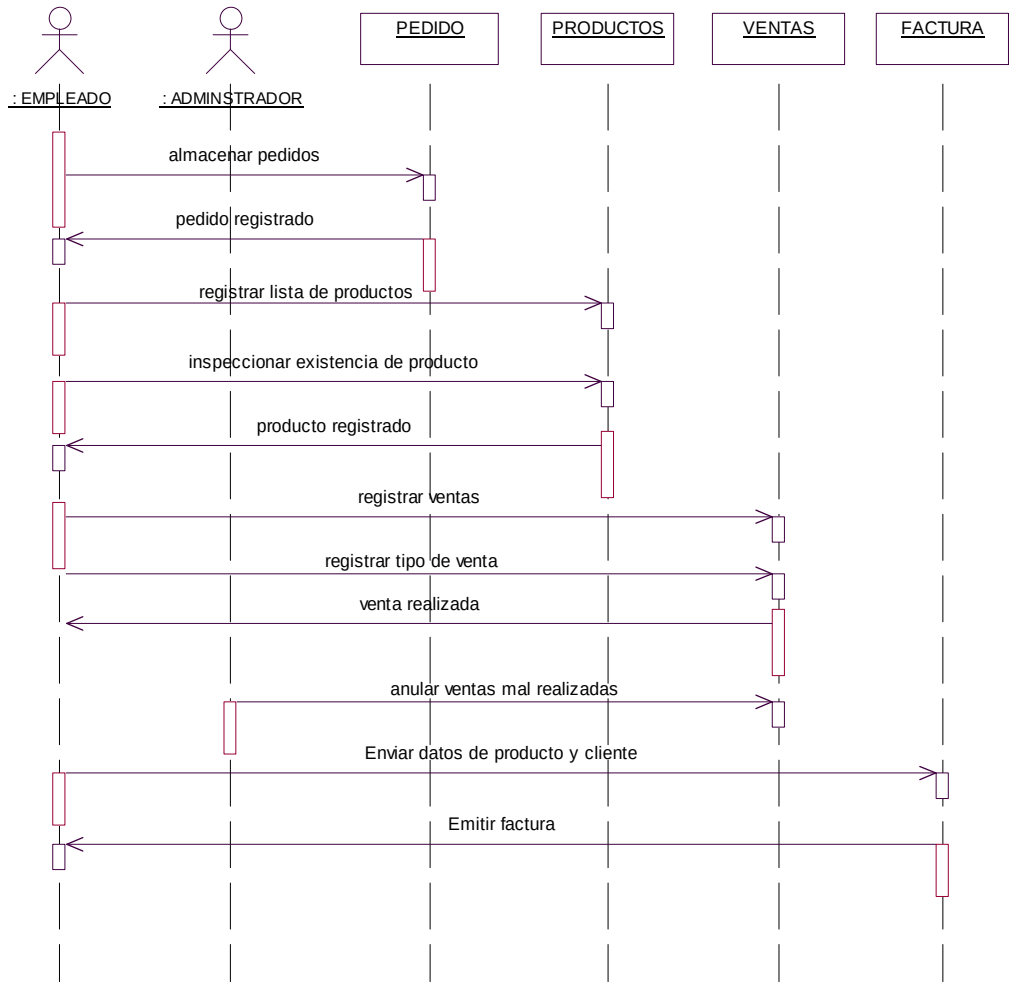


FIGURA 4.14

2.1.2Diagrama de colaboración

DIAGRAMA DE COLABORACIÓN DEL MÓDULO ADMINISTRAR PRODUCTO.

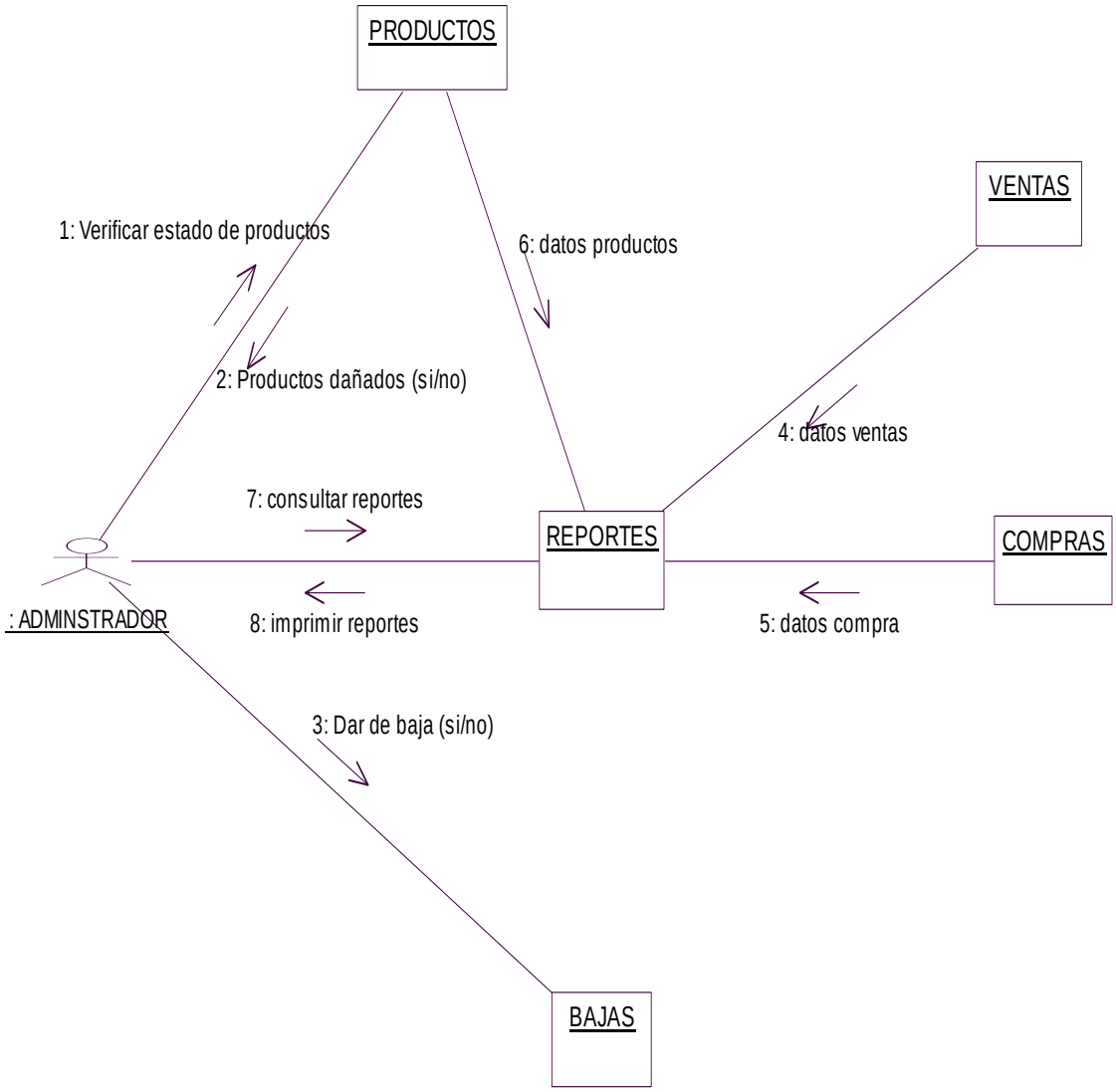


FIGURA 4.15

DIAGRAMA DE COLABORACIÓN DEL MÓDULO ADMINISTRAR PERSONAL

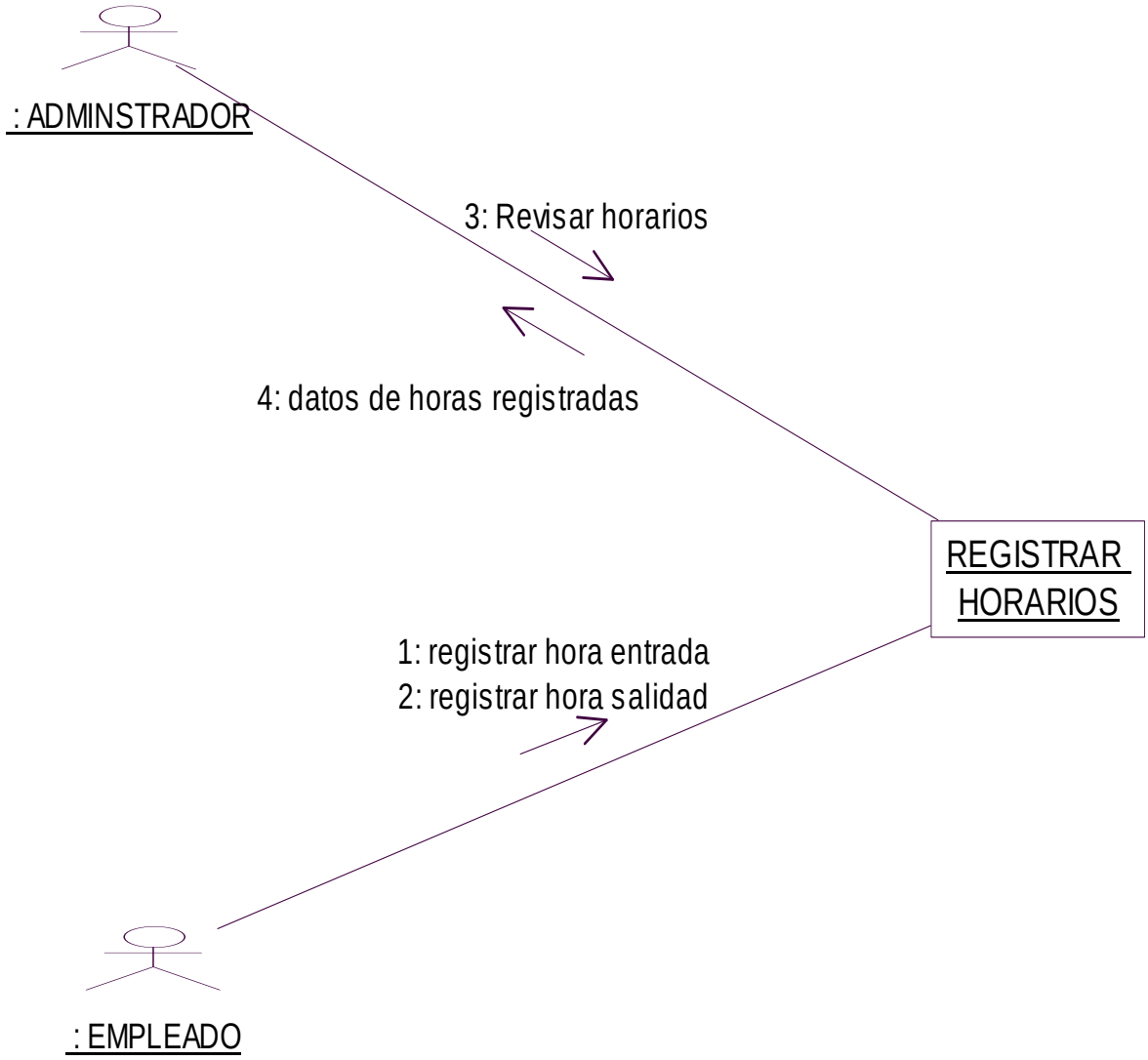


FIGURA 4.16

DIAGRAMA DE COLABORACIÓN DEL MÓDULO ADMINISTRAR PROVEEDORES

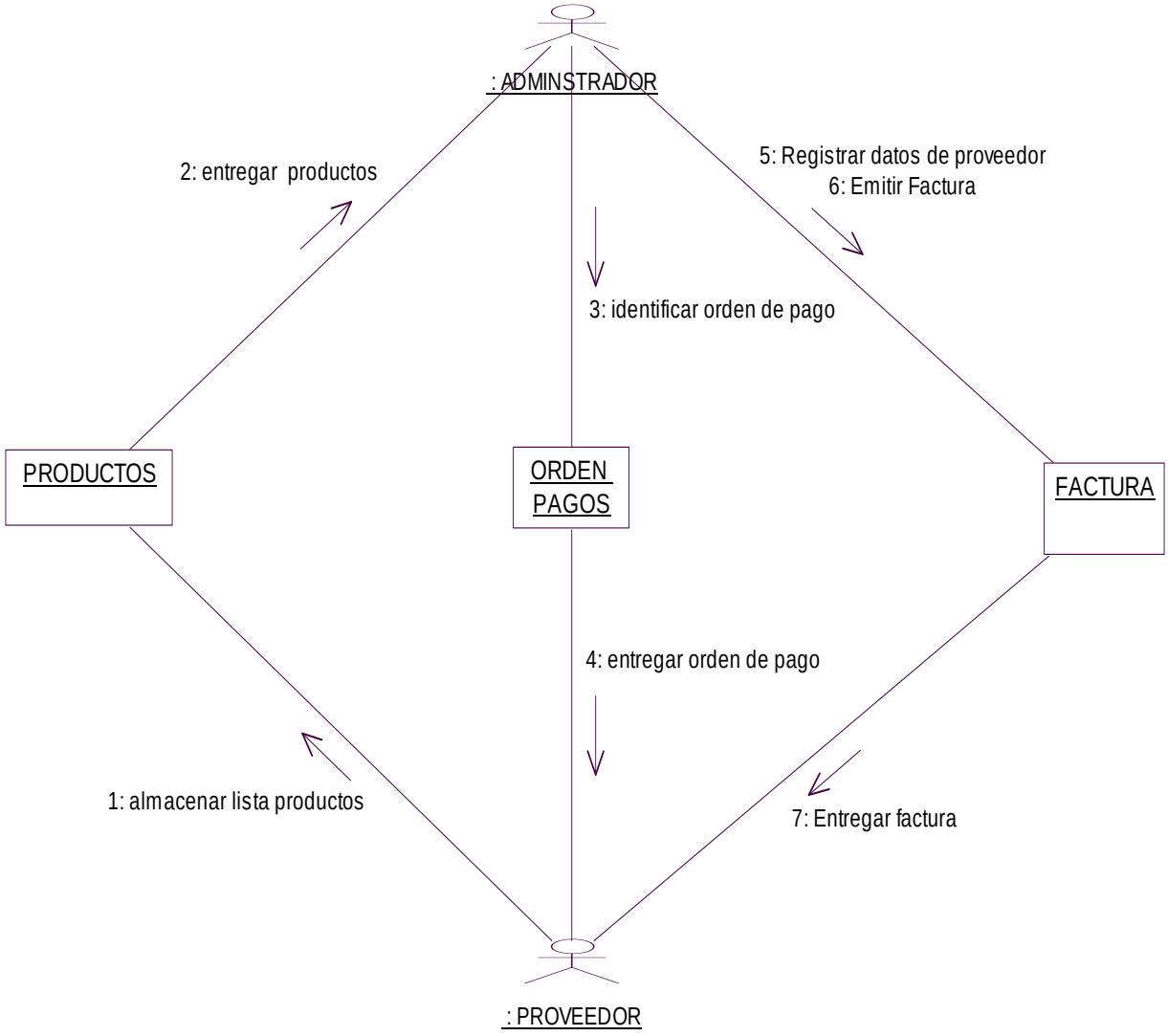


FIGURA 4.17

DIAGRAMA DE COLABORACIÓN DEL MÓDULO ADMINISTRAR ADQUISICIONES

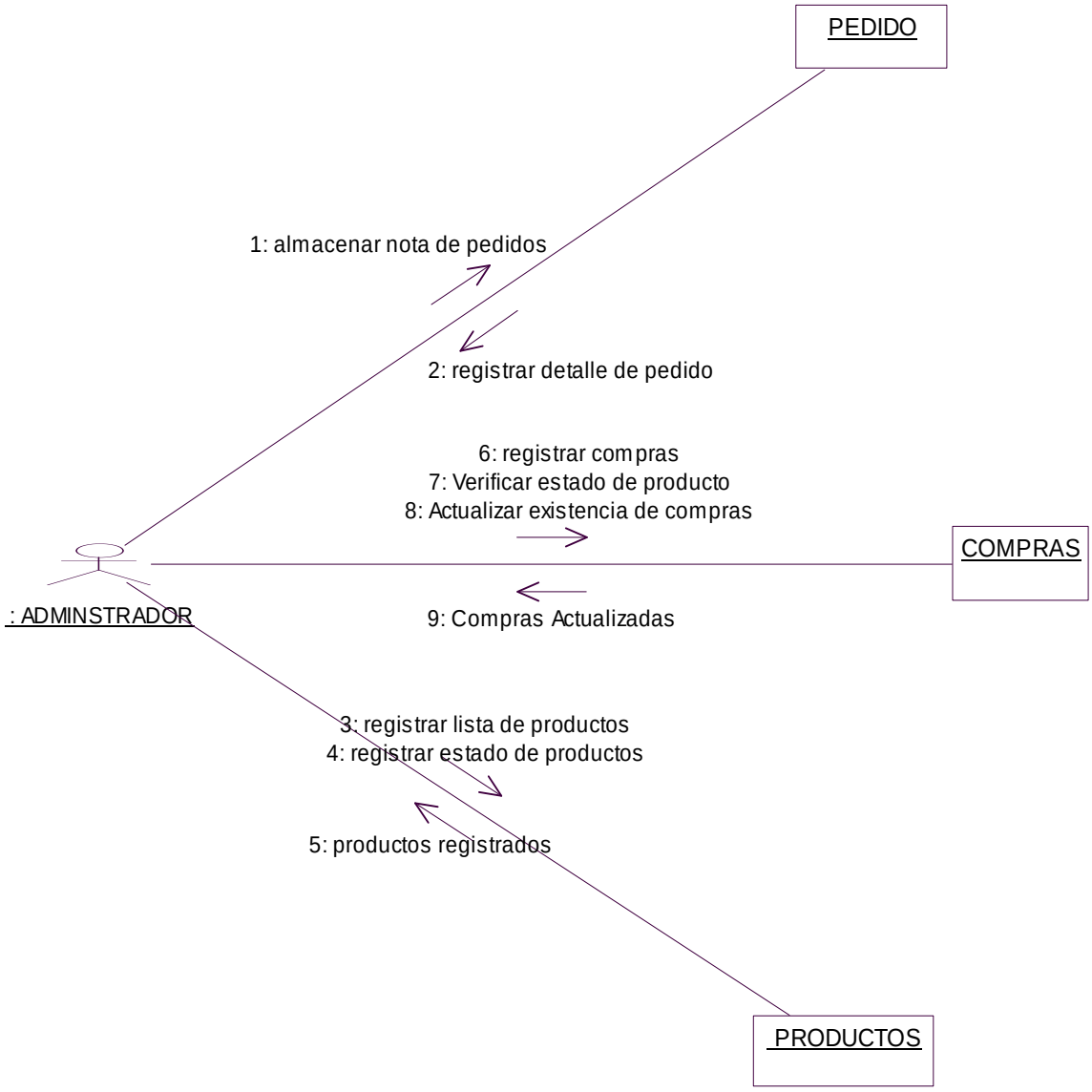


FIGURA 4.18

DIAGRAMA DE COLABORACIÓN DEL MÓDULO ADMINISTRAR VENTAS

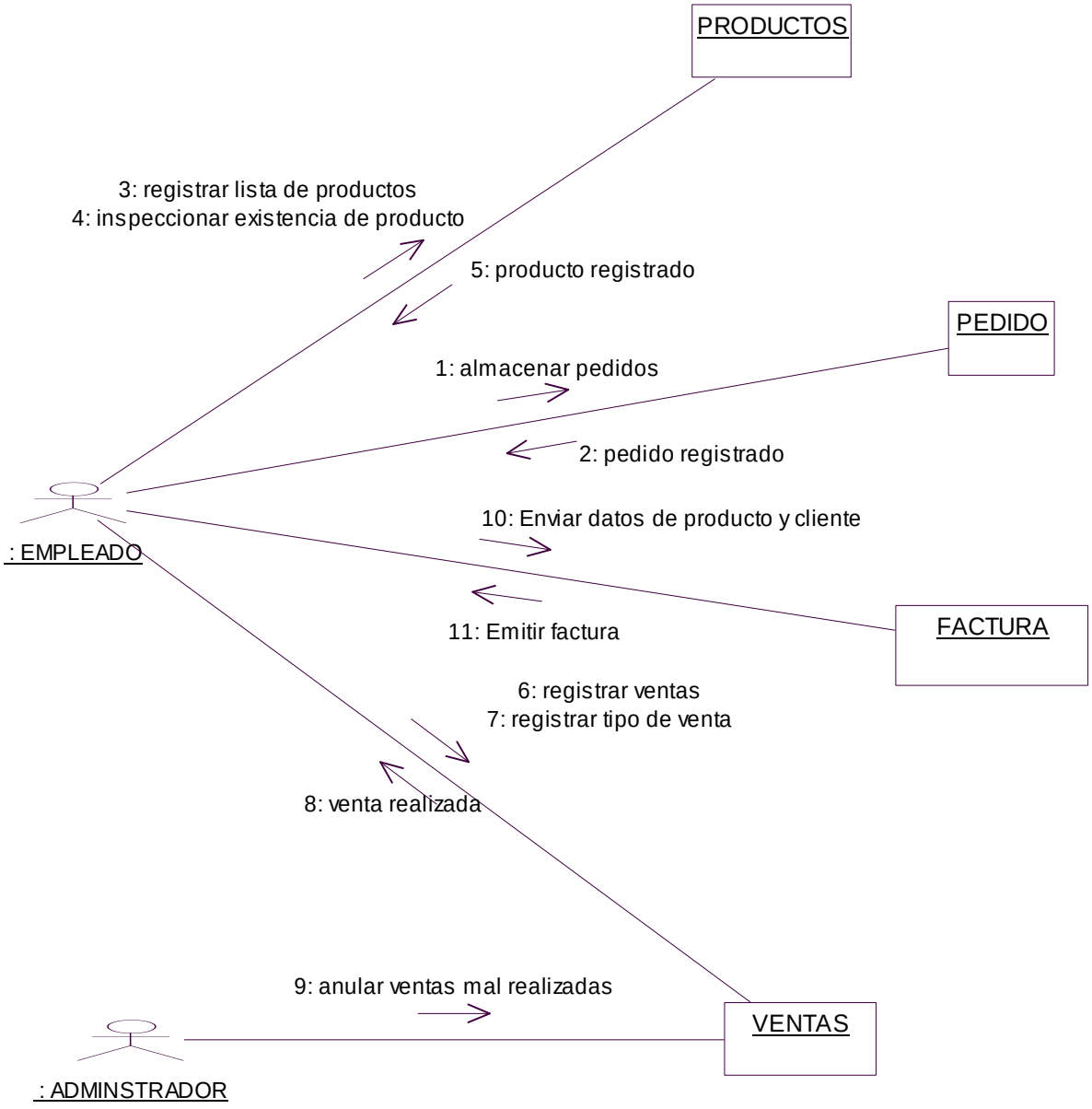


FIGURA 4.19

MODELO FUNCIONAL

6 MODELO DE FUNCIONAL

6.1 Diagrama de Actividades

DIAGRAMA DE ACTIVIDADES DEL MÓDULO ADMINISTRAR PRODUCTO.

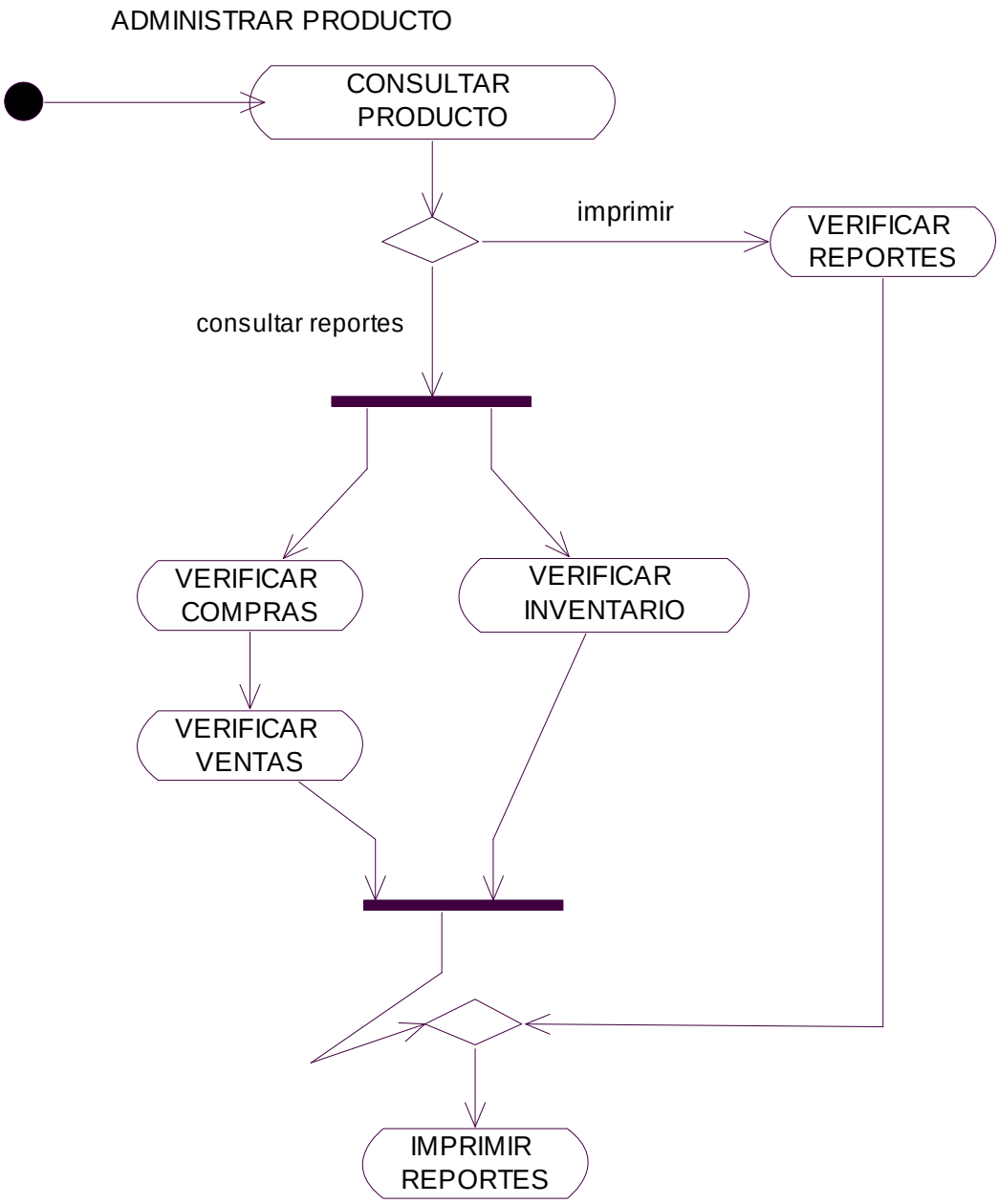


FIGURA 4.20

DIAGRAMA DE ACTIVIDADES DEL MÓDULO ADMINISTRAR PERSONAL.

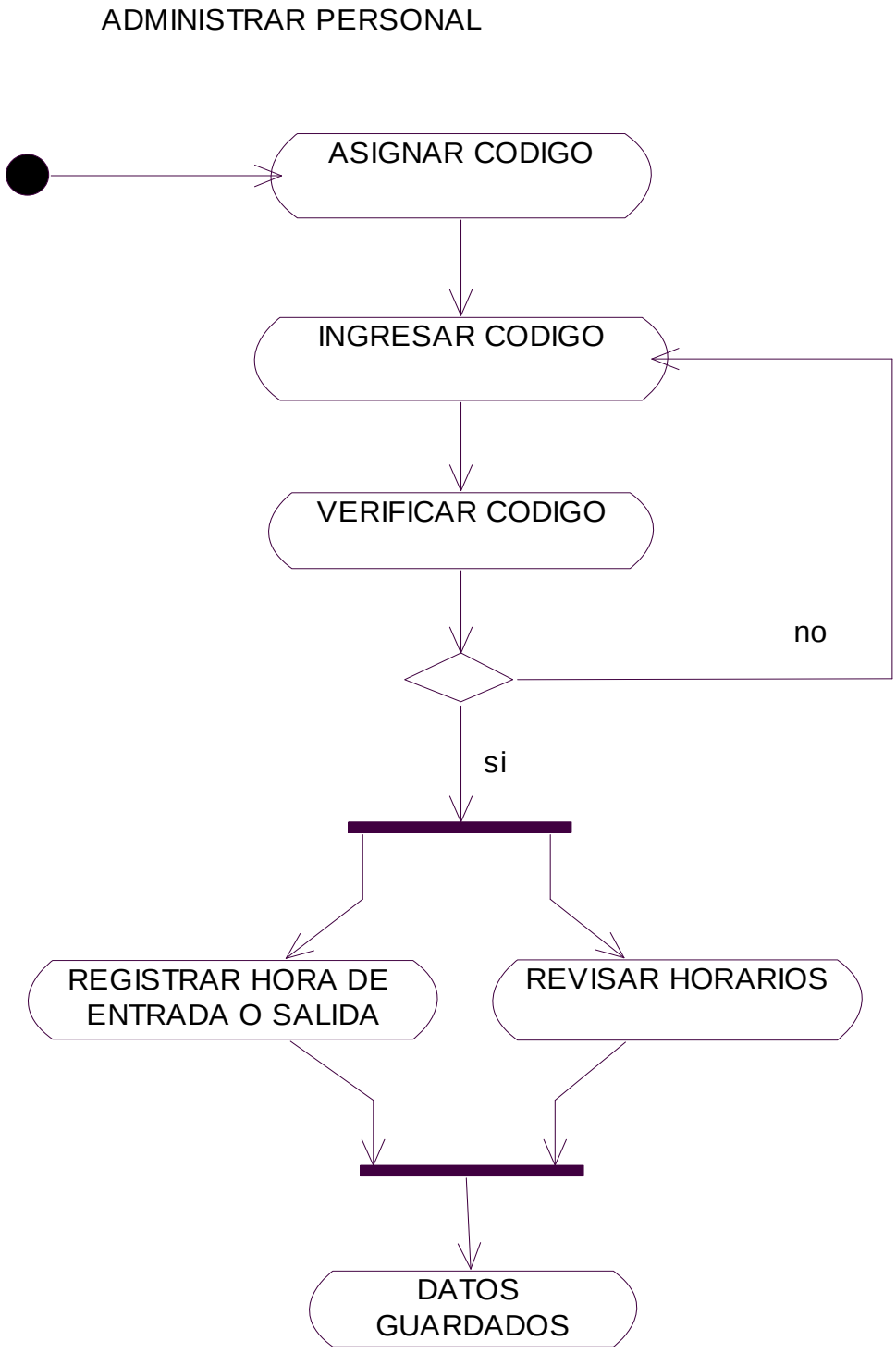


FIGURA 4.21

DIAGRAMA DE ACTIVIDADES DEL MÓDULO ADMINISTRAR PROVEEDORES.

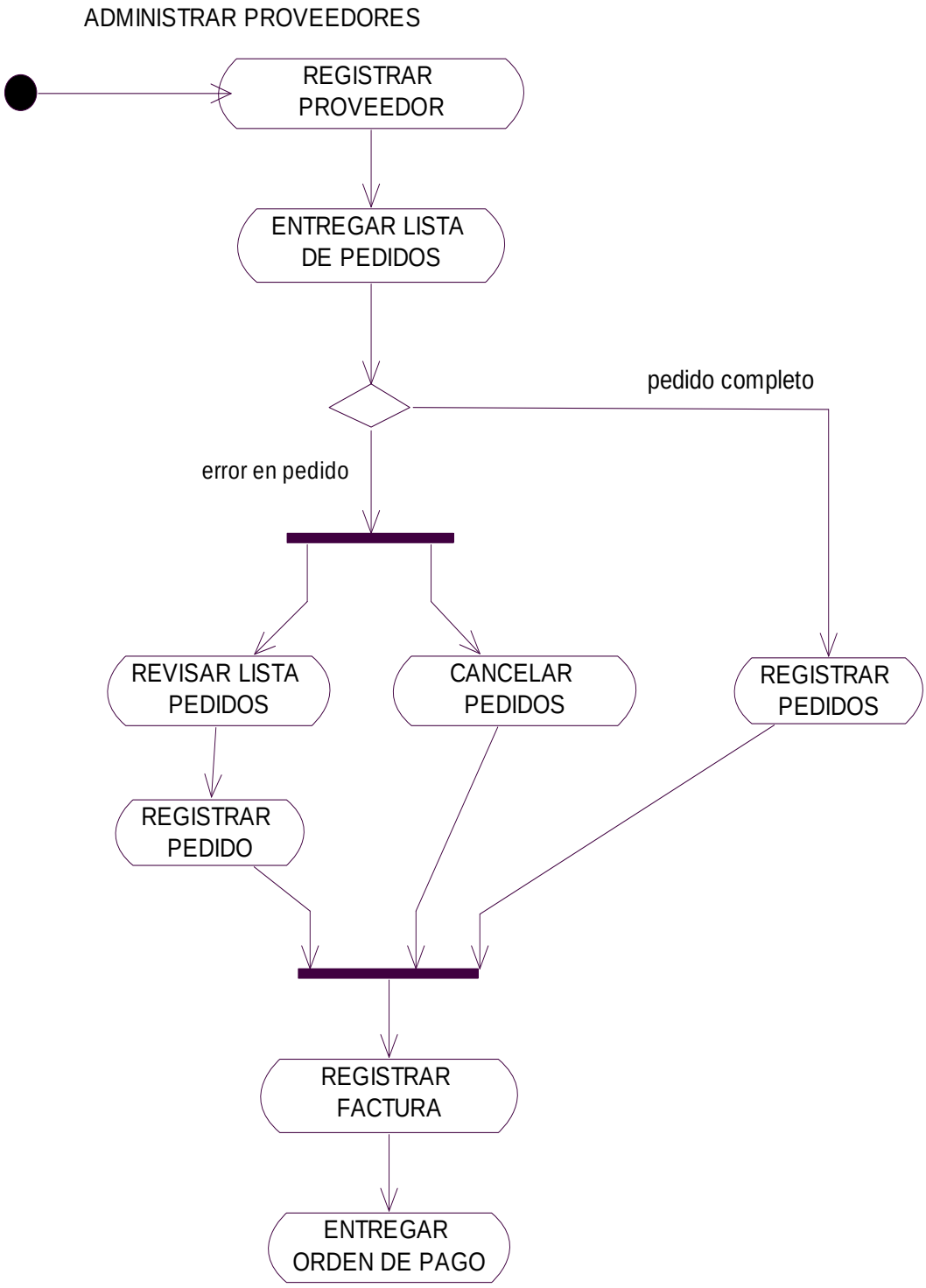


FIGURA 4.22

DIAGRAMA DE ACTIVIDADES DEL MÓDULO ADMINISTRAR ADQUISICIONES.

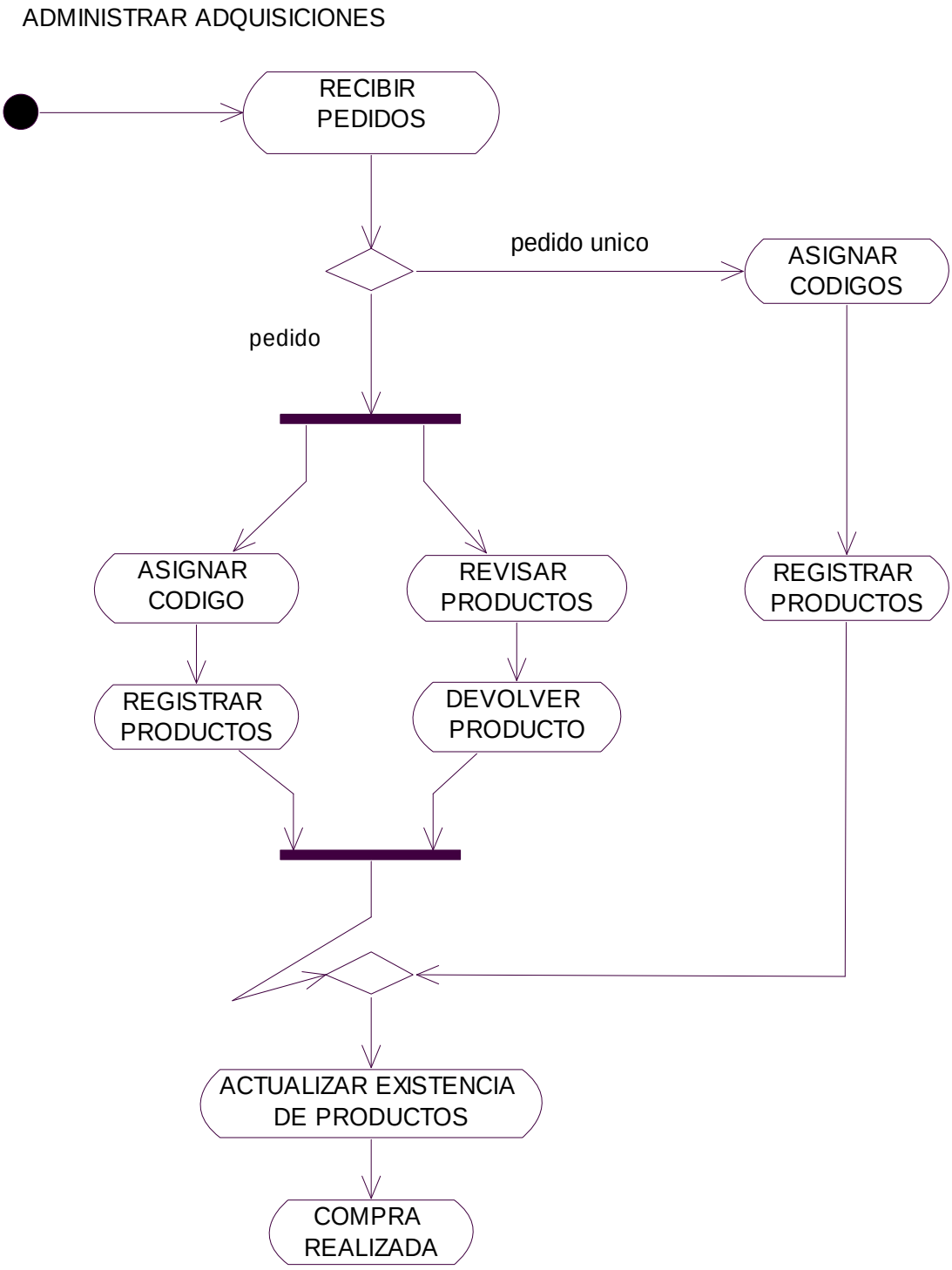


FIGURA 4.23

DIAGRAMA DE ACTIVIDADES DEL MÓDULO ADMINISTRAR VENTAS.

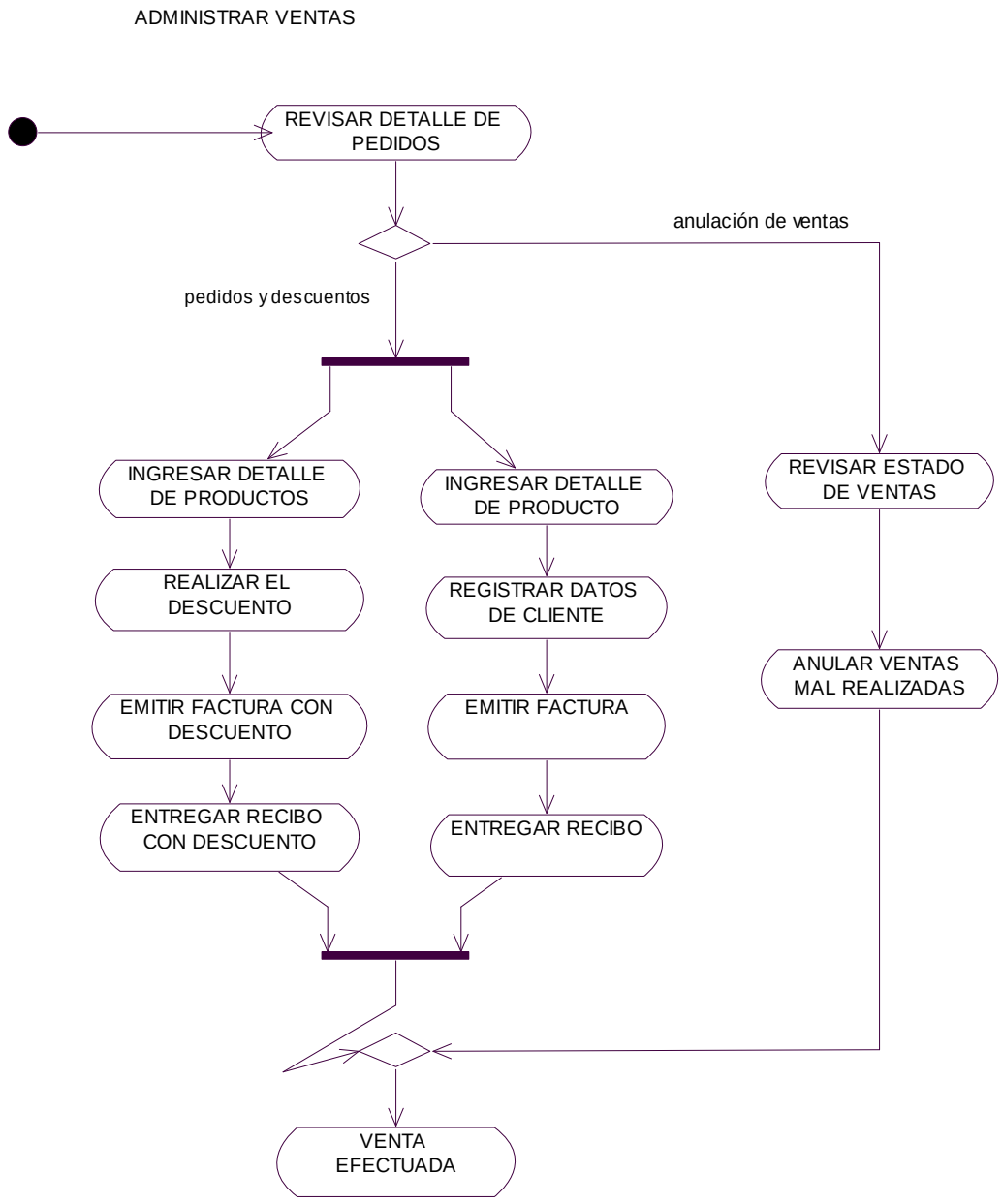


FIGURA 4.24

CONSTRUCCIÓN

7 CONTRUCCION

7.1 SQL SERVER

```
/*==============================================================*/
/* Database name:  METRO                                        */
/* DBMS name:      Microsoft SQL Server 2000                    */
/* Created on:     17/10/2005 22:15:29                          */
/*==============================================================*/


alter table ADQUISICION
   drop constraint FK_ADQUISIC_REFERENCE_PROVEEDO
go


alter table CONTROL
   drop constraint FK_CONTROL_REFERENCE_EMPLEADO
go


alter table DETALLE_ADQUSICION
   drop constraint FK_DETALLE__REFERENCE_ADQUISIC
go


alter table DETALLE_ADQUSICION
   drop constraint FK_DETALLE__REFERENCE_PRODUCTO
go


alter table DPEDIDO
   drop constraint FK_DPEDIDO_REFERENCE_PEDIDO
go


alter table DPEDIDO
   drop constraint FK_DPEDIDO_REFERENCE_DMENU
go


alter table LOG
   drop constraint FK_LOG_REFERENCE_EMPLEADO
go


alter table PEDIDO
   drop constraint FK_PEDIDO_REFERENCE_MESA
```

```
go


alter table PEDIDO
   drop constraint FK_PEDIDO_REFERENCE_EMPLEADO
go


alter table PEDIDO
   drop constraint FK_PEDIDO_REFERENCE_CLIENTE
go


alter table PRODUCTO
   drop constraint FK_PRODUCTO_REFERENCE_UNIDAD
go


alter table PRODUCTO
   drop constraint FK_PRODUCTO_REFERENCE_PROCEDEN
go


if exists (select 1
           from  sysobjects
          where  id = object_id('ADQUISICION')
           and   type = 'U')
   drop table ADQUISICION
go


if exists (select 1
           from  sysobjects
          where  id = object_id('CLIENTE')
           and   type = 'U')
   drop table CLIENTE
go


if exists (select 1
           from  sysobjects
          where  id = object_id('CONTROL')
           and   type = 'U')
   drop table CONTROL
go


if exists (select 1
           from  sysobjects
```

```
          where  id = object_id('DETALLE_ADQUSICION')
          and   type = 'U')
  drop table DETALLE_ADQUSICION
go


if exists (select 1
          from  sysobjects
          where  id = object_id('DMENU')
          and   type = 'U')
  drop table DMENU
go


if exists (select 1
          from  sysobjects
          where  id = object_id('DPEDIDO')
          and   type = 'U')
  drop table DPEDIDO
go


if exists (select 1
          from  sysobjects
          where  id = object_id('EMPLEADO')
          and   type = 'U')
  drop table EMPLEADO
go


if exists (select 1
          from  sysobjects
          where  id = object_id('LOG')
          and   type = 'U')
  drop table LOG
go


if exists (select 1
          from  sysobjects
          where  id = object_id('MESA')
          and   type = 'U')
  drop table MESA
go


if exists (select 1
          from  sysobjects
          where  id = object_id('PEDIDO')
```

```
          and   type = 'U')
  drop table PEDIDO
go


if exists (select 1
          from  sysobjects
          where  id = object_id('PROCEDENCIA')
          and   type = 'U')
  drop table PROCEDENCIA
go


if exists (select 1
          from  sysobjects
          where  id = object_id('PRODUCTO')
          and   type = 'U')
  drop table PRODUCTO
go


if exists (select 1
          from  sysobjects
          where  id = object_id('PROVEEDOR')
          and   type = 'U')
  drop table PROVEEDOR
go


if exists (select 1
          from  sysobjects
          where  id = object_id('UNIDAD')
          and   type = 'U')
  drop table UNIDAD
go


/*==============================================================*/
/* Table: ADQUISICION                                           */
/*==============================================================*/
create table ADQUISICION (
IDADQUISICION       INT IDENTITY        not null,
IDPROVEEDOR         int                 null,
CODADQUISICION      varchar(50)         not null,
NUMEROFACTURA       varchar(50)         not null,
FECHA               smalldatetime       not null,
DESCRIPCIONADQUISICION varchar(250)     not null,
TIMESTAMP           timestamp           null,
constraint PK_ADQUISICION primary key  (IDADQUISICION)
```

```
)
go


/*==============================================================*/
/* Table: CLIENTE                                               */
/*==============================================================*/
create table CLIENTE (
IDCLIENTE           INT IDENTITY        not null,
RUCCLIENTE          varchar(20)         not null,
NOMBRECLIENTE       varchar(100)        not null,
DIRECCIONCLIENTE    varchar(250)        null,
TELEFONOCLIENTE     varchar(250)        null,
EMAILCLIENTE        varchar(250)        null,
TIMESTAMP           timestamp           null,
constraint PK_CLIENTE primary key  (IDCLIENTE)
)
go


/*==============================================================*/
/* Table: CONTROL                                               */
/*==============================================================*/
create table CONTROL (
IDEMPLEADO          int                 null,
FECHA               smalldatetime       null,
HORAENTRADA         datetime            null,
HORASALIDA          datetime            null,
HORASTRABAJADAS     int                 null,
JUSTIFICACION       varchar(250)        null
)
go


/*==============================================================*/
/* Table: DETALLE_ADQUSICION                                    */
/*==============================================================*/
create table DETALLE_ADQUSICION (
IDADQUISICION       INT                 null,
IDPRODUCTO          int                 null,
CANTIDADADQUISICION  int                not null,
PRECIOADQUISICION   float               not null,
IVA                 bit                 not null
)
go


/*==============================================================*/
/* Table: DMENU                                                 */
```

```
/*==============================================================*/
create table DMENU (
IDDMENU             INT IDENTITY        not null,
CODDMENU            varchar(50)         not null,
DESCRIPCIONDMENU    varchar(250)        not null,
PVP                 float               not null,
IVA                 bit                 not null,
TIPO                int                 null,
ESTADO              bit                 null,
TIMESTAMP           timestamp           null,
constraint PK_DMENU primary key  (IDDMENU)
)
go


/*==============================================================*/
/* Table: DPEDIDO                                               */
/*==============================================================*/
create table DPEDIDO (
IDPEDIDO            INT                 null,
IDDMENU             INT                 null
)
go


/*==============================================================*/
/* Table: EMPLEADO                                              */
/*==============================================================*/
create table EMPLEADO (
IDEMPLEADO          INT IDENTITY        not null,
CODEMPLEADO         varchar(50)         not null,
NOMBREEMPLEADO      varchar(250)        not null,
DIRECCIONEMPLEADO   varchar(250)        null,
TELEFONOEMPLEADO    varchar(250)        null,
EMAILEMPLEADO       varchar(250)        null,
ESTADO              bit                 null,
TIMESTAMP           timestamp           null,
constraint PK_EMPLEADO primary key  (IDEMPLEADO)
)
go


/*==============================================================*/
/* Table: LOG                                                   */
/*==============================================================*/
create table LOG (
TIMESATMP           timestamp           not null,
IDEMPLEADO          int                 null,
TIPO                int                 null,
```

```
DESCRIPCION         varchar(1000)       null,
constraint PK_LOG primary key  (TIMESATMP)
)
go


/*==============================================================*/
/* Table: MESA                                                  */
/*==============================================================*/
create table MESA (
IDMESA              INT IDENTITY        not null,
CODMESA             varchar(50)         not null,
DESCRIPCIONMESA     varchar(250)        not null,
TIMESTAMP           timestamp           null,
constraint PK_MESA primary key  (IDMESA, CODMESA)
)
go


/*==============================================================*/
/* Table: PEDIDO                                                */
/*==============================================================*/
create table PEDIDO (
IDPEDIDO            INT IDENTITY        not null,
CODMESA             varchar(50)         null,
IDCLIENTE           int                 null,
IDMESA              int                 null,
IDEMPLEADO          int                 null,
CODPEDIDO           varchar(50)         not null,
OBSERVACION         varchar(250)        null,
FACTURADO           bit                 null,
NUMEROFACTURA       integer             null,
IVAPEDIDO           float               null,
DESCUENTORECARGO    float               null,
TOTALPEDIDO         float               null,
FECHA               smalldatetime       null,
HORA                varchar(20)         null,
constraint PK_PEDIDO primary key  (IDPEDIDO)
)
go


/*==============================================================*/
/* Table: PROCEDENCIA                                           */
/*==============================================================*/
create table PROCEDENCIA (
IDPROCEDENCIA       INT IDENTITY        not null,
DESCRIPCIONPROCEDENCIA varchar(250)      not null,
TIMESTAMP           timestamp           null,
```

```
constraint PK_PROCEDENCIA primary key  (IDPROCEDENCIA)
)
go


/*==============================================================*/
/* Table: PRODUCTO                                              */
/*==============================================================*/
create table PRODUCTO (
IDPRODUCTO          INT IDENTITY        not null,
CODPRODUCTO         varchar(50)         null,
DESCRIPCIONPRODUCTO  varchar(250)       not null,
IVA                 bit                 null,
IDUNIDAD            int                 null,
IDPROCEDENCIA       int                 null,
STOCKMIN            float               not null,
STOCKMAX            float               not null,
STOCKACTUAL         float               not null,
PORCENTAJEDESCUENTO  float              not null,
PVP                 float               not null,
ESTADO              bit                 null,
TIMESTAMP           timestamp           null,
constraint PK_PRODUCTO primary key  (IDPRODUCTO)
)
go


/*==============================================================*/
/* Table: PROVEEDOR                                             */
/*==============================================================*/
create table PROVEEDOR (
IDPROVEEDOR         INT IDENTITY        not null,
RUCPROVEEDOR        varchar(20)         not null,
NOMBREPROVEEDOR     varchar(100)        not null,
DIRECCIONPROVEEDOR   varchar(250)       null,
TELEFONOPROVEEDOR    varchar(250)       null,
EMAILPROVEEDOR      varchar(250)        null,
PRODUCTO            varchar(250)        null,
ESTADO              bit                 null,
TIMESTAMP           timestamp           null,
constraint PK_PROVEEDOR primary key  (IDPROVEEDOR)
)
go


/*==============================================================*/
/* Table: UNIDAD                                                */
/*==============================================================*/
create table UNIDAD (
```

```
IDUNIDAD            INT IDENTITY        not null,
DESCRIPCIONUNIDAD   varchar(250)        not null,
UNIDAD              float               not null,
TIMESTAMP           timestamp           null,
constraint PK_UNIDAD primary key  (IDUNIDAD)
)
go


alter table ADQUISICION
  add constraint FK_ADQUISIC_REFERENCE_PROVEEDO foreign key
(IDPROVEEDOR)
    references PROVEEDOR (IDPROVEEDOR)
go


alter table CONTROL
  add constraint FK_CONTROL_REFERENCE_EMPLEADO foreign key
(IDEMPLEADO)
    references EMPLEADO (IDEMPLEADO)
go


alter table DETALLE_ADQUSICION
  add constraint FK_DETALLE__REFERENCE_ADQUISIC foreign key
(IDADQUISICION)
    references ADQUISICION (IDADQUISICION)
go


alter table DETALLE_ADQUSICION
  add constraint FK_DETALLE__REFERENCE_PRODUCTO foreign key
(IDPRODUCTO)
    references PRODUCTO (IDPRODUCTO)
go


alter table DPEDIDO
  add constraint FK_DPEDIDO_REFERENCE_PEDIDO foreign key (IDPEDIDO)
    references PEDIDO (IDPEDIDO)
go


alter table DPEDIDO
  add constraint FK_DPEDIDO_REFERENCE_DMENU foreign key (IDDMENU)
    references DMENU (IDDMENU)
go
```

```
alter table LOG
  add constraint FK_LOG_REFERENCE_EMPLEADO foreign key (IDEMPLEADO)
    references EMPLEADO (IDEMPLEADO)
go


alter table PEDIDO
  add constraint FK_PEDIDO_REFERENCE_MESA foreign key (IDMESA, CODMESA)
    references MESA (IDMESA, CODMESA)
go


alter table PEDIDO
  add constraint FK_PEDIDO_REFERENCE_EMPLEADO foreign key (IDEMPLEADO)
    references EMPLEADO (IDEMPLEADO)
go


alter table PEDIDO
  add constraint FK_PEDIDO_REFERENCE_CLIENTE foreign key (IDCLIENTE)
    references CLIENTE (IDCLIENTE)
go


alter table PRODUCTO
  add constraint FK_PRODUCTO_REFERENCE_UNIDAD foreign key (IDUNIDAD)
    references UNIDAD (IDUNIDAD)
go


alter table PRODUCTO
  add constraint FK_PRODUCTO_REFERENCE_PROCEDEN foreign key
(IDPROCEDENCIA)
    references PROCEDENCIA (IDPROCEDENCIA)
go
```

7.2 VISUAL BASIC

Strip del Modulo de Adquisición

```
Option Explicit
Dim Tipo As Integer
Dim Var
Dim nCol As Integer
Private Sub Form_Activate()
Set Formulario = Me
centrarFormulario (1)
```

```
End Sub



Private Sub Form_Load()
Me.Width = 9510
Me.Height = 6270
Me.DTP1.Value = fechaServidor
Me.MSHFlexGrid1.Width = Me.Width - 600
Me.MSHFlexGrid1.FormatString = ">Nº  |Cod. Producto            |Descripción
" & _
"|>Cantidad|>P/U         |I.V.A |>Total          "
Call checkMSH(Me.MSHFlexGrid1, 5)
End Sub

Private Sub MSHFlexGrid1_Click()
nCol = Me.MSHFlexGrid1.MouseCol
Me.MSHFlexGrid1.Col = nCol

End Sub

Private Sub MSHFlexGrid1_DblClick()
Tipo = 3

Select Case Me.MSHFlexGrid1.Col
 Case 1, 3
  Tipo = 1
 Case 4
  Tipo = 2
End Select
If Me.MSHFlexGrid1.Col = 1 Or Me.MSHFlexGrid1.Col = 3 Or Me.MSHFlexGrid1.Col
= 4 Then
 Call EditorTexto(Me.MSHFlexGrid1, 30)
ElseIf Me.MSHFlexGrid1.Col = 5 Then
 Call editorCheck(Me.MSHFlexGrid1)
Else
Call error("Campo no editable")
End If
End Sub



Private Sub MSHFlexGrid1_KeyPress(KeyAscii As Integer)
If KeyAscii = 13 Or KeyAscii = 32 Then
 Call MSHFlexGrid1_DblClick
End If
End Sub

Private Sub MSHFlexGrid1_SelChange()
```

```
If nCol = 5 Then
 Call sumatoria
End If
End Sub


Private Sub Text1_Change()
Me.MSHFlexGrid1.Text = Me.Text1.Text
End Sub

Private Sub Text1_GotFocus()
SendKeys "{HOME}+{END}"
End Sub

Private Sub Text1_KeyPress(KeyAscii As Integer)
 KeyAscii = KeyPress(Me.Text1, KeyAscii, Tipo)
End Sub

Private Sub Text1_KeyUp(KeyCode As Integer, Shift As Integer)
If KeyCode = 45 Then
  variable = ""
  Me.Text1.Text = ""
  frmBuscar.cargaDatos (2)
  frmBuscar.Show vbModal
  If variable <> "" Then
   Me.Text1.Text = variable
   Call Text1_LostFocus
  End If
End If
End Sub


Private Sub Text1_LostFocus()
Dim ado As Adodc
Dim Temp As Integer
If nCol = 1 Then
       Set ado = selectDatos("SELECT * FROM PRODUCTO WHERE
CODPRODUCTO='" & Me.Text1.Text & "'")
       Temp = Me.MSHFlexGrid1.Col
       If ado.Recordset.RecordCount > 0 Then
        Me.MSHFlexGrid1.Col = 2
        Me.MSHFlexGrid1.Text = ado.Recordset.Fields("DESCRIPCIONPRODUCTO") '
        'Me.MSHFlexGrid1.Col = 4
        'Me.MSHFlexGrid1.Text
       ElseIf Me.Text1.Text <> "" Then
        MsgBox "EL CODIGO DE PRODUCTO INGRESADO NO EXISTE",
vbExclamation, rotuloMsg
        Me.MSHFlexGrid1.Col = 1
        Me.MSHFlexGrid1.Text = ""
```

```
        Me.Text1.Text = ""
       End If
       Me.MSHFlexGrid1.Col = Temp
ElseIf nCol = 3 Or nCol = 4 Or nCol = 5 Then
 Call sumatoria
End If
Me.Text1.Visible = False
End Sub

Private Sub Text2_KeyPress(KeyAscii As Integer)
If KeyAscii = 13 Then
  SendKeys "{TAB}"
ElseIf KeyAscii = 27 Then
  Text2.Text = ""
Else
  KeyAscii = Validar(KeyAscii, 3)
End If
End Sub


Private Sub Text3_KeyPress(KeyAscii As Integer)
If KeyAscii = 13 Then
  SendKeys "{TAB}"
ElseIf KeyAscii = 27 Then
  Text3.Text = ""
Else
  KeyAscii = Validar(KeyAscii, 3)
End If
End Sub


Private Sub Text4_KeyPress(KeyAscii As Integer)
If KeyAscii = 13 Then
  SendKeys "{TAB}"
ElseIf KeyAscii = 27 Then
  Text4.Text = ""
Else
  KeyAscii = Validar(KeyAscii, 3)
End If
End Sub


Private Sub Text5_KeyPress(KeyAscii As Integer)
If KeyAscii = 13 Then
  SendKeys "{TAB}"
ElseIf KeyAscii = 27 Then
  Text5.Text = ""
Else
  KeyAscii = Validar(KeyAscii, 3)
```

```
End If
End Sub


Private Sub Text5_KeyUp(KeyCode As Integer, Shift As Integer)
If KeyCode = 45 Then
  variable = ""
  Me.Text1.Text = ""
  frmBuscar.cargaDatos (1)
  frmBuscar.Show vbModal
  If variable <> "" Then
   Me.Text5.Text = variable
   Call Text5_LostFocus
  End If
End If
End Sub

Private Sub Text5_LostFocus()
On Error GoTo salto
Dim ado As Adodc
If Trim(Me.Text5.Text) = "" Then
 Me.Text5.Tag = 1
 Me.Text6.Text = ""
 Exit Sub
End If
Set ado = selectDatos("SELECT * FROM PROVEEDOR WHERE RUCPROVEEDOR='"
& Me.Text5.Text & "'")
If ado.Recordset.RecordCount > 0 Then
 Me.Text5.Tag = ado.Recordset.Fields("IDPROVEEDOR")
 Me.Text6.Text = ado.Recordset.Fields("NOMBREPROVEEDOR")
Else
 MsgBox "EL CODIGO DE PROVEEDOR  NO EXISTE", vbExclamation, rotuloMsg
 Me.Text5.Text = ""
 Me.Text5.SetFocus
End If
Exit Sub
salto:
MsgBox Err.Number & "   " & Err.Description, vbExclamation, rotuloMsg
End Sub

Private Sub Text6_KeyPress(KeyAscii As Integer)
If KeyAscii = 13 Then
  SendKeys "{TAB}"
ElseIf KeyAscii = 27 Then
  Text6.Text = ""
Else
  KeyAscii = Validar(KeyAscii, 3)
End If
End Sub
```

```
''''''''''''''''''
'METODOS
''''''''''''''''''

Public Sub grabar()
 Dim X As Integer
 Dim Query As String
 If Trim(Me.Text2.Text) = "" Or Trim(Me.Text3.Text) = "" Or Trim(Me.Text4.Text) = ""
Or Trim(Me.Text5.Text) = "" Or Trim(Me.Text6.Text) = "" Then
  MsgBox "LA INFORMACION DE LA ADQUISICION ESTA INCOMPLETA",
vbInformation, rotuloMsg
  Me.Text2.SetFocus
  Exit Sub
 End If
 Query = "INSERT INTO ADQUISICION VALUES(" & Me.Text5.Text & ")"

 For X = 1 To Me.MSHFlexGrid1.Rows - 1

 Next X
End Sub

Public Sub Nuevo()
If Me.Frame1.Enabled = False Then
  Me.Frame1.Enabled = True
  Me.Text2.SetFocus
Else
 Me.MSHFlexGrid1.AddItem ""
 Call checkMSH(Me.MSHFlexGrid1, 5)
End If
End Sub

Public Sub sumatoria()
      Dim X As Integer
      Dim ivaAdquisicion, subTotal, Total, ivaItem As Double
      Dim PU, cantidad
      ivaAdquisicion = 0
      subTotal = 0
      Total = 0
      ivaItem = 0
      For X = 1 To Me.MSHFlexGrid1.Rows - 1
         PU = convertirComa(devolverMSH(Me.MSHFlexGrid1, 3, X))
         cantidad = convertirComa(devolverMSH(Me.MSHFlexGrid1, 4, X))
         If Trim(PU) = "" Then
          PU = 0
         End If
         If Trim(cantidad) = "" Then
          cantidad = 0
```

```
          End If
        Me.MSHFlexGrid1.Row = X
        Me.MSHFlexGrid1.Col = 6
        Me.MSHFlexGrid1.Text = PU * cantidad
        subTotal = subTotal + devolverMSH(Me.MSHFlexGrid1, 6, X)
        If Trim(devolverMSH(Me.MSHFlexGrid1, 5, X)) = 1 Then
         ivaItem = devolverMSH(Me.MSHFlexGrid1, 6, X) * (Iva / 100)
         ivaAdquisicion = ivaAdquisicion + ivaItem
        End If
       Next X
       Label7.Caption = subTotal
       Label10.Caption = ivaAdquisicion
       Label12.Caption = subTotal + ivaAdquisicion
End Sub
```

Strip del Formulario de productos

```
Option Explicit
Dim Tipo As Integer
Dim Var
Dim nRow As Integer

Private Sub Combo1_Click()
Me.MSHFlexGrid1.Text = Me.Combo1.Text
If Me.Combo1.Tag = 1 Then
 Dim ado As Adodc
 Set ado = selectDatos("SELECT * FROM UNIDAD WHERE
DESCRIPCIONUNIDAD='" & Me.Combo1.Text & "'")
 If ado.Recordset.RecordCount > 0 Then
  Me.MSHFlexGrid1.Col = 5
  Me.MSHFlexGrid1.Text = ado.Recordset.Fields("DESCRIPCIONCONVERSION")
 End If
End If
End Sub

Private Sub Combo1_LostFocus()
Me.Combo1.Visible = False
End Sub

Private Sub Form_Activate()
Set Formulario = Me
centrarFormulario (1)
End Sub

Private Sub Form_Load()
Me.Width = 13000
Me.MSHFlexGrid1.Width = Me.Width - 300
```

```
Me.MSHFlexGrid1.FormatString = ">Nº     |Codigo              |Descripción
|I.V.A" & _
"|Unidad                |Resultante  |Procedencia            |>Stock Min.|>Stock Max.|>Stock
Actual" & _
"|% Desc.  |>P.V.P     "
Me.MSHFlexGrid1.RowHeight(Me.MSHFlexGrid1.Rows - 1) = 315
Call Me.cargaDatos
Call formatoCelda(Me.MSHFlexGrid1, 10, 1)
Call formatoCelda(Me.MSHFlexGrid1, 11, 1)
Call numerarMSH(Me.MSHFlexGrid1)
Call formatearGrid
Call Form_Activate
End Sub


Private Sub MSHFlexGrid1_Click()
nRow = Me.MSHFlexGrid1.MouseRow
End Sub

Private Sub MSHFlexGrid1_KeyPress(KeyAscii As Integer)
If KeyAscii = 13 Or KeyAscii = 32 Then
 Call MSHFlexGrid1_DblClick
End If
End Sub

Private Sub MSHFlexGrid1_DblClick()
If Me.MSHFlexGrid1.Col = 3 Then
 Call editorCheck(Me.MSHFlexGrid1)
ElseIf Me.MSHFlexGrid1.Col = 4 Then
 Call editorCombo(Me.MSHFlexGrid1, 1)
'ElseIf Me.MSHFlexGrid1.Col = 5 Then
' CALL editorCombo(Me.MSHFlexGrid1, 1)
ElseIf Me.MSHFlexGrid1.Col = 1 Then
 Tipo = 1
 Call EditorTexto(Me.MSHFlexGrid1, 30)
ElseIf Me.MSHFlexGrid1.Col = 2 Then
 Tipo = 3
 Call EditorTexto(Me.MSHFlexGrid1, 250)
ElseIf Me.MSHFlexGrid1.Col = 6 Then
 Call editorCombo(Me.MSHFlexGrid1, 2)
ElseIf Me.MSHFlexGrid1.Col = 7 Or Me.MSHFlexGrid1.Col = 8 Or
Me.MSHFlexGrid1.Col > 9 Then
 Tipo = 2
 Call EditorTexto(Me.MSHFlexGrid1, 8)
End If
End Sub

Private Sub Text1_Change()
Me.MSHFlexGrid1.Text = Me.Text1.Text
```

```
End Sub

Private Sub Text1_GotFocus()
SendKeys "{home}+{end}"
End Sub
Private Sub Text1_LostFocus()
Me.Text1.Visible = False
End Sub

Private Sub Text1_KeyPress(KeyAscii As Integer)
  KeyAscii = KeyPress(Me.Text1, KeyAscii, Tipo)
End Sub
''''''''''''''''''
'METODOS
''''''''''''''''''
Public Sub Nuevo()
Me.MSHFlexGrid1.AddItem ""
Me.MSHFlexGrid1.RowHeight(Me.MSHFlexGrid1.Rows - 1) = 315
Call checkMSH(Me.MSHFlexGrid1, 3)
End Sub

Public Sub cargaDatos()
Dim Query As String
Query = "select
codproducto,descripcionproducto,iva,descripcionunidad,descripcionconversion,descripcion
procedencia,stockmin,stockmax,stockactual,porcentajedescuento,pvp from producto
pd,unidad un,procedencia pr where pd.idunidad=un.idunidad and
pd.idprocedencia=pr.idprocedencia"
'Call selectDatos(Query)
Call llenarMSH(Me.MSHFlexGrid1, Query)
Call checkMSH(Me.MSHFlexGrid1, 3)
End Sub

Public Sub grabar()
Dim codUnidad, codProcedencia As Integer
Dim X As Integer
Dim Query As String
 For X = 1 To Me.MSHFlexGrid1.Rows - 1
 If devolverMSH(Me.MSHFlexGrid1, 1, X) <> "" Then
    codUnidad = devolverCombo("UNIDAD", "DESCRIPCIONUNIDAD",
devolverMSH(Me.MSHFlexGrid1, 4, X))
    codProcedencia = devolverCombo("PROCEDENCIA",
"DESCRIPCIONPROCEDENCIA", devolverMSH(Me.MSHFlexGrid1, 6, X))
   If selectDatos("SELECT * FROM PRODUCTO WHERE CODPRODUCTO='" &
devolverMSH(Me.MSHFlexGrid1, 1, X) & "'").Recordset.RecordCount > 0 Then
    Query = "UPDATE PRODUCTO SET DESCRIPCIONPRODUCTO='" &
devolverMSH(Me.MSHFlexGrid1, 2, X) & "', IVA=" &
devolverMSH(Me.MSHFlexGrid1, 3, X) & ", IDUNIDAD=" & codUnidad &
",IDPROCEDENCIA=" & codProcedencia & ",STOCKMIN=" &
```

```
convertirComa(devolverMSH(Me.MSHFlexGrid1, 7, X)) & ",STOCKMAX=" &
convertirComa(devolverMSH(Me.MSHFlexGrid1, 8, X)) & ",STOCKACTUAL=" &
convertirComa(devolverMSH(Me.MSHFlexGrid1, 9, X)) &
",PORCENTAJEDESCUENTO=" & convertirComa(devolverMSH(Me.MSHFlexGrid1,
10, X)) & ",PVP=" & convertirComa(devolverMSH(Me.MSHFlexGrid1, 11, X)) & "
WHERE CODPRODUCTO='" & devolverMSH(Me.MSHFlexGrid1, 1, X) & "'"
     ejecucion (Query)
    Else
     Query = "INSERT INTO PRODUCTO VALUES(" &
devolverMSH(Me.MSHFlexGrid1, 1, X) & ",'" & devolverMSH(Me.MSHFlexGrid1, 2,
X) & "'," & devolverMSH(Me.MSHFlexGrid1, 3, X) & "," & codUnidad & "," &
codProcedencia & "," & devolverMSH(Me.MSHFlexGrid1, 7, X) & "," &
convertirComa(devolverMSH(Me.MSHFlexGrid1, 8, X)) & ",0," &
convertirComa(devolverMSH(Me.MSHFlexGrid1, 10, X)) & "," &
convertirComa(devolverMSH(Me.MSHFlexGrid1, 11, X)) & ",1,NULL)"
     ejecucion (Query)
    End If
 End If
 Next X
End Sub
Public Sub BORRAR()
 If nRow <> 0 Then
   If MsgBox("DESE BORRAR EL PRODUCTO  " & devolverMSH(Me.MSHFlexGrid1,
1, nRow) & "   " & devolverMSH(Me.MSHFlexGrid1, 2, nRow) & "?", vbQuestion +
vbYesNo, rotuloMsg) = vbYes Then
    If ejecucion("DELETE FROM PRODUCTO WHERE CODPRODUCTO='" &
devolverMSH(Me.MSHFlexGrid1, 1, nRow) & "'") = True Then
     MsgBox "EL PRODUCTO FUE ELIMINADO", vbInformation, rotuloMsg
     Unload Me
     Load Me
    Else
     MsgBox "EL PRODUCTO NO PUEDE SER ELIMINADO", vbExclamation,
rotuloMsg
    End If
   End If
 End If
 nRow = 0
End Sub

Public Sub formatearGrid()
Dim X As Integer
For X = 1 To Me.MSHFlexGrid1.Rows - 1
 Me.MSHFlexGrid1.Row = X
 Me.MSHFlexGrid1.RowHeight(Me.MSHFlexGrid1.Rows - 1) = 315
Next X
End Sub
```

PRUEBAS

5 PRUEBAS

5.1 PRUEBAS DE UNIDAD

PLAN DE PRUEBA

Objetivo: Esta prueba tiene como objetivo verificar el buen funcionamiento de las interfaces.
Elemento del sistema a probar: Interfaz para ingreso de nuevos empleados (formulario empleado).
Características a probar: Funcionalidades que presenta la interfaz, verificando la validación de información que maneja la misma

DESCRIPCIÓN DE LA PRUEBA

Actividades: Compilar el proyecto y ejecutar la interfaz.
Ingresar valores a los campos de la interfaz.
Comprobar la validación de los mismos
Elaborar informe
Entorno donde se realizo la prueba: Máquina 1, Sistema Operativo Windows XP
Tiempo estimado: 10 minutos
Responsable: Osmany Aguilar

EJECUCIÓN DE LA PRUEBA

Se ejecuta la prueba, siguiendo los parámetros indicados en el punto anterior

RESULTADOS DE LA PRUEBA

Resultado: Prueba no superada.
Observaciones: La interfaz permite el ingreso de valores numéricos, donde sólo se debe permitir el ingreso de caracteres. (formulario empleado)

Conclusiones: Aplicar la respectiva validación al campo mencionado y ejecutar nuevamente la prueba.
Aprobado por: Osmany Aguilar

5.2 PRUEBAS DE INTEGRACIÓN

PLAN DE PRUEBA

Objetivo: Esta prueba tiene como objetivo verificar el correcto funcionamiento en conjunto de las interfaces con los módulos que van a manejar.
Elemento del sistema a probar: Interfaz para ingreso de nuevos empleados (formulario empleado) y el módulo para administrar personal.
Características a probar: Verificar la funcionalidad que se maneja en la interfaz sea la misma que está ejecutando el módulo.

DESCRIPCIÓN DE LA PRUEBA

Actividades: Compilar el proyecto y ejecutar la interfaz.
Ejecutar funcionalidades de la interfaz.
Elaborar informe
Entorno donde se realizo la prueba: Máquina 1, Sistema Operativo Windows XP
Tiempo estimado: 20 minutos
Responsable: David Guijarro

EJECUCIÓN DE LA PRUEBA

Se ejecuta la prueba, siguiendo los parámetros indicados en el punto anterior

RESULTADOS DE LA PRUEBA

Resultado: Prueba no superada.
Observaciones: La funcionalidad que presenta la interfaz para modificar los datos del empleado, no está habilitada.

Conclusiones: Verificar la propiedad enable y ejecutar nuevamente la prueba.
Aprobado por: David Guijarro
Dentro de esta prueba, se aplicó la integración incremental.

5.3 PRUEBAS DEL SISTEMA

PLAN DE PRUEBA

Objetivo: Esta prueba tiene como objetivo que el sistema está cumpliendo todos los propósitos que se fueron especificados.
Elemento del sistema a probar: Sistema del Restaurante Metro Café
Características a probar: Verificar la funcionalidad de todo el sistema demostrando que se está cubriendo todos los requerimientos.

DESCRIPCIÓN DE LA PRUEBA

Actividades: Compilar el proyecto y ejecutarlo.
Ejecutar funcionalidades del proyecto
Elaborar informe
Entorno donde se realizo la prueba: Máquina 1, Sistema Operativo Windows XP
Tiempo estimado: 2 días
Responsable: Osmany Aguilar

EJECUCIÓN DE LA PRUEBA

Se ejecuta la prueba, siguiendo los parámetros indicados en el punto anterior

RESULTADOS DE LA PRUEBA

Resultado: Prueba superada satisfactoriamente.
Observaciones: Ninguna
Conclusiones: Ninguna.

Aprobado por: Osmany Aguilar

5.4 PRUEBAS DE ACEPTACIÓN

PLAN DE PRUEBA

Objetivo: Esta prueba tiene como objetivo comprobar que el sistema está listo para ser implantado.
Elemento del sistema a probar: Sistema del Restaurante Metro Café.
Características a probar: Verificar la funcionalidad de todo el sistema demostrando.

DESCRIPCIÓN DE LA PRUEBA

Actividades: Compilar el proyecto y ejecutarlo.
Ejecutar funcionalidades del proyecto
Elaborar informe
Entorno donde se realizo la prueba: Máquina 1, Sistema Operativo Windows xp
Tiempo estimado: 2 días
Responsable: David Guijarro, Osmany Aguilar.

EJECUCIÓN DE LA PRUEBA

Se ejecuta la prueba, siguiendo los parámetros indicados en el punto anterior

RESULTADOS DE LA PRUEBA

Resultado: Prueba superada satisfactoriamente.
Observaciones: Ninguna
Conclusiones: Ninguna.
Aprobado por: David Guijarro, Osmani Aguilar