ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

DESARROLLO DE UNA INTERFAZ BIOMÉTRICA BASADA EN LA LECTURA DE HUELLAS DACTILARES PARA AUTENTICACIÓN DE USUARIOS EN UN CAJERO AUTOMÁTICO

TOMO I

PROYECTO PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRÓNICA Y REDES DE INFORMACIÓN

JUAN FRANCISCO SALCEDO POLANCO

pancho_salcedo85@hotmail.com

PAOLA CECILIA SEMPÉRTEGUI JÁCOME

psempertegui@hotmail.es

DIRECTOR: Ing. PABLO HIDALGO LASCANO
phidalgo@ieee.org

Quito, Agosto 2009

DECLARACIÓN

Nosotros, Juan Francisco Salcedo Polanco y Paola Cecilia Sempértegui Jácome, declaramos bajo juramento que el trabajo aquí descrito es de nuestra autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que hemos consultado las referencias bibliográficas que se incluyen en este documento.

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

Juan Francisco Salcedo Polanco Paola Cecilia Sempértegui Jácome

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por Juan Francisco Salcedo Polanco y Paola Cecilia Sempértegui Jácome, bajo mi supervisión.

Ing. PABLO HIDALGO LASCANO
DIRECTOR DEL PROYECTO

DEDICATORIA

Porque el camino fue arduo y complicado, con trabas e inconvenientes difíciles de superar; porque la lucha fue intensa pero gracias a Dios encontré la fuerza para salir adelante por mis hermanos Anita María y Fernando Esteban que son mi inspiración en la lucha del día a día; con el amor, paciencia y sacrificio de mis padres Susana y Fernando que los llevo en mi corazón y son mi mayor apoyo, modelo y fortaleza, con el cariño de mis abuelitas Martha y Yoquita que siempre me dieron los ánimos para continuar; con la preocupación de mi tía Liliana que se convirtió en mi consejera; y con ejemplo de mi padrino Eugenio que es uno de los modelos en mi vida… Hoy les dedico el presente proyecto de titulación como gratitud por darme las herramientas, la inspiración y la fuerza para no claudicar en mis esfuerzos para finalizar esta travesía esperando jamás decepcionarlos y estar junto a ustedes cuando me necesiten. Siempre permanecerán en mí, y que Dios los bendiga por la eternidad.

A la mujer que tocó mi vida y me regaló su silencio…

Juan Francisco

DEDICATORIA

Este proyecto quiero dedicarlo a Dios, por ser el escudo alrededor de mí, por ser quien levanta mi cabeza, por su inmenso, infinito y eterno amor. También se lo dedico a mi Esteban por ser la alegría más grande de mi vida.

Paola

AGRADECIMIENTO

Los amigos son la familia que uno ha decidido escoger. Yo cuento con la bendición de mi Dios a quien estoy agradecido por todo lo que me ha dado; por su amistad y palabras, por el regalo de la vida en más de una ocasión, por brindarme una familia que son mis amigos y que nunca me han dejado solo, por mis amigos que son como mi familia, por la inspiración y la fuerza para continuar sin importar el reto que se presente.

Mi agradecimiento de corazón por acompañarme en esta travesía y regalarme momentos invaluables a mis tías Jacqueline, Elizabeth, Ximena, Nicole y Miriam; a mis tíos Marcelo P., Juan, Ariel, Wladimir, Marcelo A., Diego, Enrique y Fabián; a mis hermanas de corazón Andrea y Pamela; a primos Roberto, Sarita, José Antonio, Álvaro, Holger, Gabriela y Juan José; a mi china bella Ana Paula; a mis amigas Estefy, Natasha, Paulina, María José y Marielisa; a mis amigos Carlos Q., Santiago, Daniel, Miguel, Patricio, Carlos E., Xavier y Andrés; a mis profesores Pablo H., Fernando F., Ramiro P., Erick G., y Ángel V.; hoy no estaría aquí si no fuera por ustedes que han aportado en mi vida grandes enseñanzas.

Un agradecimiento especial a mi compañera Paola que ha tenido la paciencia necesaria para aguantarme durante todo este tiempo.

Juan Francisco

AGRADECIMIENTO

Todo lo que he logrado a lo largo de mi vida se lo debo a Dios. Quiero agradecer a mi familia por apoyarme, por el valor que siempre me han dado para seguir adelante, por haber estado a mi lado en los buenos y malos momentos. A mi compañero de proyecto de titulación por su paciencia y sus conocimientos. Gracias a mis madres por su apoyo incondicional y confianza.

Gracias Señor por tus promesas porque Tú eres quien peleas mis batallas.

Paola

ÍNDICE

TOMO II

ANEXO A: ESTÁNDAR DE NIST

ANEXO B: ESTRUCTURAS DE CONTROL Y BUCLES DE VISUAL BASIC

ANEXO C: MÉTODOS ESENCIALES DEL SDK GRIAULE BIOMETRICS

ANEXO D: NITGEN FINGERPRINT READER

ANEXO E: CÓDIGO DE LA APLICACIÓN

ANEXO F: DICCIONARIO DE DATOS

ANEXO G: SCRIPT DE LA BASE DE DATOS

ANEXO H: MANUAL DE USUARIO

ÍNDICE DE FIGURAS

TOMO I

TOMO II

ÍNDICE DE TABLAS

TOMO I

RESUMEN

El presente proyecto de titulación se ha estructurado en cuatro capítulos más ocho anexos que contienen tanto la sustentación teórica necesaria para el proyecto, así como el diseño, desarrollo y pruebas de la aplicación. En él también se han incorporado las conclusiones del proyecto realizado y recomendaciones para su posible ampliación.

En el primer capítulo se tienen todas las bases teóricas empleadas para la realización del presente proyecto de titulación. Se puede encontrar la teoría del funcionamiento y estructura de los cajeros automáticos. Además se presenta un estudio de los métodos biométricos utilizados para la autenticación de personas, haciendo énfasis en la huella dactilar que es el método que se empleará en este proyecto (el estándar empleado se encuentra en el Anexo A). Este capítulo también incluye la revisión de los conceptos de Base de Datos, Programación Orientada a Objetos y el modelado UML, los cuales representaron las bases para el diseño y desarrollo de la aplicación. Contiene además las especificaciones del kit de desarrollo empleado (descrito mayormente en el Anexo C), el tipo de lectores de huellas existentes (las especificaciones del lector seleccionado se encuentra en el Anexo D), las etapas que permiten dimensionar el servidor que alojará la aplicación y los conceptos básicos de seguridad informática empleados.

El diseño, desarrollo e implementación de la aplicación se encuentran en el segundo capítulo. En éste se pone en práctica el desarrollo completo de la metodología de modelado de aplicaciones basado en UML, sustentado en el primer capítulo. Además se emplean los conceptos de modelado de base de datos para el diseño e implementación de la base de datos que almacenará las huellas, la información del cliente y las transacciones realizadas. En esta etapa del proyecto se realiza la implementación de la aplicación y como producto se tienen, los anexos E, F y G. Adicionalmente, en este capítulo se realiza el dimensionamiento del servidor requerido, el cálculo de espacio en disco de la base de datos y los detalles de la implementación de la aplicación.

El tercer capítulo describe las pruebas que se realizaron y los escenarios en las que se ejecutaron para comprobar el correcto funcionamiento de la aplicación y las pruebas para validar el grado de confiabilidad y respuesta del producto desarrollado. Se describen también los resultados que se obtuvieron en cada prueba y se presenta el análisis de los mismos.

Finalmente, el cuarto capítulo contiene las conclusiones tanto de las pruebas realizadas así como las conclusiones del proyecto de titulación. Se tienen también las recomendaciones que se realiza para ampliar el presente proyecto y poder brindar nuevas funcionalidades a la aplicación.

PRESENTACIÓN

En los últimos años se ha registrado un incremento en el uso de los cajeros automáticos por los servicios que éstos ofrecen. Los cajeros han facilitado considerablemente la vida de sus usuarios por el corto tiempo que toma realizar una transacción en el mismo y por la gran gama de posibilidades que ofrecen, debido a su interrelación con diferentes tipos de servicios como retiro de dinero, pago de servicios, transferencias, etc. Por este motivo los cajeros automáticos se los encuentra en casi cualquier lugar y tienen una gran aceptación a nivel de usuarios.

Generalmente, el medio por el cual una persona se autentica en un cajero automático es a través de una tarjeta de banda magnética, que junto con una clave (normalmente de 4 dígitos y también conocido como PIN) verifican la identidad de la persona. Este tipo de autenticación no garantiza que la persona que ingresa la clave es quien dice ser; además este número puede ser olvidado, compartido o robado.

En nuestro país aunque el uso de los cajeros automáticos se ha masificado, existen grupos poblaciones que no pueden acceder a estos servicios por su analfabetismo; quedando aislados de las ventajas que brinda el mundo tecnológico actual y las facilidades de los servicios provistos por los cajeros.

El presente proyecto de titulación, presenta la implementación de una interfaz biométrica basada en la lectura de huellas dactilares como medio de autenticación en un cajero automático, como una alternativa a la autenticación mediante el PIN. El proyecto está enfocado a proponer una solución para estos dos hechos: al aumentar la seguridad en los cajeros automáticos, y crear un puente entre las personas analfabetas y el acceso a los servicios de los cajeros automáticos (específicamente al cobro del bono).

El primer aspecto se lo cumple ya que la autenticación de una persona mediante su huella dactilar permite garantizar que la persona que desea realizar una transacción es quien dice ser, ya que los rasgos de la huella dactilar de cada

persona son únicos, y junto con la tarjeta del usuario permitirán verificar su identidad.

En cambio el segundo aspecto es posible ya que una persona analfabeta ya no necesita recordar una contraseña (o un su defecto tener que aprenderla) para poder realizar una transacción, sino que es autenticada mediante su huella y el ingreso de la tarjeta. Mediante esta posibilidad el cobro del bono de solidaridad puede llegar a un mayor número de lugares, evitará colas de personas en las ventanillas y garantizará la posibilidad que todas las personas puedan cobrar el bono de desarrollo.

CAPÍTULO 1

MARCO TEÓRICO

1.1 CAJEROS AUTOMÁTICOS [1]

El primer cajero automático a nivel mundial apareció en el año de 1939, su funcionamiento era bastante rústico ya que funcionaba de forma mecánica y solo podía entregar dinero. Este cajero fue creado para reducir las largas colas para retiro de dinero al interior de los bancos. Debido a su funcionamiento mecánico las transacciones de retiro de dinero tardaban mucho tiempo, motivo por el cual su uso no fue muy acogido por los usuarios de aquella época.

En Londres, dos décadas después aparece el primer cajero automático electrónico; este cajero mantenía en su memoria la información del cliente (número de identificación y PIN). Cuando un cliente deseaba retirar dinero depositaba un *voucher* (comprobante o recibo) en el cajero indicando el valor a retirar, el cajero validaba la información del cliente con la que tenía almacenada y si la información era válida permitía el retiro de dinero. Fue solamente hasta el año 1968 en que se logró implementar el primer cajero automático en red.

Este cajero se encontraba conectado a un equipo encargado de procesar la transacción. Un año después se utilizó por primera vez la tarjeta con banda magnética para identificar al usuario; y en el año 1971 se introdujo el primer cajero con todas las funcionalidades similares a las encontradas en la actualidad. En los años recientes, los cajeros automáticos han evolucionado para prestar servicios adicionales (como por ejemplo el pago de impuestos, pago de servicios básicos, recargas para telefonía móvil, depósitos de cheques, entre otros).

1.1.1 CONCEPTOS

- *Cajero Automático:* También conocido como ATM por sus siglas en inglés (*Automated Teller Machine*), es un dispositivo computarizado que permite que un cliente pueda realizar transacciones financieras, generalmente mediante el uso de una tarjeta plástica que contiene un número único y cierta información de seguridad utilizada para identificar al usuario.

- *PIN:* Número normalmente de cuatro dígitos ingresado por un usuario en un cajero automático junto al número único de tarjeta para que el cliente sea autenticado.
- *Red Interbancaria de Cajeros***:** Consiste en una red de varios cajeros automáticos de diferentes instituciones financieras y que sin importar la institución a la que pertenece el cajero se puede hacer transacciones en él.

1.1.2 ESQUEMA DE FUNCIONAMIENTO

Un cajero automático representa un terminal de datos conectado a un Host Central, al que pueden estar conectados varios cajeros. El Host Central se encarga del procesamiento de los mensajes provenientes del cajero y retorna mensajes con respuestas a las solicitudes o con información de administración. Las transacciones no son más que el intercambio de mensajes entre el host central y el cajero; y son ejemplificadas en el diagrama de la figura 1.1. El host central se encuentra a su vez conectado a un servidor de aplicación y de servicios; este servidor es el encargado de ofrecer y procesar todos los servicios que se tienen en el cajero automático.

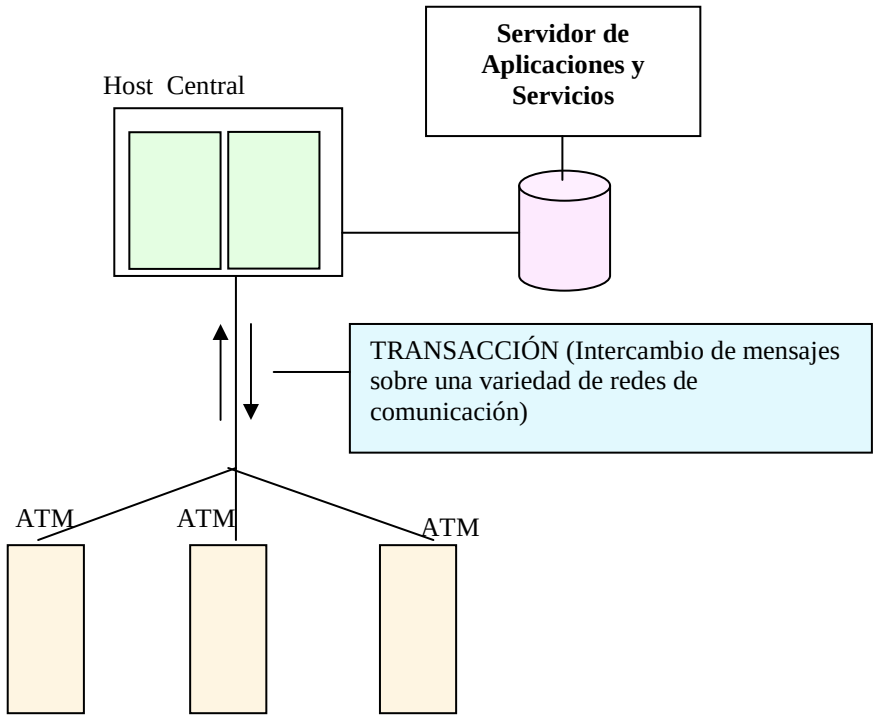


Figura 1. 1: Esquema de Funcionamiento de los Cajeros Automáticos

1.1.2.1 Formato de los Mensajes de Transacción [2]

El sistema de mensajería entre un ATM y el Host Central para transacciones financieras se encuentra especificado en la norma ISO 8583; esta norma indica el formato general para el intercambio de mensajes relacionados a actividades financieras o autorización de transacciones.

La norma especifica que el mensaje debe constar de los tres campos que se indican en la figura 1.2.

Figura 1. 2: Formato Mensaje de Transacciones Financieras (ISO 8583)

- *Identificador de Tipo de Mensaje:* Este campo representa la clase y la función del mensaje mediante 4 dígitos decimales. Los valores posibles que puede tener este campo se encuentran en la tabla 1.1.

VALOR	**DESCRIPCIÓN**
02XX	Mensajes de Transacción Financiera
04XX	Mensaje Reverso
08XX	Mensaje de Administración de Red

Tabla 1. 1: Valor y Descripción del campo Identificador de Mensajes

Para los mensajes de transacción financiera el valor de los dos últimos dígitos dependen del tipo de transacción. Si el mensaje es una petición se usa el valor 0200 y si es una respuesta a una petición se usa el valor 0210.

Un mensaje reverso es un mensaje de alerta o de información. Estos mensajes son enviados cuando por ejemplo uno de los terminales no recibe la respuesta a una petición después de cierto tiempo. Estos mensajes pueden darse de forma interactiva o no interactiva. La interactividad indica que el terminal que envía el mensaje espera que sea procesado inmediatamente por el receptor y sea contestado con una respuesta. De esta manera en la tabla 1.2 se tienen los posibles valores para mensajes reversos:

VALOR	DETALLE
0400	Petición de transacción reversa interactiva.
0410	Respuesta de transacción reversa interactiva.
0420	Petición de transacción reversa no interactiva.
0430	Respuesta de transacción reversa no interactiva.
0401	Mensajes reversos repetitivos interactivos.
0421	Mensajes reversos repetitivos no interactivos.

Tabla 1. 2: Campo Identificador de Mensajes para Mensajes Reversos

Finalmente, los mensajes 08XX son utilizados para determinar el estado de la comunicación entre los terminales. En el caso particular del sistema de cajeros automáticos hacen referencia a la comunicación entre el ATM y el Host Central.

- *Mapa de Bits:* El mapa de bits es un campo que permite indicar qué bits del campo de datos contienen información válida. El primer bit indica si existe uno o dos mapas de bits (0 o 1 respectivamente). Los valores presentes en el mapa de bits dependen de la presencia (con el uso del valor 1) o no (con el valor 0) de un dato en el campo de datos.
- *Datos:* Es el campo donde se encuentra toda la información relacionada a la transacción, dentro de este campo se encuentran algunos elementos cada uno con una longitud definida y su presencia o ausencia dependerá del mapa de bits como se indicó anteriormente. La tabla 1.3 indica la correspondencia entre algunos elementos del campo de datos y los bits del campo mapa de bits.

1.1.2.2 Estados de un Cajero Automático

Un cajero trabaja bajo el concepto de estados, donde un estado representa la espera de un evento externo o interno a la aplicación del mismo. En la figura 1.3 se presentan los estados por los que generalmente pasa un cajero, basados en la aplicación utilizada en los cajeros automáticos NCR que utilizan el PIN para verificar al cliente.

NÚMERO DE BIT (Mapa de Bits)	ELEMENTO (Campo de datos)
2	Número de Cuenta Primario
3	Código de Procesamiento
4	Monto de la Transacción
7	Día y Hora de la transacción
12	Hora, transacción local
13	Día, transacción local
32	Código de Identificación de la institución a la que pertenece la cuenta
39	Código de Respuesta
48	Datos adicionales
49	Código de moneda
90	Elementos de datos originales

Tabla 1. 3: Correspondencia campo mapa de bits y elementos del campo datos

- *Estado 0:* El cajero se encuentra esperando a que un cliente ingrese una tarjeta para ser leída y pasar al siguiente estado.
- *Estado 1:* El cajero espera que el cliente ingrese el PIN desde el teclado.
- *Estado 2:* El cliente selecciona la transacción a realizar.
- *Estado 20:* En caso de ser un retiro, un avance o una transferencia, el cajero espera que el cliente ingrese el monto de la transacción.
- *Estado 21:* El cajero espera que el cliente confirme la transacción que desea realizar para enviar la petición a la aplicación del host central, la cual se encarga de procesar y autorizar la transacción. Luego se comunica al ATM un mensaje con la respuesta a la petición y un comando que ordena al cajero imprimir el resultado, dispensar el dinero si se trata de un retiro o un avance y posteriormente pasar al siguiente estado.
- *Estado 150:* El cajero espera que el cliente termine de realizar sus transacciones para volver al estado inicial.

La figura 1.3 representa el flujo normal de una transacción en un cajero automático; en este flujo se representan los estados anteriormente indicados.

Cabe la pena mencionar que cuando se produce un error en cualquiera de los estados, el flujo cambia al estado 150, después del cual el cajero regresa al estado 0.

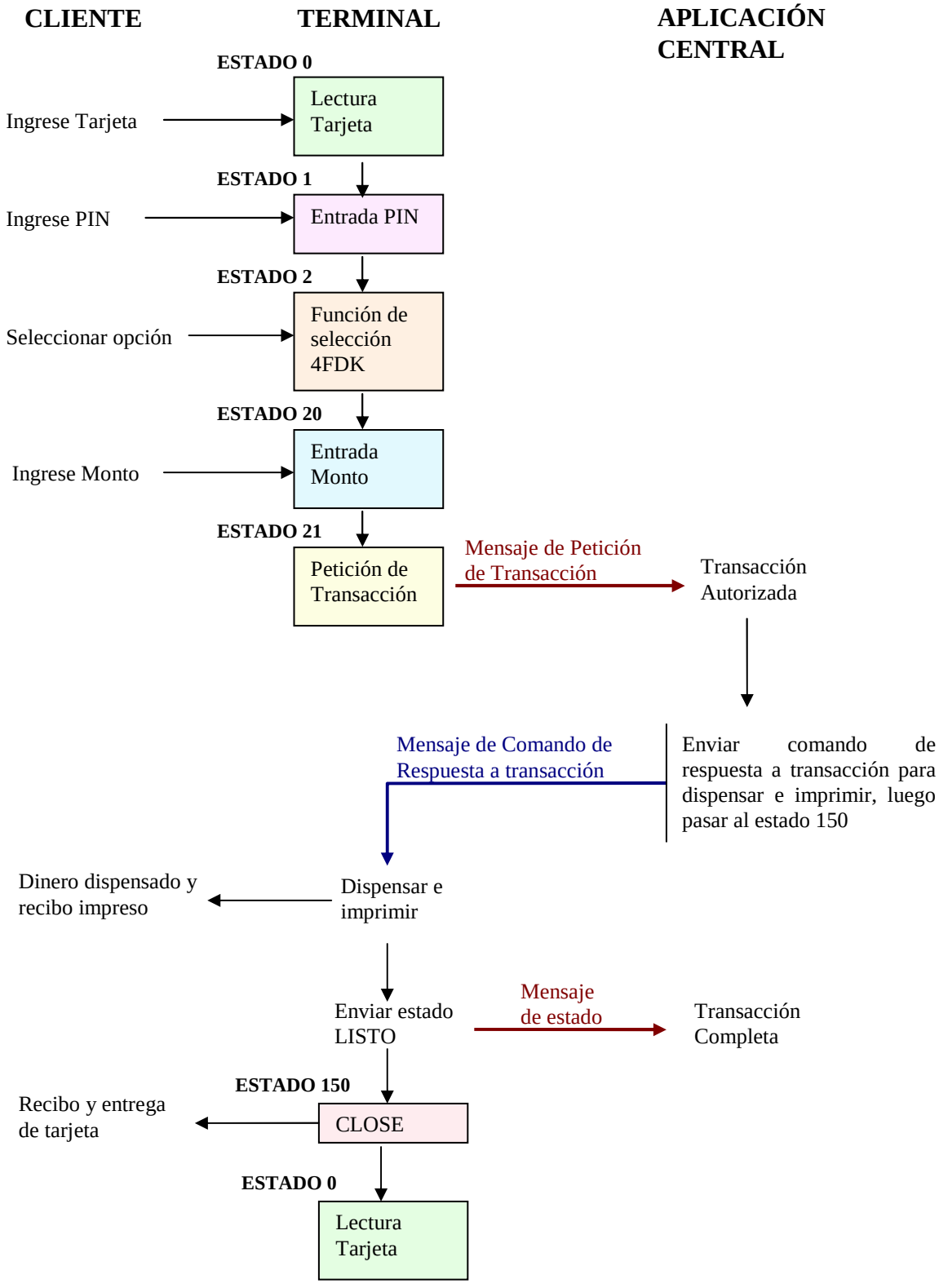


Figura 1. 3: Estados de un cajero automático NCR

1.1.2.3 Seguridad

La seguridad es un aspecto importante que se debe considerar cuando se trata de cajeros automáticos por el hecho de portar dinero en su interior y de manejar información confidencial como cuentas y claves de usuarios. Un cajero puede estar expuesto a distintos tipos de ataques, por lo que se explican algunos medios de prevención a dichos ataques:

- *Ataque físico:* En este tipo de ataques se agrede la integridad física del cajero. Normalmente, el delincuente intenta robar la caja fuerte. La solución más común es blindar el armazón del cajero y reforzar las cajas fuertes.
- *Robo de Tarjetas:* O de la información de una tarjeta, este ataque puede dar lugar a falsificaciones para lo cual se implementan mecanismos sofisticados (con ayuda de sensores por ejemplo) para detectar elementos fraudulentos en el cajero.
- *Robo de Información:* En este ataque personas no autorizadas intentan acceder a la información confidencial relacionada a las transacciones a través de la red por donde viajan las mismas. Para evitar este ataque un mecanismo muy usado es la criptología. Este mecanismo de seguridad es usado en la etapa de la autenticación de un usuario, ya sea por medio de una llave (un PIN por ejemplo), una contraseña o mediante algún método biométrico (por ejemplo huella dactilar, reconocimiento facial, u otros).

1.1.3 HARDWARE Y PERIFÉRICOS

Un cajero, en términos generales es una computadora común que tiene dispositivos periféricos y que se conecta con un servidor. Cada cajero está normalmente conformado por los siguientes dispositivos:

- *Procesador:* Puede ser el mismo procesador usado en una PC y se encuentra incorporado en un *motherboard* como se observa en la figura 1.4. Este procesador es el encargado de controlar la interfaz del usuario y las aplicaciones que interactúan con el mismo, y de enviar las peticiones del cliente al Host Central, así como controlar las operaciones de los otros dispositivos del cajero.

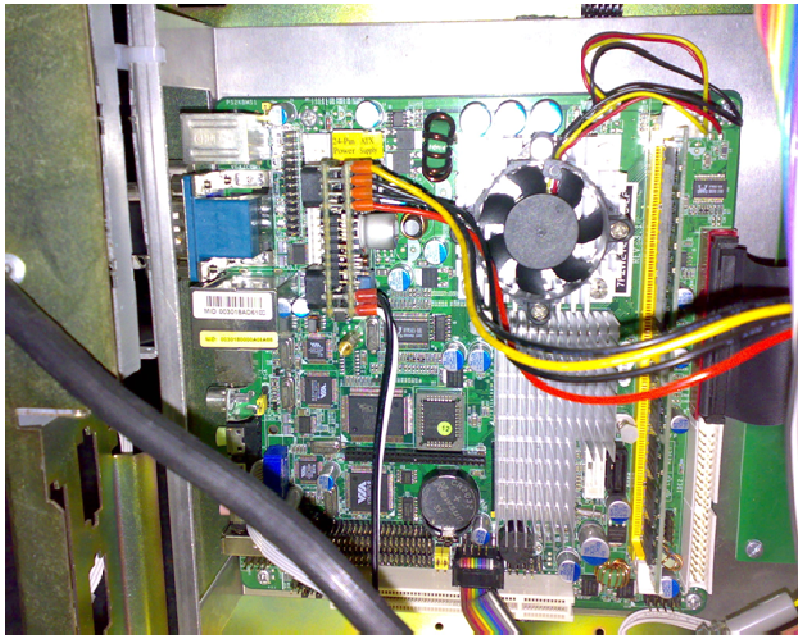


Figura 1. 4: *Motherboard* de un Cajero Automático

- *Lectora de Tarjetas:* Puede ser una lectora de tarjetas magnéticas o de *Chip*; se encuentra conectada a una tarjeta que sirve de interfaz (a través de una comunicación serial) entre la computadora y el dispositivo. Esta lectora extrae el número de identificación del usuario de la tarjeta y se lo comunica a la aplicación. Una lectora de tarjetas se presenta en la figura 1.5.

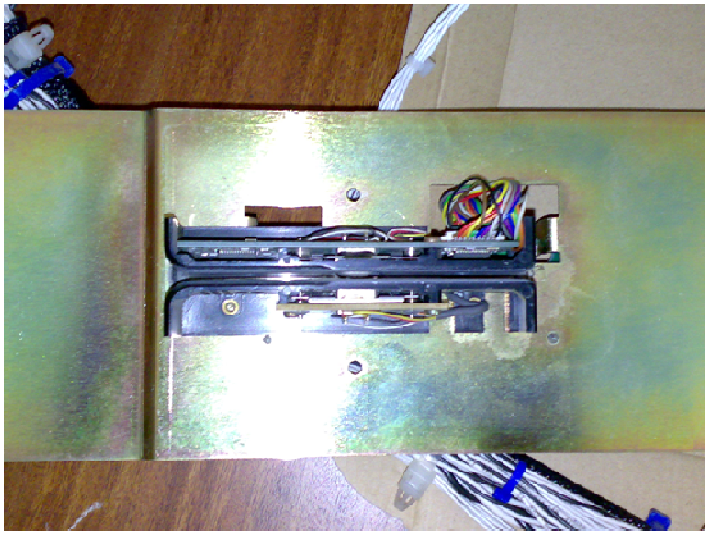


Figura 1. 5: Lectora de Tarjetas

- *Teclado:* El teclado cumple con varias funciones en el cajero, sirve entre otras para indicar el tipo de transacción que se desea realizar, el monto de

la transacción, e ingresar el PIN (en caso de que el cajero lo necesite), entre otras funciones. El teclado se conforma de dos partes, la primera de ellas son las teclas que normalmente se encuentran en el monitor del cajero y la segunda es el teclado numérico en sí. Una característica que se debe destacar es que el teclado encripta la información usando 3DES pero a nivel de hardware. El teclado se conecta a través del puerto USB al *motherboard* del cajero. En la figura 1.6 se tiene un modelo de teclado que normalmente es empleado en los cajeros automáticos.

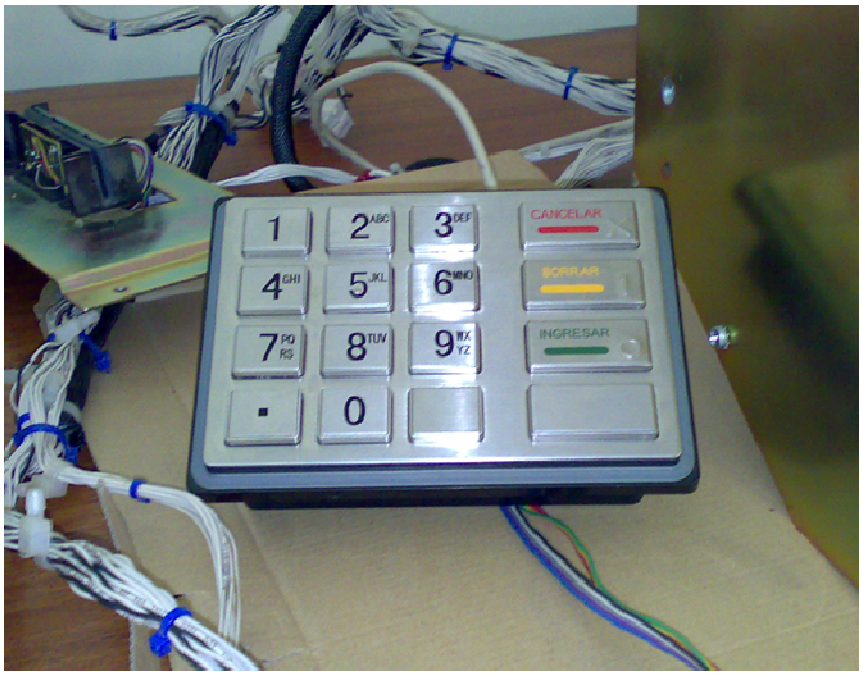


Figura 1. 6: Teclado

- *Pantalla:* Muestra el proceso de la transacción mediante diferentes pantallas con las que interactúa el cliente, encontrándose el monitor conectado al puerto de video del *motherboard*. Su ilustración se la puede observar en la figura 1.7.
- *Botones de Pantalla:* Generalmente se encuentran en los costados de la pantalla como se observa en la figura 1.7, son parte del teclado y se conectan a él como lo indica la figura 1.8. Estos botones permiten al usuario realizar selecciones con respecto a la transacción.
- *Dispensador:* Es uno de los componentes más importantes ya que entrega el dinero al cliente. Se encuentra conectado a una tarjeta controladora que

a su vez se encuentra conectada al *motherboard*. Esta tarjeta envía las señales al dispensador para que entregue los billetes si el cliente ha solicitado un retiro o avance en efectivo. Como se puede observar en la figura 1.9 cada división del dispensador sirve para agrupar los billetes por sus diferentes denominaciones.

Figura 1. 7: Pantalla

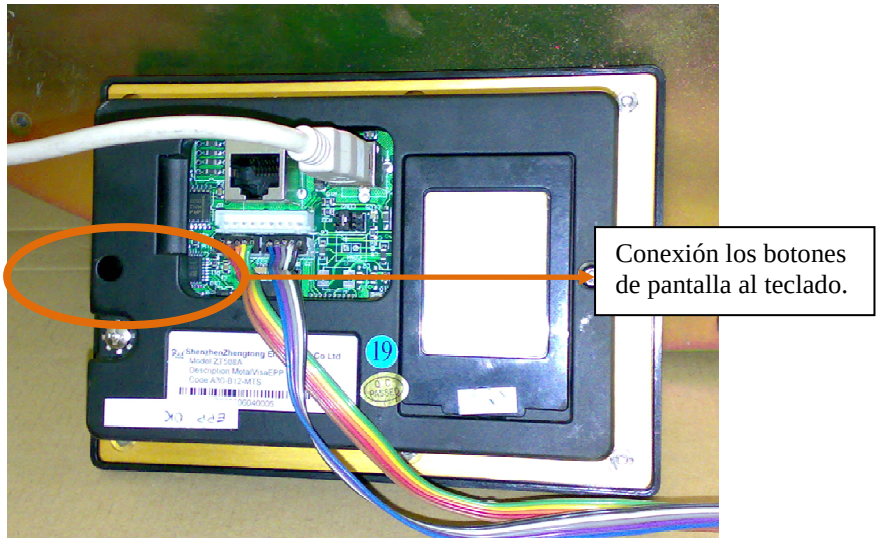


Figura 1. 8: Conexión de los botones de pantalla

Figura 1. 9: Dispensador

- *Impresora:* Imprime el comprobante de la transacción. Al igual que el dispensador se encuentra conectada a una tarjeta controladora que a su vez se conecta al *motherboard* a través de una interfaz serial. La figura 1.10 ilustra una de las impresoras comúnmente encontradas en el mercado.
- *Caja Fuerte:* Protege al dispensador para que no sufra un ataque físico. Como se puede apreciar en la figura 1.11 este componente es blindado.
- *Envoltura:* Es la carcasa externa del cajero, sirve para proteger a los diferentes dispositivos tanto del medio ambiente como de los ataques físicos. En la actualidad también sirve para adornar el sitio en el que se encuentra el cajero ya que se les da diferentes formas. Dos tipos de envolturas se pueden apreciar en la figura 1.12.

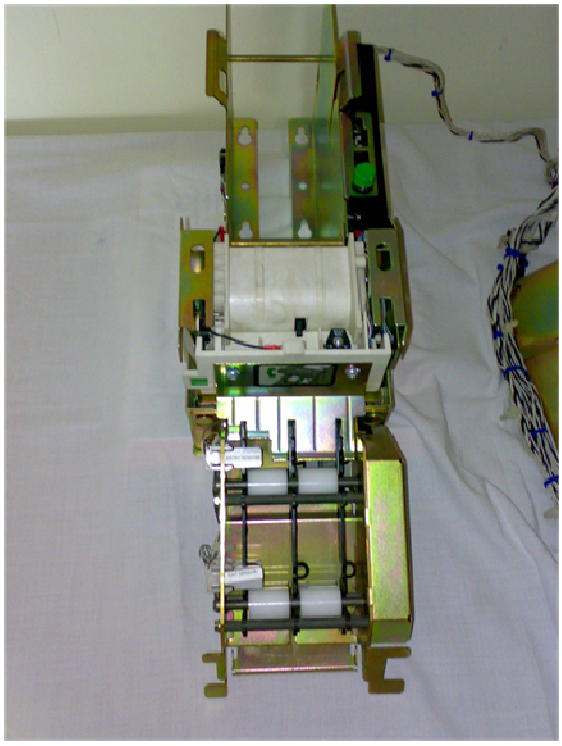


Figura 1. 10: Impresora

Figura 1. 11: Caja Fuerte

Figura 1. 12: Envolturas

1.1.4 SOFTWARE EMPLEADO

El cajero automático es un computador que maneja ciertos dispositivos adicionales. Al igual que cualquier computador necesita de un sistema operativo, el cual permite administrar sus recursos. Se han utilizado varios sistemas operativos a lo largo de la historia de los ATMs pero los más usados han sido RMX, OS/2, Linux y sistemas operativos de Microsoft dentro de los cuales están Windows 2000, Windows XP Professional y/o Windows XP Embebido.

OS/2 fue desarrollado por IBM y Microsoft, posteriormente fue distribuido sólo por IBM. Este sistema operativo se lo implementó en cajeros automáticos debido a que es un sistema estable, rápido y seguro; en los últimos años se ha comenzado a promover la migración de OS/2 a Windows XP ya que este sistema da la posibilidad de añadir mayores funcionalidades al cajero automático, manejar otro tipo de dispositivos y aplicaciones que OS/2 como tal no las puede manejar. Generalmente la versión de Windows que se maneja en los ATMs es XP Embebido el cual está basado en *XP Professional* pero sólo tiene las funciones básicas y necesarias, con el propósito de consumir menos recursos.

En cuanto al software utilizado a nivel de aplicación para controlar las transacciones, generalmente éste es provisto por el fabricante del cajero

automático; los más difundidos son los fabricados por: *NCR, Diebold e IBM.* De éstos, NCR *Direct Connect* (*NDC*) es el más utilizado en América y Europa. De la misma forma cada fabricante ofrece una plataforma de arquitectura de software que permita desarrollar aplicaciones para que el usuario acceda a los servicios del cajero.

Los lenguajes utilizados para el desarrollo de las aplicaciones de los ATMs pueden ser: *C, C++, Java, Visual Basic, Visual Basic .NET, Cobol*, entre otros dependiendo del sistema operativo con el que se esté trabajando. *Cobol* es un lenguaje que aún es usado en instituciones financieras por la compatibilidad que presta con sistemas antiguos, mientras que *Visual Basic.NET* por las facilidades y herramientas que provee para el desarrollo de aplicaciones se ha convertido en el nuevo lenguaje para el desarrollo de aplicaciones orientadas a los cajeros.

1.1.5 RED DE CAJEROS [3]

La mayoría de cajeros automáticos se conectan a redes interbancarias, permitiendo que un cliente pueda realizar una transacción en cualquier cajero independientemente del banco al que su cuenta pertenece o de la ubicación en la que se encuentre. En el Ecuador la red interbancaria de cajeros automáticos más grande es BANRED con aproximadamente 2180 cajeros automáticos; esta red se encuentra conformada por las siguientes instituciones financieras.

- *Bancos:* Bolivariano, Comercial de Manabí, De Guayaquil, De Machala, Del Austro, Del Pacífico, Delbank, Internacional, M.M. Jaramillo Arteaga, Pichincha, Produbanco, Solidario y Unibanco.
- *Tarjetas de Crédito:* Diners Club.
- *Cooperativas:* Alianza del Valle, El Sagrario, Pablo Muñoz Vega, Tulcán y 29 de Octubre.

Las transacciones que se pueden realizar en los cajeros automáticos de BANRED son:

- *Cuenta Corriente y Cuenta de Ahorros:* Retiros, consultas y transferencias entre cuentas.

- *Tarjeta de Crédito:* Avance de efectivo, pago de la tarjeta y consulta de saldo.

Otras redes interbancarias importantes en el país son NEXO, CIRRUS, MAESTRO donde las dos últimas mencionadas corresponden a redes interbancarias internacionales.

1.1.6. BENEFICIOS DE LOS CAJEROS AUTOMÁTICOS

La evolución tecnológica ha traído muchas facilidades a la vida del ser humano ya que se ha visto orientada a satisfacer las necesidades del mismo. El cajero automático es un dispositivo que con el paso de los años ha ido evolucionando para brindar un mejor servicio a los usuarios y mejorar su calidad de vida, así por ejemplo se pueden enumerar los siguientes beneficios:

- Reduce las colas en el interior de los bancos para realizar una transacción en ventanilla y por ende el tiempo que toma realizar una transacción.
- Permite a los clientes realizar transacciones como retiros o transferencias a cualquier hora del día en cualquier lugar del mundo.
- Se pueden encontrar cajeros automáticos en cualquier lugar tanto en zonas urbanas (centros comerciales, gasolineras entre otros sitios de gran afluencia de personas), así como también en zonas rurales.
- El cliente puede acceder a servicios adicionales como pago de impuestos o servicios básicos.

1.2 IDENTIFICACIÓN BIOMÉTRICA

A lo largo de la historia, los diferentes sistemas computacionales se han visto en la necesidad de brindar seguridad en el acceso a la información que custodian. Por estos motivos se han desarrollado diferentes métodos de autenticación para controlar el acceso de los usuarios; estos métodos de autenticación se los puede clasificar en tres grupos:

- *Métodos Basados en Tokens:* Estos métodos usan dispositivos físicos y portables como tarjetas, pasaportes, tarjetas de banco, etc.

- *Métodos Basados en Contraseñas:* Estos métodos hacen referencia al conocimiento del usuario de una contraseña o un PIN por ejemplo y basan su seguridad en el secreto de la misma.
- Métodos basados en Biométrica: Estos métodos se basan en la autenticación de las personas mediante el uso de una característica física o de su comportamiento.

1.2.1 CONCEPTOS

El término biométrica está conformado por los términos *bio* que significa vida y *métrica* que hace referencia a la medición; la biométrica consiste en la aplicación de métodos estadísticos al gran rango de las características medibles en la biología; en la actualidad se ha orientado el uso de este término a la identificación automática de personas a través de sus características físicas o comportamiento.

Los métodos de autenticación biométrica se pueden implementar de dos maneras; la primera de ellas es la *verificación* que consiste en comparar el rasgo biométrico del usuario que busca autenticación con el que teóricamente le pertenece (en la figura 1.13a se representa el diagrama de bloques de este comportamiento). El segundo método en cambio es el de *identificación,* este método se encarga de buscar al usuario a través de su rasgo biométrico en toda la base de datos existente (en la figura 1.13b se representa el diagrama de bloques de este método).

Sin importar el tipo de implementación o el tipo de método de autenticación biométrica que se seleccione para un sistema, se pueden destacar tres ventajas importantes de la biometría frente a los sistemas de autenticación tradicionales (sistemas basados en *tokens* o en contraseñas):

- *Seguridad:* Los sistemas biométricos incrementan el nivel de seguridad en un sistema ya que una característica biométrica no puede ser robada, perdida, compartida o adivinada como suceden en los sistemas basados en contraseñas o *tokens*.

- *Administración:* Ya que una característica biométrica no puede ser usada por otra persona que no sea el usuario, se puede saber a ciencia cierta quién está ingresando al sistema; pudiendo de esta manera crear un registro de todas las actividades realizadas por el usuario.

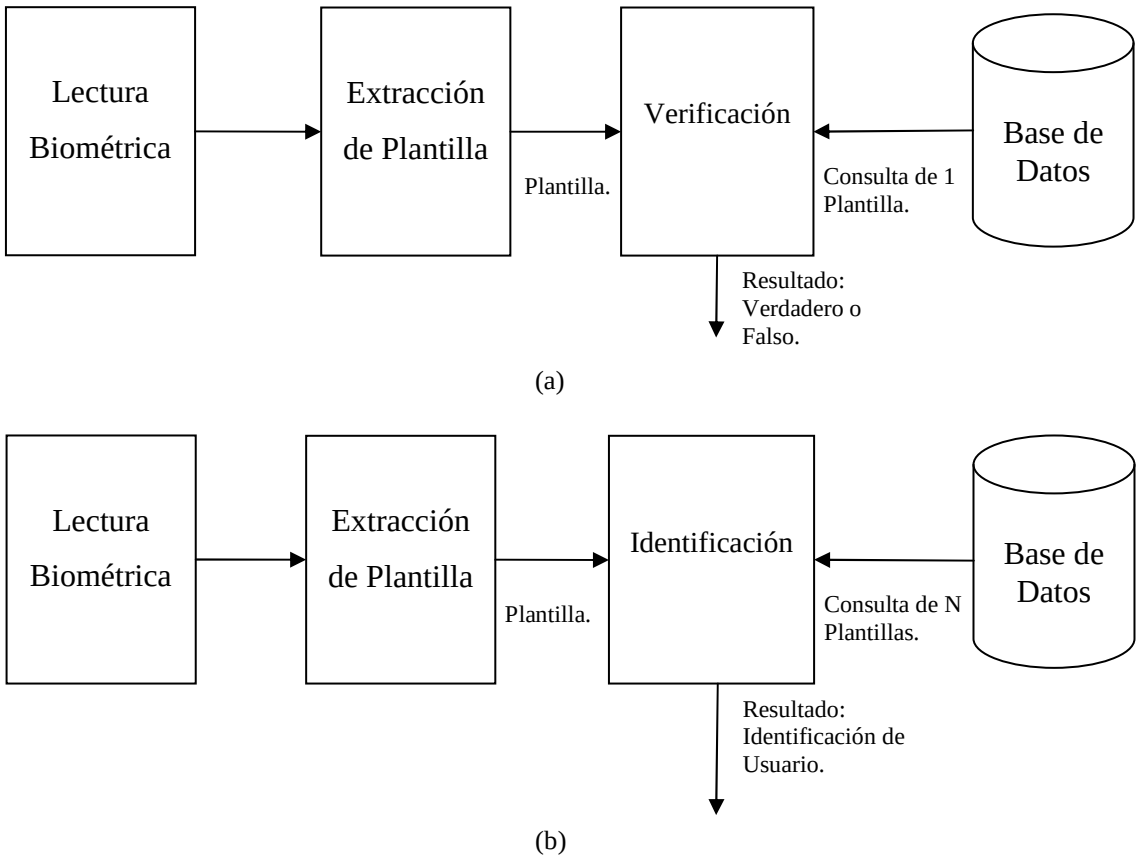


Figura 1. 13: Tipos de Implementación Biométrica: Verificación y Autenticación

- *Conveniencia:* Los sistemas biométricos son empleados en ambientes en donde los privilegios de acceso son importantes por los niveles de seguridad y administración que ofrecen. Adicionalmente, los sistemas biométricos simplifican la autenticación ya que pueden reemplazar el uso de varios *tokens* y contraseñas por el uso de solo una característica biométrica.

Además de todas las ventajas previamente descritas, se pueden obtener mayores ventajas en robustez de esquemas de seguridad en un sistema de autenticación que combine el uso de estos métodos (basados en *tokens* o contraseñas) con el uso de características biométricas, incluso pudiéndose combinar el uso de varias

características biométricas. Estos sistemas son conocidos como sistemas híbridos de autenticación y las combinaciones de métodos de autenticación dependen de las necesidades de seguridad de la aplicación que se está diseñando y/o implementando.

1.2.2 MÉTODOS DE IDENTIFICACIÓN BIOMÉTRICA

Las características biométricas pueden ser separadas en dos grupos: características físicas y de comportamiento. El primer grupo enmarca todas las características biométricas que físicamente se encuentran en la persona como huellas dactilares, iris, rostro, retina, etc.; mientras que el segundo grupo engloba todas las características biométricas que dependen del comportamiento de la persona como la firma, la voz, la forma de caminar, etc.

Cada una de estas características por su naturaleza posee ventajas y desventajas, por lo tanto la selección de la característica biométrica para ser empleada en un sistema dependerá de los requerimientos y niveles de seguridad deseados en el mismo.

Para seleccionar una característica biométrica, normalmente se evalúan los siguientes parámetros:

- *Universabilidad:* Toda persona que emplee el sistema debe tener la característica biométrica.
- *Permanencia:* La característica biométrica tiene que mantenerse igual en el tiempo sin sufrir cambios mayores.
- *Medible:* La característica biométrica debe ser capaz de ser medible cuantitativamente.
- *Distintiva:* La característica biométrica debe ser única para cada persona en el grupo de usuarios de la aplicación.
- *Rendimiento:* Hace referencia al rendimiento de la aplicación tanto para extraer la plantilla de la característica como en la velocidad para verificarla e identificarla.
- *Aceptabilidad:* Representa el nivel de aceptación del usuario para usar un sistema que implemente la característica biométrica.

- *Robustez:* Indica el grado de seguridad del sistema frente a ataques fraudulentos al mismo.

1.2.2.1 Reconocimiento de Huellas Dactilares

Una huella dactilar consiste básicamente en un patrón único de valles y crestas en la superficie de los dedos que se forman desde los primeros meses de la vida de un feto. La principal desventaja del uso de estos sistemas es que el usuario debe estar dispuesto a colaborar para proveer su huella al sistema y que debido a la interacción que tiene el usuario con los dispositivos de captura se puede obtener imágenes movidas o desalineadas.

En la actualidad, el uso de huellas dactilares en sistemas biométricos constituye el sistema más confiable por su universabilidad, por ser una característica distintiva y por su robustez.

1.2.2.2 Reconocimiento Facial

Este método biométrico usa el rostro de las personas para autenticarlas midiendo la simetría, distancias y tamaños de las partes faciales. Normalmente en este sistema se tienen 3 pasos: detección, localización y reconocimiento del rostro. Las principales desventajas que se tienen giran en torno al hecho que las imágenes son capturadas en ambientes en donde no se tiene control de las condiciones de iluminación y fondo, ni tampoco se puede controlar las expresiones faciales por parte de la persona. Este tipo de método de autenticación es probablemente el sistema más amistoso y aceptado para autenticación de usuarios.

1.2.2.3 Reconocimiento de Voz

El reconocimiento de voz es un método que combina las características físicas y de comportamiento de los patrones de las señales de voz de una persona. Las características físicas se relacionan a los tonos de la señal que forman el sonido, mientras que las características del comportamiento hacen referencia al estado de ánimo del individuo. Los problemas que enfrenta este método se deben justamente al estado de ánimo de las personas al hablar, además del ruido (sonidos externos) que pudiese existir en el ambiente en donde se encuentra la aplicación.

1.2.2.4 Reconocimiento de la Forma de la Mano

Este método de autenticación se basa en la geometría de la mano en donde se toman en cuenta factores como la forma y tamaño de la mano, así como el largo y el ancho de los dedos. La principal ventaja de este método es la simplicidad de su implementación y bajo costo; mientras que sus principales desventajas son su baja capacidad para discriminar a los usuarios (ya que se pueden tener usuarios que tengan similares características biométricas en el rango permitido) y el hecho que esta característica es cambiante durante el período de crecimiento de una persona.

1.2.2.5 Reconocimiento de Iris

El iris es una textura que se encuentra en el ojo humano, la información (color, rasgos y forma) de esta textura es formada durante el período fetal y su formación se extiende hasta los dos primeros años de vida. Las principales ventajas de este método es que la información contenida del iris es única para cada individuo y no cambia con el tiempo, es muy difícil poder modificarla quirúrgicamente y su falsificación demanda de mucho tiempo y precisión convirtiéndose prácticamente en una tarea imposible; mientras que su principal desventaja radica en el hecho de que los sistemas que emplean este método necesitan de la colaboración del usuario para poder autenticarlo.

1.2.2.6 Reconocimiento de la Retina

La retina es un patrón de venas ubicadas en la parte posterior del globo ocular. Este patrón es único para cada persona y se considera que el uso de esta característica biométrica es la más segura de todas. Los sistemas que implementan este método biométrico deben escanear la retina por medio de un rayo infrarrojo de baja intensidad que se proyecta sobre el ojo para poder digitalizar el patrón de venas. Las desventajas de este método radican en el hecho que se necesita la cooperación del usuario y en el alto costo de un escáner de retina ya que es un método poco difundido y normalmente aplicado en sistemas de alta seguridad.

1.2.2.7 Reconocimiento de la Firma

Los métodos de autenticación basados en el reconocimiento de la firma se basan principalmente en la geometría de la misma, aunque existen sistemas más complejos que toman en cuenta además de la geometría de la firma, la velocidad, la presión que se ejerce al escribir, la aceleración y la trayectoria de la firma. Normalmente este método no funciona bien en grandes sistemas porque los parámetros antes mencionados dependen de estados emocionales y físicos de la persona.

1.2.2.8 Reconocimiento de Termo gramas de Venas

Los sistemas de autenticación basados en el reconocimiento de termo gramas de venas usan el hecho que los patrones de las venas en la mano o dedos son únicos e invariables en la vida de un individuo. Para obtener un patrón de venas se emplea un escáner con un sensor infrarrojo que captura una imagen de la hemoglobina de la sangre. Normalmente, el oxígeno presente en la hemoglobina aparece en patrones negros en las imágenes capturadas. El principal problema que enfrenta esta tecnología es que las venas son flexibles frente a la cantidad de sangre que pasa por ellas lo que causa el engrosamiento (o afinamiento) y el movimiento de las mismas. La desventaja de este método está involucrada con el costo de los sensores infrarrojos que son necesarios para obtener los patrones de las venas.

1.2.2.9 Comparación de Métodos

En la tabla 1.4 se realiza una comparación de los métodos biométricos antes descritos. Cada método se califica en una escala de 5 puntos (en donde 0 es la más baja calificación y 5 la más alta) según los parámetros especificados en la sección 1.2.2.

Característica Biométrica	***Universabilidad***	***Permanencia***	***Medible***	***Distintiva***	***Rendimiento***	***Aceptabilidad***	***Robustez***	***Resultado***
Huellas Dactilares	3	5	4	5	5	5	5	***32***
Facial (Rostro)	5	3	4	2	2	5	1	***22***
Voz	3	2	3	1	1	5	2	***17***
Forma de la Mano	4	3	5	3	3	3	3	***24***
Iris	5	5	3	5	5	2	5	***30***
Retina	5	3	2	5	5	2	5	***27***
Firma	1	1	5	1	2	4	2	***16***
Termograma de Venas	4	3	5	5	3	5	5	***30***

Tabla 1. 4: Comparación de Métodos Biométricos

1.2.3 RECONOCIMIENTO DE HUELLAS DACTILARES

Como se mencionó en la sección 1.2.2.1 una huella dactilar consiste en un patrón único de valles, crestas y minucias sobre la superficie de los dedos. La primera característica que se destaca en las huellas dactilares es que cada una de ellas es *única* entre diferentes individuos y única también entre las diferentes huellas del mismo individuo. Otra característica importante asociada a las huellas dactilares es la *persistencia* que hace referencia al hecho que las huellas no cambian a lo largo de la existencia de una persona. Básicamente, debido a estas dos características se considera que los sistemas basados en esta característica biométrica son altamente seguros y por tal motivo han sido ampliamente aceptados a nivel mundial.

1.2.3.1 Conceptos

- *Estructura de las Huellas Dactilares:* La estructura de las huellas dactilares se conforma de dos niveles. El primero de ellos es el patrón de valles y crestas que son empleados como elementos de clasificación dependiendo del patrón formado. A este nivel no se puede autenticar un usuario ya que estos patrones no proveen suficiente información para establecer la identidad de un usuario. La figura 1.14 representa la imagen de una huella dactilar en donde los valles corresponden a los patrones de líneas blancas, mientras que las crestas a los patrones de líneas negras.

Figura 1. 14: Patrones de valles y crestas

Por otro lado, el segundo nivel que forma la estructura de las huellas dactilares está conformado por puntos característicos de los patrones de las crestas. Estos puntos son conocidos como minucias y corresponden a los puntos en donde las crestas terminan, se bifurcan o se trifurcan. En este nivel es en donde se puede autenticar a un usuario ya que el conjunto de estos puntos son únicos por cada huella e individuo, proveyendo así la suficiente información discriminatoria para la identificación del usuario. En la figura 1.15 se muestra un ejemplo de las diferentes minucias que se encuentran presentes en una huella.

Figura 1. 15: Ejemplos de minucias presentes en las huellas dactilares

- *Formación:* Las huellas dactilares se forman durante la fase fetal. Alrededor de la semana número ocho del embarazo, las crestas comienzan a aparecer en la superficie del feto pero no es sino hasta veinte semanas después, o sea en la semana veintiocho aproximadamente, que las huellas están formadas completamente. A partir del séptimo mes (semana veintiocho aproximadamente) la estructura de las huellas se mantiene igual a lo largo de la vida del individuo salvo en los casos en que la huella sufre un daño externo como cortes, quemaduras, magulladuras, etc.

- *Individualidad:* La estructura de una huella dactilar depende tanto de factores genéticos como de factores asociados a la fase fetal (por ejemplo: la posición del feto en el útero, la cantidad de líquido amniótico entre otros) que causan pequeñas deformaciones en las crestas de las huellas dactilares. De tal suerte, las huellas dactilares son únicas entre los individuos así como entre los diferentes dedos del mismo individuo, entre hermanos del mismo padre y madre (se comparten cadenas de ADN), y en el caso particular de gemelos idénticos que comparten la misma cadena de ADN (factor genético). El concepto de individualidad ligado a los sistemas biométricos garantiza la unicidad de cada huella debido al segundo grupo de factores (factores asociados a la fase fetal).
- *Revisión de Calidad:* En un sistema de autenticación de usuarios basado en características biométricas, uno de los factores más importantes a tomar en cuenta es la fuente desde la cual se extraerá dicha caracterìstica. Se puede configurar como fuentes al sonido (grabaciones), video, imágenes (fotografías, imágenes escaneadas o imágenes de un dispositivo especializado) entre otras fuentes. En el caso particular de las huellas dactilares las fuentes desde las cuales se extraen las imágenes de las huellas se las obtiene de un escáner de huellas.

1.2.3.2 Normativa

En el campo de las huellas dactilares existen cuatro estándares principales y reconocidos a nivel mundial que dan la normativa básica para el tratamiento y procesamiento tanto de la imagen correspondiente a una huella dactilar así como para la plantilla que se genera a partir de dicha imagen. En el diseño de una aplicación que emplee huellas dactilares, el uso de uno de estos estándares permite mantener la compatibilidad con otros sistemas que manejen el mismo estándar, sin importar el SDK (proveedor del software) que se vaya a emplear. Normalmente el formato que se emplea para la compresión de las huellas dactilares, tanto para almacenamiento como para transmisión, es *WSQ (Wavelet*

Scalar Quantization)[1] aunque muchas organizaciones desarrolladoras de productos para identificación de huellas dactilares han definido sus propios estándares.

Los principales estándares normados por organismos de acogida mundial son:

- *ANSI:* Este organismo a través del grupo de trabajo X9 y los varios subcomités que lo conforman generó el estándar americano para seguridad en aplicaciones denominado *X9.84-2001, Administración y Seguridad de Información Biométrica.*

- *ANSI/NIST/FBI:* El estándar desarrollado por este organismo se basa en el estándar creado por ANSI. Éste es el estándar que mayor acogida ha tenido a nivel mundial; es empleado por el FBI para el manejo de interoperabilidad e intercambio de información entre las diferentes bases de datos del sistema *AFIS*[2], además de ser empleado por la INTERPOL (Organización Internacional de Policía) y varias agencias criminalísticas de diferentes países alrededor del mundo. Este estándar define los requerimientos y formatos para las imágenes y plantillas de las huellas dactilares, rostro, marcas y tatuajes. El estándar se denomina ANSI/NIST-ITL 1-2000. En el anexo A del presente proyecto de titulación se incluye una copia de este estándar debido a su importancia, ya que este proyecto se ha desarrollado en base al mismo.

- *ISO/IEC: ISO* (*International Standard Organization*) junto a *IEC* (*International Electrotechnical Commission*) establecieron un comité técnico, llamado JTC1, para el desarrollo de un estándar de seguridad de la información a través de tres subcomités. El primero de los tres subcomités se conoció como JTC1/SC17 y se enfocó en el desarrollo de estándares para la identificación de documentos. El segundo subcomité, JTC1/SC27, trabajó sobre estándares asociados a sistemas de autenticación biométrica. Finalmente, el tercer subcomité denominado

[1] WSQ es un método de compresión con pérdida perfecto para preservar los detalles de alta resolución de imágenes en escala de grises, manteniendo altos índices de compresión, normalmente 12:1 a 15:1.
[2] *AFIS:* Es un sistema desarrollado por el FBI para la identificación automática de huellas dactilares.

JTC1/SC37, se enfocó en la estandarización de tecnologías biométricas genéricas para el soporte de interoperabilidad e intercambio de datos entre sistemas y aplicaciones biométricas.

- *BioAPI:* BioAPI es un consorcio que desarrolló un estándar informal implementado sobre una API[3] que ha sido aceptado por vendedores, investigadores, y desarrolladores en el área de las huellas dactilares. La idea de crear esta API es que las aplicaciones biométricas puedan ser implementadas independientemente de la tecnología biométrica que se esté usando.

1.2.3.3 Principios de Funcionamiento

La detección de los patrones únicos presentes en una huella dactilar se lo realiza a través de la captura de la imagen de la huella dactilar mediante el uso de algún tipo de dispositivo que sirva para este fin. La imagen capturada es procesada para mejorar su calidad y así poder extraer la plantilla (formada por vectores y distancias entre minucias) que servirá para la comparación con la plantilla previamente almacenada.

En el proceso de captura de la imagen de la huella dactilar el principal parámetro a tomar en cuenta es la calidad de la imagen, la misma que dependerá de factores como: tipo de sensor usado en el dispositivo de captura, presión ejercida por la persona sobre el sensor, condiciones de la huella (mojada, seca, o las rugosidades de la misma), presencia de ruido en el momento de la captura, la resolución de la imagen, y marcas residuales de huellas previas sobre el lector.

La primera solución para este tipo de problemas es la tecnología empleada por el lector de huellas dactilares, ya que según el método que emplea minimiza o maximiza los problemas antes mencionados. En la actualidad el lector de huellas basado en lectura óptica es el que provee la mejor calidad de imagen frente a otros existentes en el mercado. En la sección 1.2.3.6 se profundiza en los tipos de lectores según la tecnología que emplean.

[3] API (***A****pplication* ***P****rogramming* ***I****nterface*): Es la interfaz que define funciones y métodos entre dos componentes de software para su correcta adaptación y comunicación.

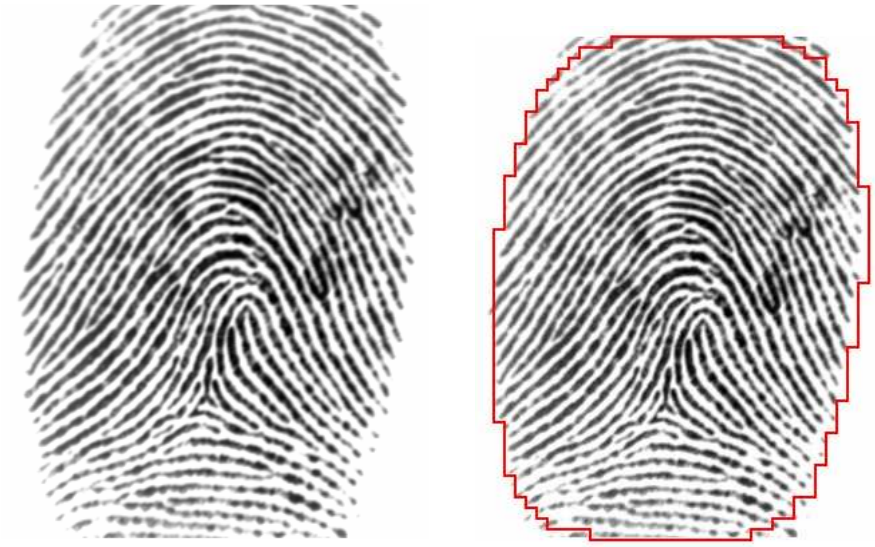
La segunda solución se la asocia al procesamiento de la imagen de la huella dactilar. Una vez capturada la imagen y validada su calidad se procede a realizar diferentes procesos para mejorar la misma y poder extraer la plantilla para la comparación y validación del usuario.

Los procesos que normalmente se siguen en el procesamiento de una imagen son los siguientes:

- Segmentación de la huella.
- Optimización de la huella a través de la normalización o el procesamiento basado en FFT[4].
- Determinación de los patrones característicos.
    - Determinación del plano de orientación.
    - Detección de las singularidades.
    - Extracción de minucias.
    - Extracción del patrón de la huella.
- Extracción de la plantilla y validación de la huella (que se puede basar en procesos de correlación, en base a la comparación de minucias, o en base a la comparación de singularidades).

La *segmentación* es el proceso a través del cual se separa la imagen de la huella del fondo para reducir el tiempo de procesamiento de la imagen y evitar que se tengan falsos patrones de la huella. Normalmente, este proceso emplea métodos que se aplican en bloques como la detección de contraste entre zonas, y la comparación de orientación y frecuencia de los patrones para poder delimitar la zona. Este proceso se complica cuando se tienen imágenes con ruido o de baja calidad ya que no se tienen límites claros de la imagen de la huella. En la figura 1.16 se tiene un ejemplo de una huella segmentada a partir de su huella original, la línea roja marca la segmentación de la huella a partir de su entorno.

[4] FFT (Transformada Rápida de Fourier): Representa al algoritmo que permite calcular la transformada de Fourier discreta y su inversa. La esencia de este algoritmo es descomponer la transformada en otras más simples y con límites reducidos.



(a) (b)

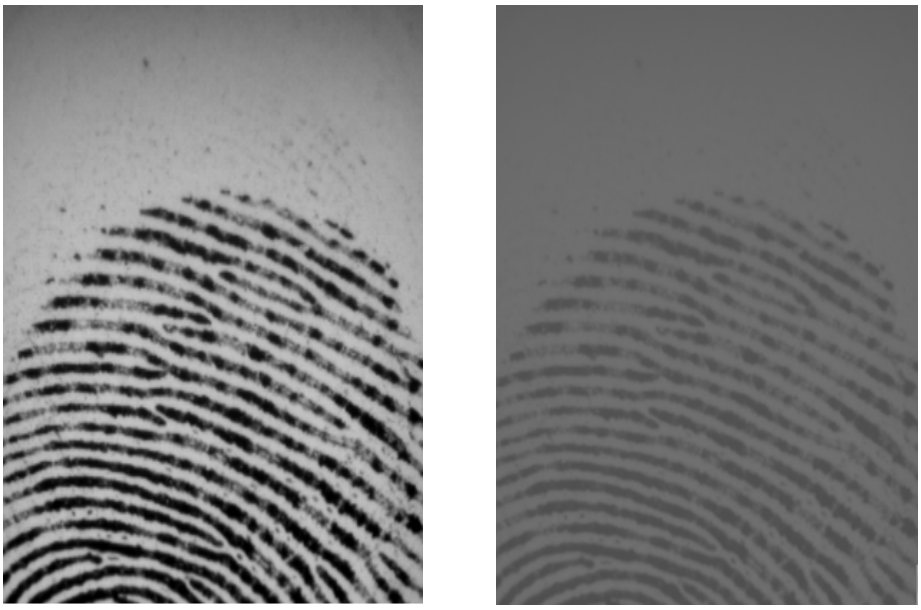
Figura 1. 16: Ejemplo de segmentación de huellas. (a) Huella original. (b) Huella segmentada [16]

El segundo proceso que se aplica a la imagen después de delimitar la zona de la huella (segmentación), es el de la *optimización de la imagen*. En esta segunda etapa se pueden aplicar dos métodos: la normalización o el procesamiento basado en FFT. Esta etapa se la emplea como un preprocesamiento de la imagen y permite mejorar la calidad de la misma al realzar la estructura de patrones o mejorar la consistencia de la orientación de la huella, consiguiendo así reducir los cortes entre crestas, reducir el ruido entre patrones y mejorar el contraste de los patrones presentes en la imagen.

Cuando se emplea la *normalización* como método de optimización de la imagen, se cambia la intensidad de cada pixel de la imagen a ciertos valores predefinidos para eliminar el ruido y corregir deformaciones preservando la claridad y el contraste entre crestas y valles. El resultado del proceso de normalización se lo puede apreciar en la figura 1.17.

El otro método empleado en la optimización de huellas es el basado en *FFT.* Este método es usado normalmente en el procesamiento de señales e imágenes para detectar las altas y bajas frecuencias presentes. En el caso particular del procesamiento de huellas dactilares, la transformada de Fourier permite detectar

la frecuencia y la orientación de los patrones presentes en la huella. En la figura 1.18 se aprecian las etapas que se siguen en el procesamiento de huellas empleando la transformada rápida de Fourier.



(a) (b)

Figura 1. 17: Proceso de normalización. (a) Huella original. (b) Huella procesada [16]

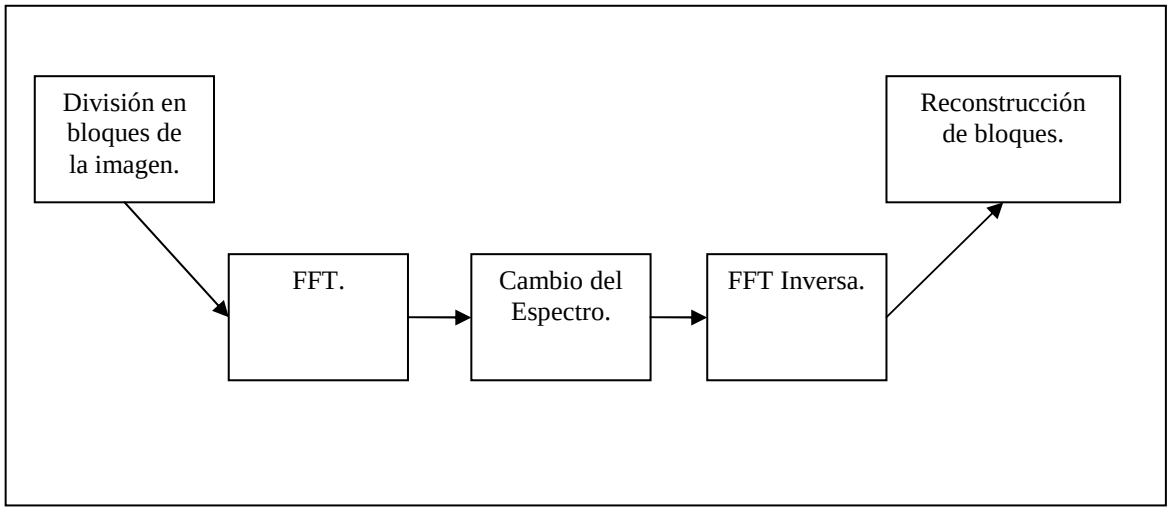


Figura 1. 18: Diagramas de Bloques del procesamiento de huellas dactilares basado en la transformada Rápida de Fourier

En el primer paso se procede a dividir la imagen en bloques de menor tamaño a los cuales se les aplica la transformada rápida de Fourier para después multiplicarlos por su espectro de potencia elevado a algún valor. Luego de esto se aplica la FFT inversa, y se reconstruyen los bloques usando ventanas sobrelapadas para evitar el efecto de borde producido por la transformada rápida de Fourier, obteniendo así la imagen optimizada. Un ejemplo de una imagen de huella procesada con este método se encuentra en la figura 1.19.

(a) (b)

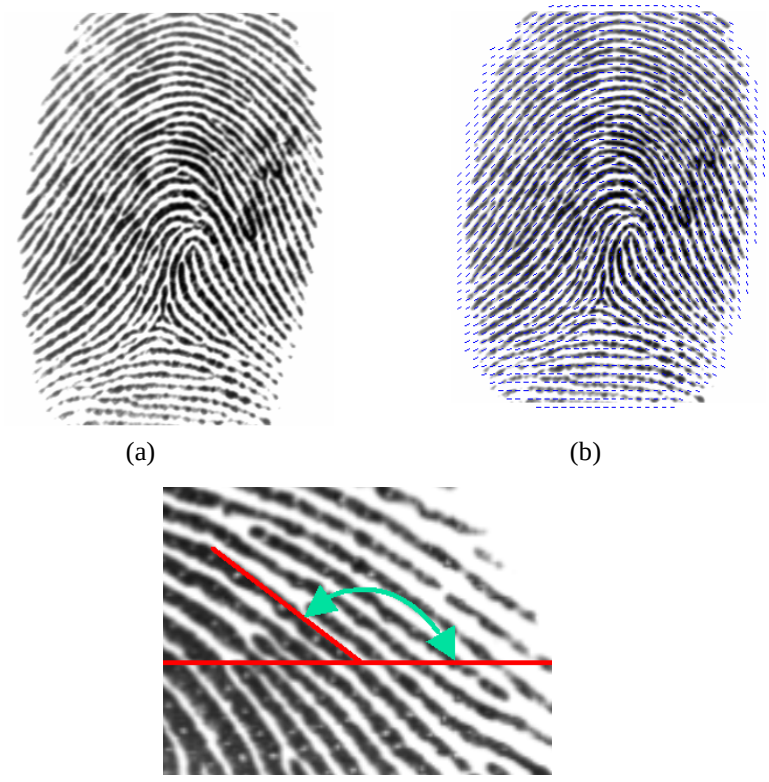
Figura 1. 19: Proceso de FFT. (a) Huella original. (b) Huella procesada [16]

Una vez que se tiene la imagen optimizada mediante el uso de uno o de los dos métodos descritos anteriormente, se procede a *extraer los patrones característicos* asociados a cada huella dactilar.

La huella dactilar se caracteriza por patrones de líneas de valles y crestas, que van en paralelo sobre la superficie del dedo; estos patrones pueden terminar, bifurcarse o cambiar su curvatura. Cuando se habla de los patrones de la huella dactilar en general, se determinan las *singularidades* que son los grandes cambios de curvatura de los patrones de la huella dactilar. En cambio, cuando se habla de los patrones de la huella dactilar en términos locales, se determinan las minucias que son, como antes se mencionó, los puntos característicos de la huella. Una minucia puede ser la bifurcación, trifurcación o terminación de los patrones de los valles y crestas, así como la presencia de otras características como los poros que son pequeños espacios en una cresta, los puntos (pequeñas

crestas) o lagos (dos bifurcaciones muy unidas). Adicionalmente, en la huella se puede determinar la orientación y frecuencia de las zonas de crestas.

El mapa de orientación permite detectar singularidades en la huella dactilar; como la huella no tiene un plano de referencia o de orientación, esta característica hace referencia a una zona o bloque de cierta dimensión en donde se detecta si se tiene o no un ángulo común de orientación de las crestas. Cuando no se tiene un ángulo común de las crestas en un bloque se puede afirmar que se tiene una singularidad. El proceso que se emplea es sencillo, ya que se calcula primero el gradiente horizontal y vertical de cada píxel usando operadores como Sobel[5]. Luego, se divide la imagen en pequeños bloques y se calcula el ángulo de la zona procesando la información de los pixeles del bloque. El mapa de orientación se puede apreciar en la figura 1.20.



(a) (b)

(c)

Figura 1. 20: Extracción del Mapa de Orientación. (a) Huella original. (b) Mapa de Orientación. (c) Ejemplo del cálculo del mapa de Orientación [16]

[5] Operador Sobe [17]: "Técnicamente es un operador diferencial discreto que calcula una aproximación al gradiente de la función de intensidad de una imagen. Para cada punto de la imagen a procesar, el resultado del **operador Sobel** es tanto el vector gradiente correspondiente como la norma de éste vector."

Los cambios considerables de ángulos en los mapas de orientación se clasifican en *CORE y DELTA*; éstos permiten detectar las singularidades y así clasificar las huellas dactilares además de ser un punto de referencia para alinear la huella dactilar durante el proceso de validación. El procesamiento de singularidades se basa en la comparación de los ángulos de los bloques previamente definidos mediante un método como Poincaré.

Para cada punto coordenado i,j del mapa de orientación se calcula el índice de Poincaré P(i,j) definido como la suma de la diferencia de la orientación de un punto i,j con sus vecinos. Esto se lo aprecia en la figura 1.21 (a). Si el índice de Poincaré es 360 o 180 el punto se lo conoce como CORE; si el índice es -180 el punto se lo conoce como DELTA y para cualquier otro caso el punto no representa una singularidad. Un ejemplo de CORE y DELTA se puede apreciar en la figura 1.21 (c).

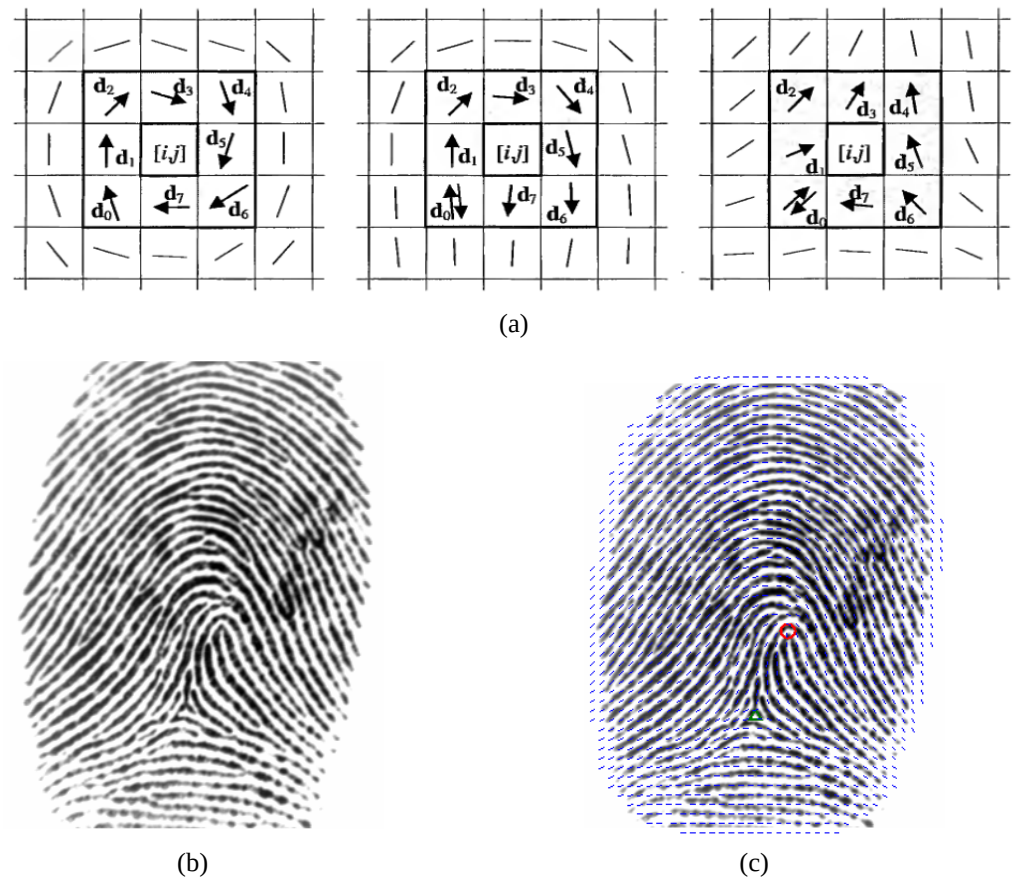


Figura 1. 21: Detección de singularidades – Método de Poincaré (a) Detección de Singularidades. (b) Huella Original. (c) Detección de Core (verde) y Delta (rojo) [16]

Aunque las singularidades pueden ayudar en la verificación de las huellas dactilares, éstas no son lo suficientemente precisas como para garantizar la identidad de un individuo y normalmente se las emplea como un parámetro adicional en la verificación con fines de orientación de la huella dactilar al momento de la verificación.

El parámetro que se emplea como verificador de huellas dactilares son *las minucias de la huella* que son los rasgos característicos de la huella. Para su extracción se tienen disponibles diferentes procesos, aunque el método tradicional y más empleado es el que se describe a continuación y consta de tres pasos (el diagrama de bloques se encuentra en la figura 1.22):

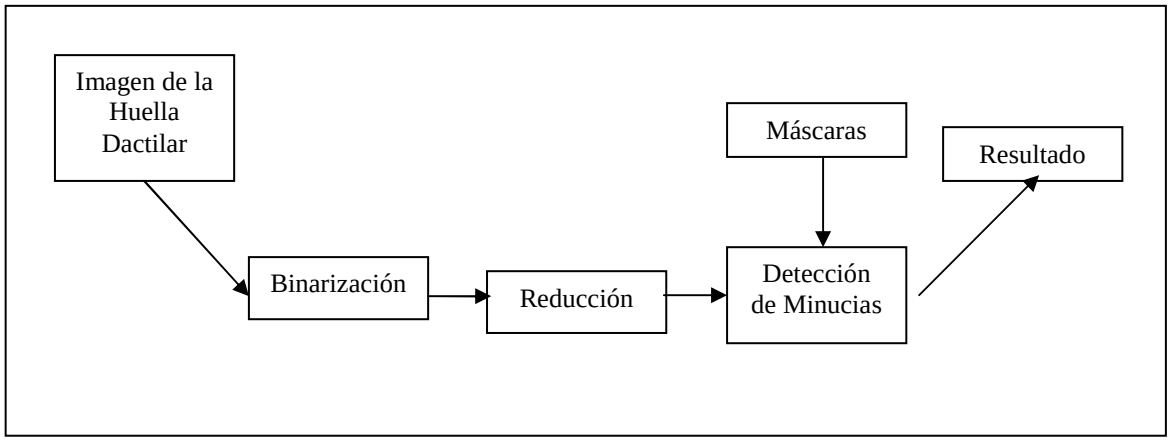


Figura 1. 22: Proceso de detección de minucias

- Binarización: Es el proceso mediante el cual se convierte la imagen que se encuentra en escala de grises a una imagen binaria.
- Reducción: Es el proceso mediante el cual el tamaño del ancho de las crestas se reduce a un ancho de un pixel, reduciendo sucesivamente las líneas conectadas hasta alcanzar dicho valor. En este proceso es importante tener el cuidado de mantener tanto la topología de la huella así como la consistencia de la misma.
- Detección de minucias: Es el proceso mediante el cual se detectan las minucias presentes en la huella dactilar a partir de la imagen reducida, la misma que se divide en bloques de 3x3 y mediante el uso de máscaras que contienen los patrones de minucias se las compara iterativamente para ver si existe o no una minucia en ese bloque.

Todas las minucias encontradas se las almacena en una matriz o plantilla indicando su posición, ángulo y tipo como una combinación para después compararla con la matriz o plantilla que se quiere verificar. Estas plantillas se conocen como plantillas de referencia (figura 1.23) y se almacenan en una base de datos junto con un identificador de pertenencia; ésta es la base de datos a la que se accederá para verificar si una huella corresponde o no a una persona.

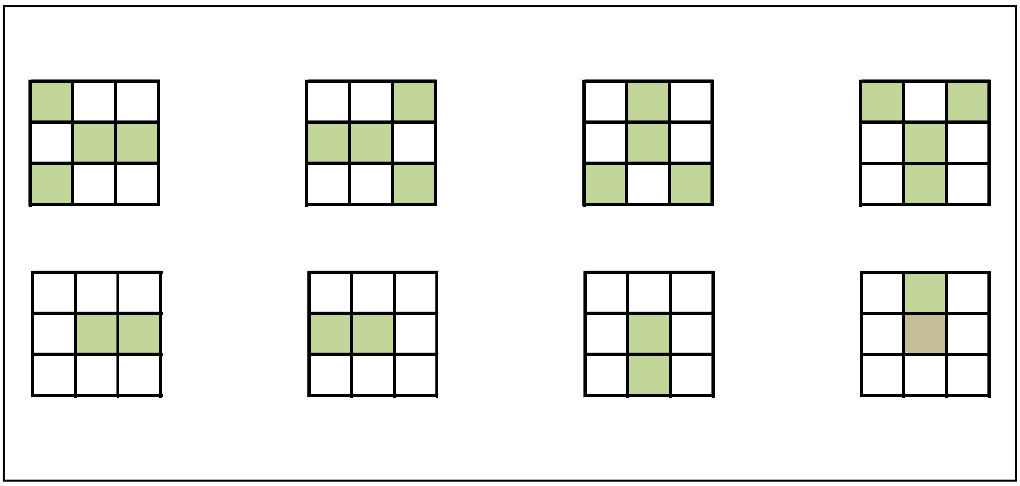


Figura 1. 23: Máscaras de patrones de minucias

En imágenes de baja resolución el proceso de detección de minucias se vuelve difícil por lo que también se emplea un método de detección de patrones, en el que se analiza el patrón de la huella dactilar a través del código del dedo, que no es más que un vector de la característica de la huella que se obtiene después de aplicar una convolución con el filtro de *Gabor* a la imagen original. En el vector se almacena la desviación estándar de cada una de las regiones. De manera estándar el filtro de Gabor tiene un tamaño de 32X32 y con 4 filtros direccionales se puede obtener las características globales de la huella, pero para realizar un proceso de verificación se necesitan 8 filtros.

Este método aunque sea bueno y útil para imágenes de baja calidad sigue siendo inferior y no tan seguro en la verificación frente a la detección de minucias.

Después de tener la plantilla de referencia almacenada en la base de datos y frente a una solicitud de verificación se realiza la verificación de la misma. Para este proceso es necesario realizar el mismo proceso que se indicó para detectar las minucias de la huella y poder compararlas con la plantilla de referencia.

Cuando se realiza la verificación de huellas no se comparan directamente los vectores o el número de los mismos, ya que por diferentes motivos (como por ejemplo tener diferentes imágenes para detectar las minucias) éstos no pueden coincidir, por lo que la verificación se la realiza en base a un rango de tolerancia en el que se analiza el grado de similitud y las características de cada huella. Durante este proceso, los algoritmos que permiten la comparación tienen que considerar factores como la rotación de la huella, la posición del dedo en el momento de obtener la imagen, la distorsión no lineal, el sobrelapamiento parcial de la imagen, el ruido, la condición de la piel, la temperatura, etc.

Los algoritmos basan su funcionamiento en tres tipos de verificación:

- *Correlación:* En este método las huellas se alinean y se realiza la verificación comparando cada uno de los pixeles de las dos imágenes. Como no se tiene datos sobre la rotación o desalineamiento de las imágenes se realiza el proceso de correlación para cada una de las posibles alineaciones de las imágenes de las huellas. En la actualidad los algoritmos que basan su funcionalidad en este método mediante un proceso previo, detectan las singularidades y las usan para aproximarse a una alineación que pueda ser la correcta. Los mayores desventajas que presenta este método son dos: una de ellas es la complejidad computacional que se requiere en el proceso de comparación y la otra de ellas es el grado bajo de tolerancia frente al ruido y a variaciones de contraste o distorsiones no lineales.
- *Singularidades:* Este método se basa en la comparación de las singularidades como patrones de las huellas dactilares empleando los vectores de código de huellas. Los algoritmos que emplean este método calculan la diferencia entre los vectores (de la huella de referencia y de la huella que se desea validar) y mientras menor sea la misma se puede considerar que las huellas son las mismas. Los inconvenientes de este método radica en el hecho que para poder comparar los vectores, éstos deben estar alineados y la validación de la huella no es altamente seguro puesto que se basan en singularidades y no en minucias. Normalmente, este método se lo emplea en combinación con los otros métodos para

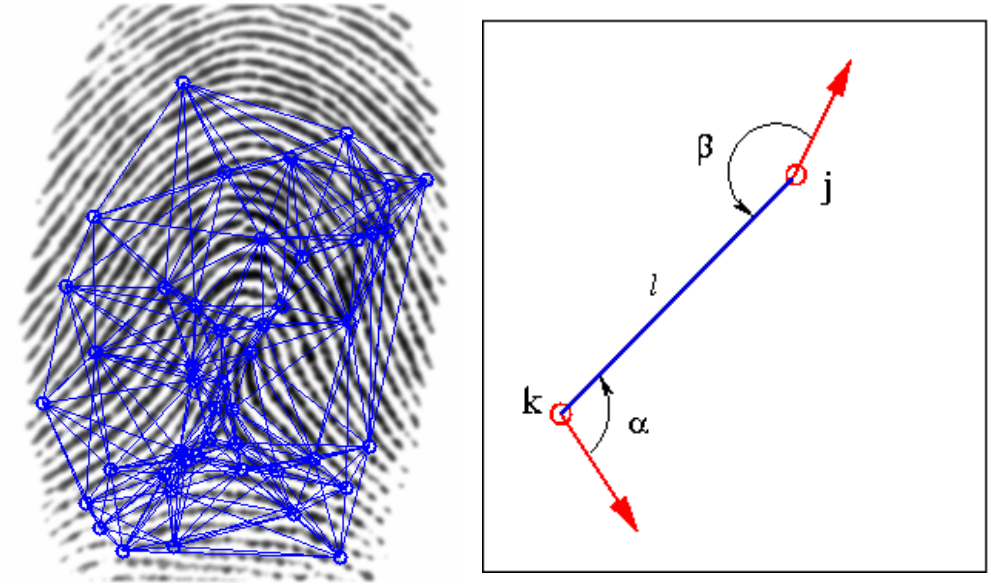
aumentar la seguridad en el proceso de verificación. En la figura 1.24 se presentan diferentes huellas con patrones diversos.

Figura 1. 24: Ejemplos de Patrones y Singularidades

- *Minucias:* Este método es el más seguro de todos y es el más difundido en los sistemas de verificación de huellas dactilares. Los algoritmos que emplean este método generan la plantilla de minucias para la huella que se desea verificar y la alinean con la plantilla de referencia que previamente ha sido almacenada en la base de datos. Una vez alineadas las dos plantillas los algoritmos buscan el número de coincidencias en minucias, en donde para dar como válida una coincidencia verifican la diferencia entre la distancia espacial y dirección entre dos minucias y la comparan con una tolerancia dada; si es menor se acepta como una coincidencia válida.

  Para garantizar el proceso y obtener el mayor número de minucias es importante que el algoritmo realice una correcta alineación de las plantillas, para esto se puede emplear el método descrito anteriormente,

además de realizar cálculos de traslación y rotación de la información así como de otras transformaciones geométricas como el escalado y la distorsión. Uno de los métodos empleados para el cálculo de la diferencia es la *formación por minucias*, en donde un segmento se forma por dos pares de minucias de la misma huella (las minucias entre las que se forma el segmento pueden ser seleccionadas entre varios criterios como por ejemplo las minucias más cercanas). Un ejemplo de la formación de los segmentos se encuentra en la figura 1.25 (a) mientras que en la figura (b) se encuentra un ejemplo de la formación de un segmento entre un par de minucias, en donde la distancia entre dos minucias se la representa con la letra *l* y los ángulos que se forman entre el segmento y la dirección de la minucia se la representa con alpha y beta.



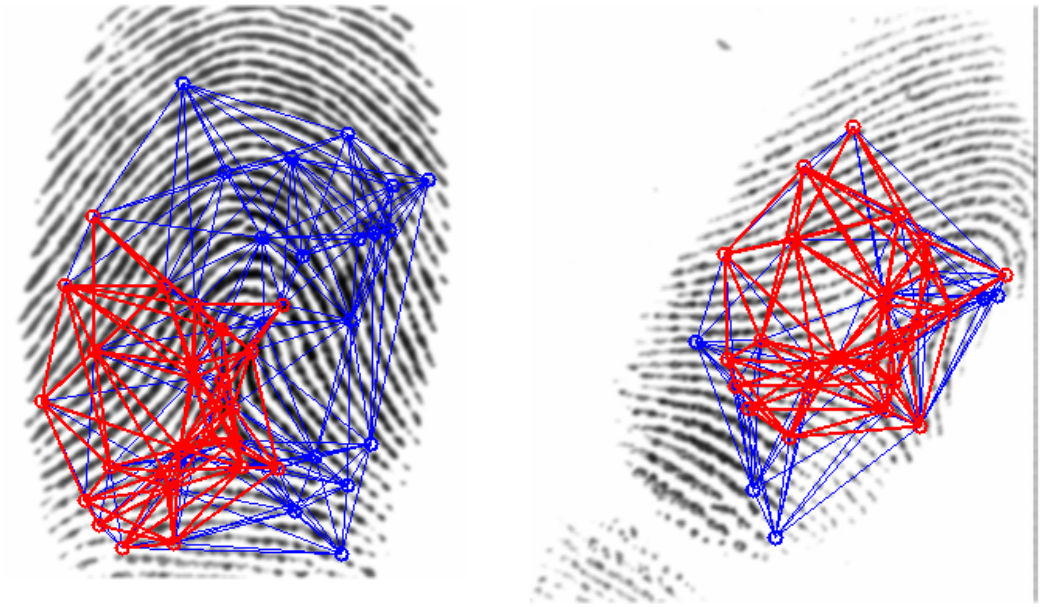
(a) (b)

Figura 1. 25: Minucias (a) Múltiples segmentos formados entre minucias.
(b) Representación de un segmento entre minucias [16]

El proceso de formación de segmentos se realiza tanto en la plantilla almacenada como en la plantilla que se quiere verificar. El proceso de verificación empieza comparando cada segmento de la plantilla que se desea verificar con todos los segmentos de la plantilla almacenada, para considerar si un par de segmentos

coinciden la diferencia entre las distancias y la diferencia entre sus ángulos debe ser menor a un límite dado. Si se encuentra un par de segmentos que cumplan con dicha regla se almacenan en una lista. Todas las diferencias de los ángulos se emplean para construir un histograma que permitirá encontrar el ángulo de rotación, el cual se define a partir de la selección de la zona de la huella que tiene el mayor número de segmentos coincidentes. Del mismo modo se construye un histograma en el cual se refleja la diferencia de la posición a través del desplazamiento de sus puntos en el plano XY para encontrar la traslación. A partir del ángulo de rotación y la traslación se pueden alinear las dos huellas y así verificar las dos huellas.

En el proceso de verificación se comparan la lista de los segmentos coincidentes, si superan un valor dado se acepta que las huellas son las mismas; caso contrario se rechaza la autenticación. En la figura 1.26 se presenta la superposición de los segmentos de la huella de referencia (a) y la huella que se consulta (b). Las marcas azules corresponden a los segmentos que se forman en la huella mientras que las marcas rojas corresponden a los segmentos coincidentes.



(a) (b)

Figura 1. 26: Coincidencias de Segmentos (a) Huella de Referencia. (b) Huella Consultada [16]

1.2.3.4 Software para Manejo de Huellas Dactilares

El software para manejo de huellas dactilares se puede clasificar en dos grandes grupos; el primero de ellos es el software comercial, es decir software que ya pone a disposición la tecnología para el usuario final, mientras que el segundo grupo es el software para desarrollo el cual permite crear aplicaciones comerciales (o aplicaciones de uso interno pero que ponen a disposición la tecnología de huellas dactilares a los usuarios). Este segundo grupo de aplicaciones se las conoce como SDK (*Software Development Kit*), empleándose en el presente proyecto SDK creado por Griaule Biometrics.

1.2.3.5 Problemas Potenciales

Como tecnología, el manejo de las huellas dactilares como método de verificación de identidad es robusto debido a los altos niveles de confiabilidad que presenta así como el fácil uso de sus dispositivos para la captura de huellas. Pero esta tecnología presenta principalmente 4 desventajas o problemas:

- *No todas las huellas se pueden almacenar:* Esto se debe a que aproximadamente el 2% de la población tiene huellas de baja calidad, es decir, que sus patrones de huella no son bien definidos. Esto sucede especialmente con personas ancianas.
- *Cambio de la huella dactilar con el tiempo:* Este inconveniente se da porque la huella sufre a lo largo del tiempo pequeños cambios debido a cortes o heridas. Para solucionar este inconveniente es necesario reingresar la huella de una persona cada cierto tiempo a la base de datos.
- *Dispositivos especializados:* Debido a que los sistemas biométricos basados en huellas dactilares no son ampliamente difundidos, la adquisición de dispositivos lectores aún se la tiene que realizar en lugares especializados y sus costos siguen siendo altos debido a la falta de masificación de la tecnología.
- *Asociación con aplicaciones forenses o legales:* Esta desventaja se produce ya que muchas personas asocian las huellas dactilares con aplicaciones forenses o legales, lo que causa el temor o la disconformidad por parte.

1.2.3.6 Dispositivos Existentes

Comercialmente existen diferentes empresas que fabrican dispositivos lectores de huellas dactilares, siendo los modelos muy variados dependiendo del ámbito en el que se vayan a emplear; por ejemplo existen dispositivos diseñados para interiores y para exteriores, o dispositivos de mayor o menor precisión por el tipo de seguridad que se desea implementar. Pero en general estos dispositivos se basan en 5 tecnologías que son las más comunes, cada una con sus pros y contras.

- *Lectores basados en el Efecto Piezoeléctrico:* Este tipo de dispositivos se conforman por una matriz de sensores de presión los cuales generan una señal eléctrica a partir de una fuerza mecánica. Debido a los valles y crestas presentes en la huella dactilar, la presión que se ejerce sobre la matriz de sensores es diferente, lo que produce una imagen de la huella dactilar. La principal desventaja de este método se basa en su tipo de funcionamiento, debido a que depende de la presión que se puede o no ejercer, por lo que la imagen de la huella es de baja calidad y en ocasiones puede ser borrosa o distorsionada.
- *Lectores basados en Ultrasonido:* Al igual que un sonar, los dispositivos que emplean esta tecnología usan señales acústicas que se envían contra la huella, capturando el eco que se produce y representando a través del eco la estructura de la huella dactilar. Esta tecnología tiene dos desventajas grandes por las cuales no es empleada comercialmente, una de ellas es el costo de los dispositivos y la otra es el ruido de la imagen que se produce por las regiones grasosas propias de la piel humana así como la resequedad de la piel.
- *Lectores basados en Capacitores:* Los lectores que emplean esta tecnología se componen de chips semiconductores que tienen una matriz celdas; las cuales al acercar el dedo producen pequeñas descargas eléctricas y debido a las diversas distancias entre los valles y crestas producen diferentes patrones capacitivos. El sensor que posee el lector procesa las respuestas de las celdas generando así la imagen de la huella.

- *Lectores Térmicos:* Estos dispositivos se conforman por celdas piro eléctricas, las cuales son las encargadas de detectar las diferentes temperaturas producidas por la huella (esta diferencia de temperatura se produce porque los valles y las crestas se encuentran a diferentes distancias de las celdas). La diferencia de temperatura es interpretada por el lector produciendo así una imagen de la huella. La desventaja de este tipo de lectores es que se necesita un tiempo para que sus celdas se calienten y su operación degrada el tiempo de vida del sensor.
- *Lectores basados en la Reflexión interna Total Óptica (FTIR):* Esta tecnología es también conocida como óptica y es la más empleada comercialmente. Normalmente se la emplea en ámbitos en donde se necesita la huella en tiempo real. Un lector que emplea este método tiene un prisma el cual refleja un láser hacia su superficie que es la que está en contacto con la huella. El encargado de capturar la luz reflejada es un CCD (*Carge Cupled Device*) o un CMOS (*Complementary Metal Oxide Semiconductor*) el cual procesa los diferentes patrones de luz reflejada. Según el patrón de reflejo se pueden tener zonas de mayor o menor luminosidad, que dependen de los valles o crestas, produciendo así una imagen de la huella. La principal desventaja de estos lectores es la distorsión de la imagen que se puede generar cuando el láser no es enfocado adecuadamente, aumentando este efecto cuando no existe la distancia correcta entre el prisma y el sensor, lo que da a lugar una desventaja adicional, estos dispositivos no pueden ser pequeños.

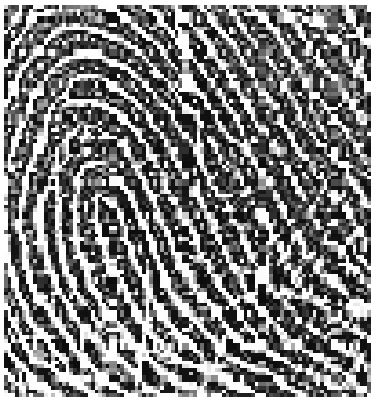
En la figura 1.27 se muestran algunos ejemplos de huellas dactilares capturados por diferentes tipos de lectores.

1.2.4 APLICACIONES

Existe una amplia gama de aplicaciones en donde se pueda utilizar las características biométricas del ser humano, generalmente se las emplea como mecanismo de seguridad, pero otro uso común es como método de identificación de personas. A continuación se indicarán algunos de los ámbitos en donde comúnmente se las puede encontrar y las aplicaciones de cada ámbito:

- *Aplicaciones Gubernamentales*
- *Aplicaciones Legales*
- *Aplicaciones Comerciales*
- *Aplicaciones Médicas*

Lector Óptico



Lector Eléctrico

Lector Térmico

Lector Capacitivo

Figura 1. 27: Ejemplos de imágenes obtenidas de diferentes lectores [16]

1.3 BASE DE DATOS

El sistema de base de datos en una organización constituye una parte fundamental en sus sistemas de información; su función es servir de fundamento a dichos sistemas para la gestión corporativa.

En la actualidad se rescata la importancia de la información en las estructuras organizacionales y se pueden definir ciertas cualidades inherentes a la información que se deben garantizar en todo momento y éstas son:

- *Precisión:* Proporción de información correcta.
- *Oportunidad:* Tiempo transcurrido desde que se produjo el hecho que originó un dato hasta el momento en que dicho dato está a disposición del usuario.
- *Compleción:* La información debe estar completa para cumplir con los fines para los que se gestiona.
- *Significado:* El contenido semántico de los datos debe ser el adecuado para facilitar la asimilación por parte del usuario.
- *Coherencia:* La información que se almacena debe ser consistente en sí misma y con las restricciones semánticas del ámbito al que representa.
- *Seguridad:* Se debe proteger la información de los accesos no autorizados así como de su deterioro (físico o lógico). La seguridad debe abarcar tres aspectos que a continuación son descritos (se profundiza en el tema de seguridad en la sección 1.7):
    - *Confidencialidad:* La información solo puede ser accesible por los usuarios que están autorizados.
    - *Disponibilidad:* La información debe estar disponible para cuando el usuario desee usarla.
    - *Integridad:* La información debe mantener su coherencia, y no puede ser modificada por usuarios no autorizados.

Un sistema de información se diseña para garantizar las cualidades de la información previamente descritas, así como para satisfacer las necesidades de

información de la organización. El sistema de información toma datos de la organización, de fuentes externas y de los resultados de las operaciones sobre esos datos, los almacena y los pone a la disposición de los usuarios para su adecuada utilización.

1.3.1 CONCEPTOS

- *Datos:* Información útil para una organización.
- *Datos Referenciales:* Datos que contienen información de referencia para saber donde se almacenó la información.
- *Datos Factuales:* Datos que contiene información en sí.
- *Datos Estructurados:* Datos que tienen una estructura bien definida. Son del tipo de datos factuales.
- *Datos No Estructurados:* Datos que por sus características no tienen una estructura definida y son del tipo de datos factuales.
- *Base de Datos:*

  > *"Una base de datos es una colección de información estructurada según un cierto modelo…*
  >
  > *Este modelo de datos, o esquema, define las unidades mínimas de información que podrán almacenarse y las relaciones que entre ellas existen. El modelo se explicita al crear la base de datos abstrayendo las características de cada entidad de datos y las relaciones y dependencias entre las mismas."* [7]

- *Sistema de Gestión de Base de Datos (SGBD):* Conjunto de procedimientos, estándares, métodos, lenguajes y programas que actúan como intermediarios entre los usuarios, las aplicaciones y los datos, proporcionando los medios para utilizar (describir, almacenar, recuperar y manipular) la información, garantizando sus cualidades.
- *Abstracción:* Es la acción de capturar lo relevante y eliminar lo insignificante de un detalle relacionado a los objetivos de la organización.

- *Modelo:* Es un conjunto de abstracciones que forman una visión parcial útil de una parte del mundo.
- *Modelo de Datos:* Es una representación de la información que la organización necesita, en donde se representa y se describen los datos, sus relaciones, su concepto, sus limitantes y su semántica.
- *Esquema de la Base de Datos:* Diseño lógico de la base de datos.
- *Instancia de la Base de Datos:* Información contenida en la base de datos en un determinado momento.

1.3.2 MODELOS DE DATOS

A lo largo de la teoría de las bases de datos se han propuesto diferentes modelos de datos, y se los podría clasificar en tres grupos: *los modelos lógicos basados en objetos, los modelos lógicos basados en registros y los modelos físicos de datos.*

Para describir cada uno de estos modelos lógicos se han creado tres niveles de abstracción (presentados en la figura 1.28) y éstos son:

- *Nivel de Visión:* En este nivel se procura describir una parte de la base de datos, para simplificar la interacción entre los usuarios y el sistema.
- *Nivel Conceptual:* En este nivel se procura describir las relaciones que existen entre los datos, especificando cuáles de ellos están almacenados.
- Nivel Físico: En este nivel se procura describir cómo se almacenan los datos, detallando las estructuras de los mismos.

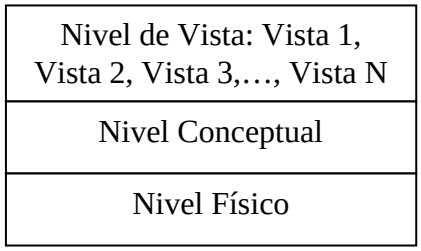


Figura 1. 28: Niveles de Abstracción

1.3.2.1 Modelo de Red

La principal característica de este modelo es que los datos se expresan por registros y sus relaciones mediante ligas. La parte estructural de los datos se representan en forma de grafo. En este modelo los datos se manipulan de registro en registro y se asume que la estructura interna de la base de datos es conocida por el programador.

El Modelo de Red más difundido y más representativo es el *Modelo en Red CODASYL (Conference on Data Systems Languages)*. Este modelo fue desarrollado por DBTG (Grupo de trabajo de base de datos) en la década de los 60's y constituye la primera especificación estándar para una base de datos.

En la especificación CODASYL se manejan dos estructuras de datos fundamentales: los *tipos de registros* que se definen como colecciones de elementos de datos lógicamente relacionados y los *conjuntos* que expresan uno de dos tipos de interrelaciones: uno a muchos (1: N), o uno a uno (1: 1).

Cada registro está compuesto por *campos* que es la unidad de datos más pequeña que puede ser representada en este modelo. En los conjuntos se definen dos tipos de nodos, los *propietarios* que son los nodos raíces y los *miembros* que son los nodos dependientes. Un conjunto tiene un registro propietario y uno o más registros miembros. Adicionalmente un registro puede pertenecer a diferentes conjuntos como propietarios o como miembros. A las interrelaciones se las conoce como ligas y estas relaciones son del tipo binarias.

CODASYL esquematiza el diseño de la base de datos mediante dos componentes básicos: *Cuadros* que corresponden a los registros y *Líneas* que corresponden a las ligas. Este modelo se encuentra ejemplificado en la figura 1.29.

1.3.2.2 Modelo Jerárquico

El modelo jerárquico puede ser considerado como un caso particular del modelo en red. En este modelo al igual que el modelo CODASYL se definen registros y ligas con los cuales se representan los datos y sus relaciones. La diferencia radica en que los registros se organizan para formar conjuntos de árboles y no gráficas arbitrarias.

Se define un *árbol* como una estructura que se organiza por niveles, en donde el registro de mayor jerarquía se conoce como *raíz*. La estructura jerárquica no puede contener ciclos, un nodo de nivel superior (nodo padre) puede tener un número ilimitado de nodos hijos, pero un nodo hijo solo puede tener un nodo padre. En cada diagrama existe únicamente una instancia del árbol de base de datos, es decir, no se admite más de una relación entre dos segmentos.

En este modelo se definen estructuras recursivas, en donde el árbol tiene que recorrerse en un preorden comenzando necesariamente en el nodo raíz. Un ejemplo de este modelo se muestra en la figura 1.30.

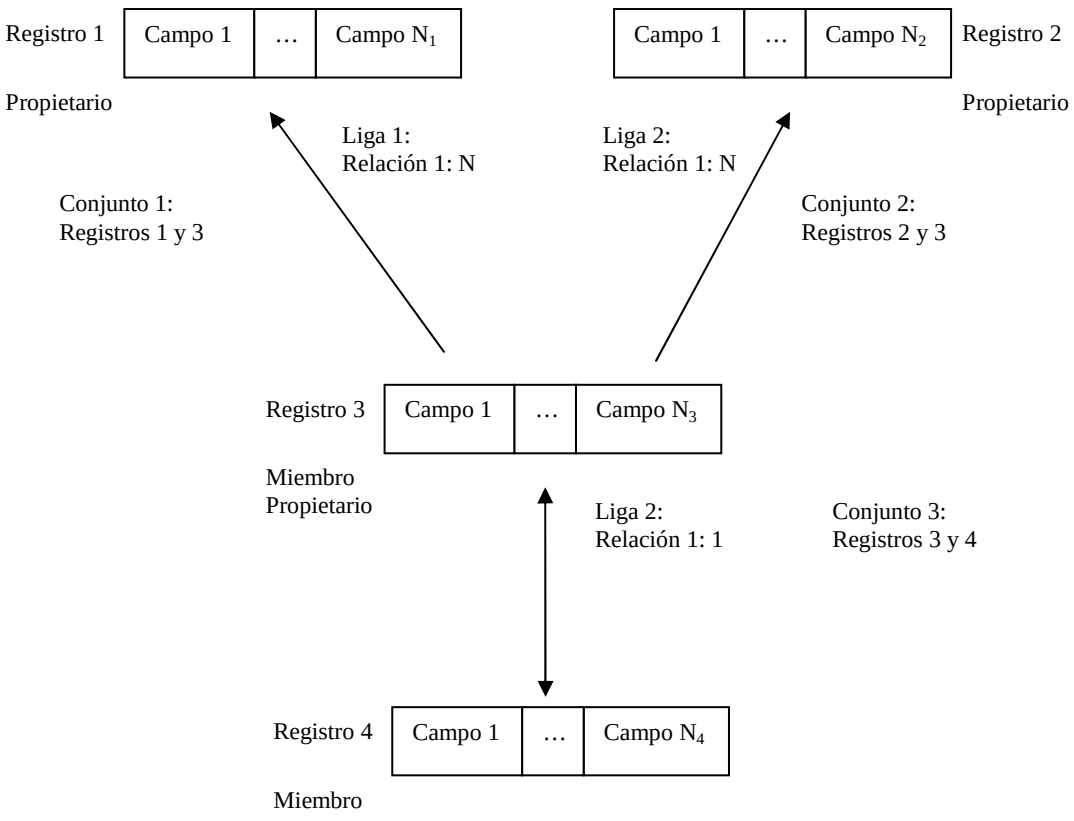
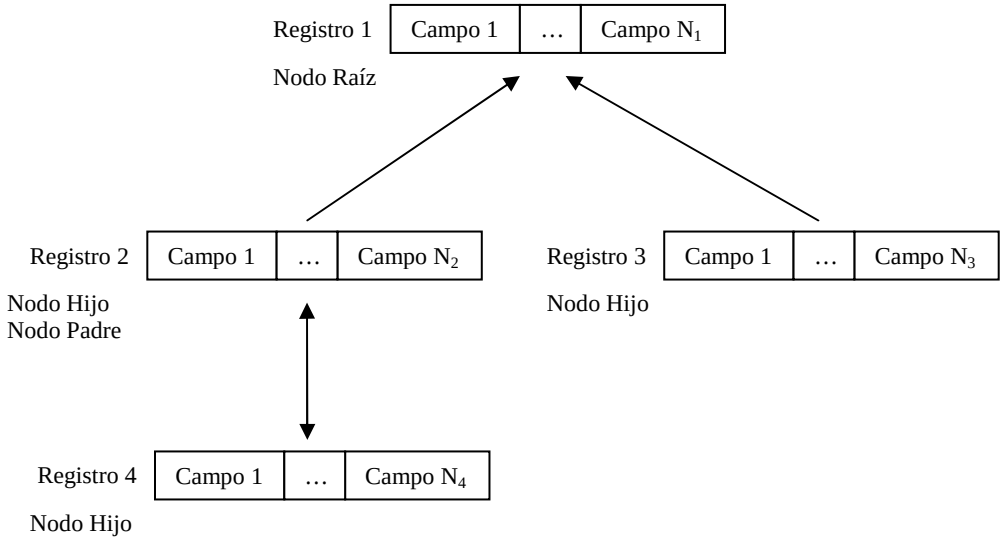


Figura 1. 29: Ejemplo de Diagrama de Estructura de Datos del Modelo de Red CODASYL



Árbol 1

Figura 1. 30: Ejemplo de Diagrama de Estructura de Datos del Modelo Jerárquico

1.3.2.3 Modelo Entidad - Relación

El modelo Entidad - Relación intenta hacer una abstracción del mundo real a través de objetos básicos llamados *entidades* y las *relaciones* entre estos objetos. Esta percepción del mundo real como una serie de objetos relacionados entre sí pretende ser representada gráficamente mediante un mecanismo de abstracción basado en una serie de símbolos, reglas y métodos. Esta representación gráfica muestra los datos de interés del mundo real. Se puede decir que el modelo Entidad - Relación fue creado como una metodología gráfica para el diseño de base de datos.

La robustez de este modelo se debe a dos aspectos: su potencia para representar prácticamente todas las restricciones posibles del diseño de datos, y su flexibilidad para admitir la evolución en el tiempo del sistema de información diseñado. Esta robustez se debe a que en una organización el sistema de información permanece casi invariable en lo que a sus datos de interés se refiere, mientras que es cambiante el tratamiento dado a los mismos. En el modelo Entidad - Relación se logra obtener una representación más natural de los datos, lo que se consigue al separar los objetos de sus asociaciones, así como al intentar mantener un alto grado de independencia entre los datos y su

tratamiento, y al establecer un cierto grado de jerarquía entre los distintos elementos componentes del modelo.

El modelo Entidad - Relación maneja estructuras propias que son denominadas Diagramas Entidad/Relación en donde se representan gráficamente los elementos propios del modelo como son: las entidades, los atributos, los conjuntos de entidades, y las relaciones.

Se define una *entidad* como un objeto que existe (real o abstracto) que se puede distinguir entre otros objetos; este objeto es de interés para la organización ya que de él se puede obtener cierta información característica o propiedades que son conocidas como *atributos*. De esta manera, se puede decir que un conjunto de atributos representan una entidad. Cada atributo tiene un rango de valores permitidos conocido como *dominio*. La *ocurrencia de entidad* es la realización concreta de una entidad.

Una entidad debe cumplir con las siguientes reglas:

- Debe tener existencia propia.
- Cada ocurrencia de un tipo de entidad debe poder distinguirse de las demás.
- Todas las ocurrencias de una entidad deben tener los mismos atributos.
- Cada entidad debe tener un atributo o un conjunto de atributos que se conoce como *Atributo Identificador Principal (AIP)*, que sirve para identificar unívocamente cada una de las ocurrencias de esa entidad.

Cuando existe un conjunto de atributos que sirven para verificar la condición de identificador unívoco, cada atributo se le conoce como *Atributo Identificador Candidato (AIC),* de los cuales se escoge un AIP y el resto de los atributos se les conoce como *Atributos Identificadores Alternativos (AIA)*.

Una *relación o interrelación* es una asociación entre entidades y se caracterizará por determinadas restricciones que determinan las entidades que pueden o no participar de dicha relación. Cada relación viene caracterizada por tres propiedades: el nombre que identifica la relación unívocamente, el grado que es el número de tipos de entidades sobre las que se realiza la asociación, y el tipo de

correspondencia que representa el número máximo de ocurrencias de cada tipo de entidad. Los tipos de correspondencia válidos para este modelo son: una a una, una a muchas, muchas a una y muchas a muchas.

Los diagramas Entidad - Relación, que representan gráficamente el esquema conceptual de la base de datos utilizan los elementos señalados en la figura 1.31.

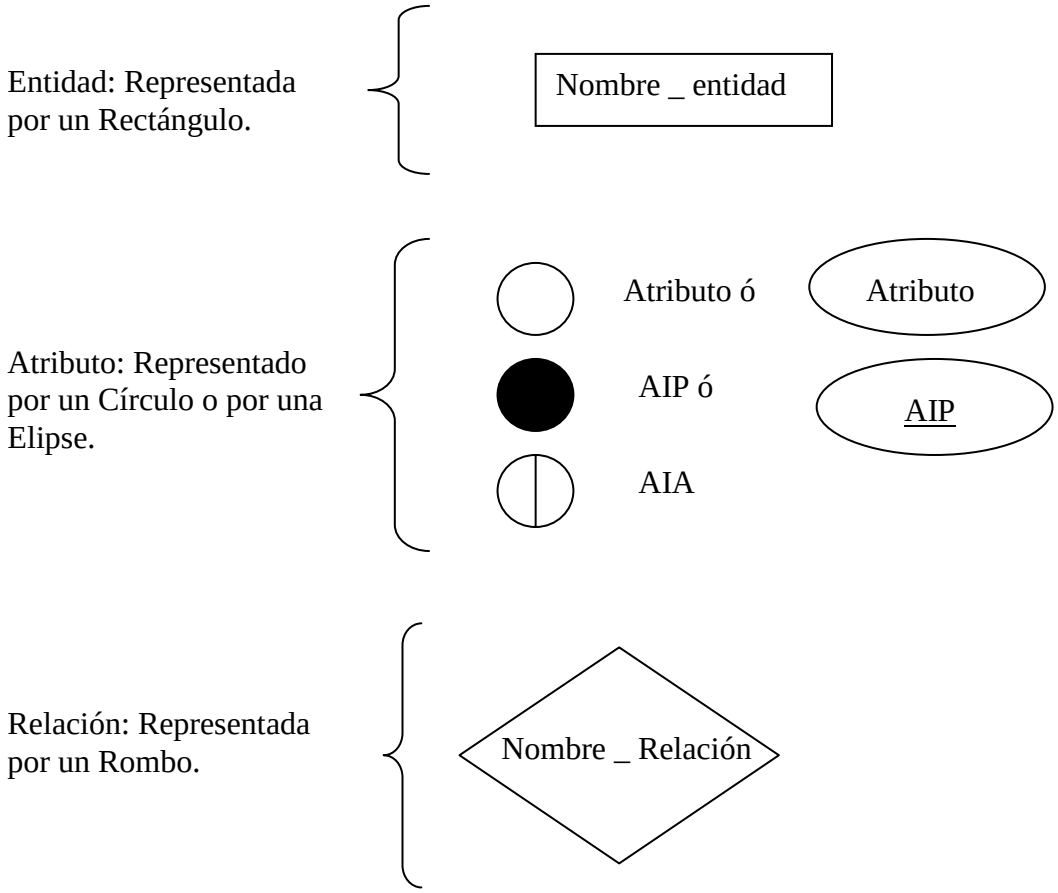


Figura 1. 31: Símbolos empleados los diagramas de Entidad - Relación

Adicionalmente a los símbolos que se representan en la Figura 1.31, se emplean las líneas que conectan los atributos a las entidades, y las entidades a las relaciones. Otro aspecto que se debe tener en cuenta es que las relaciones pueden tener atributos propios que permitan describir de mejor manera la relación, y cada relación debe tener un AIP. Un ejemplo genérico de cómo se modela en este tipo de diagramas se encuentra en la figura 1.32.

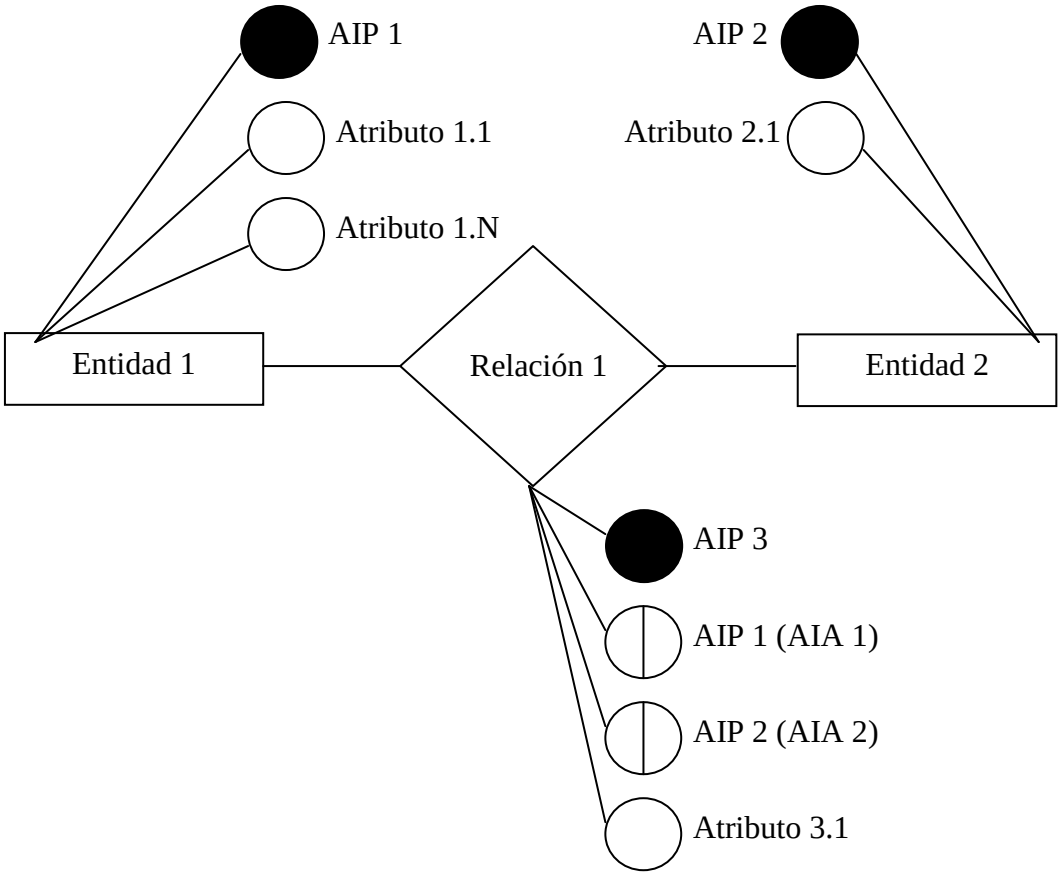


Figura 1. 32: Ejemplo de Diagrama de Entidad - Relación

1.3.2.4 Modelo Relacional

En este modelo los datos son representados lógicamente a través de estructuras en forma de relaciones mediante el uso de un conjunto de tablas; se tiene como una correspondencia directa el concepto de la tabla con el concepto matemático de una relación.

Uno de los objetivos básicos de este modelo es mantener la *independencia física* de tal manera que la forma en que se almacenan los datos no altere la manipulación lógica, logrando de esta manera que no sea necesario la modificación de las aplicaciones por cambios en el almacenamiento físico; otro objetivo es la *independencia lógica* que intenta no repercutir en las aplicaciones o en los usuarios que acceden a los objetos de la base de datos por modificaciones, añadiduras o eliminación de los mismos. L*a flexibilidad* para poder presentar a los

usuarios los datos de la forma en que ellos necesiten, es otro de los objetivos que se pretende alcanzar, así como mantener *la uniformidad* en las estructuras lógicas de los datos para facilitar la visión conceptual de la base de datos.

El elemento básico que se define en este modelo es la *relación,* que se la puede representar en forma de tabla. En esta relación o tabla se pueden representar *atributos* que no son más que un conjunto de columnas que representan características o propiedades de la relación; cada atributo tiene un nombre y un conjunto de *tuplas* que son filas que representan ocurrencias de la relación (las tuplas son instancias de la base de datos, por lo tanto son variantes en el tiempo). El número de tuplas existente en una relación se conoce como *cardinalidad* y el número de atributos se conoce como *grado de la relación.* De manera similar al modelo Entidad - Relación, cada atributo tiene un rango o tipo de valores permitidos al cual se lo conoce como *dominio.* Cada relación debe tener un nombre descriptivo que la caracterice y en él es posible distinguir una cabecera que define la estructura de dicha relación (en donde se representan los atributos) y un cuerpo que se conforma por las tuplas.

El dominio de un atributo está caracterizado por un nombre y debe ser un conjunto finito de valores homogéneos (porque todos sus elementos son del mismo tipo) y atómicos (porque sus elementos son indivisibles en lo que al modelo se refiere). Todo dominio tiene un tipo de datos y es el mismo para todos sus valores). Existen instancias de la base de datos en la cual un valor de un atributo en un momento dado puede tener un valor desconocido, asociándose ese valor desconocido a un valor nulo para el atributo. No se debe confundir un valor nulo con un valor numérico igual a cero o una cadena de caracteres vacía.

En la noción de relación se manejan dos conceptos básicos:

- *Intención de la Relación:* también conocida como esquema de la relación y representa la cabecera de la tabla como una parte definitoria y estática de la relación.
- *Extensión de la Relación:* también conocida como ocurrencia o instancia de la relación y son los conjuntos de tuplas que existen en un instante dado. Estos valores deben satisfacer la intención de la relación.

En la Figura 1.33 (a) se representa la intención de la relación mientras que en la Figura 1.33 (b) se ejemplifica la extensión de la relación.

Relación_1 (Atributo_1: Tipo_de_Datos, Atributo_2: Tipo_de_Datos,…, Atributo_N: Tipo_de_Datos)

(a)

Relación_1

Atributo_1	**Atributo_2**	**…**	**Atributo_N**
Dato_1.1	Dato_2.1	…	Dato_N.1
Dato_1.2	Dato_2.2	…	Dato_N.2
Dato_1.3	Dato_2.3	…	Dato_N.3

(b)

Figura 1. 33: Representación de la intención y la extensión de la relación

En el modelo relacional existe un atributo que de manera similar al atributo identificador principal (AIP) del modelo Entidad - Relación sirve para identificar unívocamente cada tupla; este atributo se lo conoce como *clave primaria* y se lo selecciona de un conjunto de *claves candidatas.* La clave primaria no puede ser nula. Las claves candidatas que no han sido escogidas como clave primarias se conocen como *claves alternativas*. En este modelo se define también las *claves ajenas* que es un conjunto de atributos (o un atributo) de una relación cuyos valores coinciden con los valores de la clave primaria de otra relación.

El modelo relacional presenta ciertas restricciones que son inherentes al diseño y a la conceptualización propia del modelo y la base de datos; éstas son:

- Ningún atributo que forme parte de la clave primaria de una relación puede tener un valor nulo. Esta regla se conoce como *integridad de entidad.*
- Cada atributo puede tomar un solo valor del dominio. No se admiten los grupos repetitivos.

- El orden de las tuplas y/o de los atributos no es significativo.
- No pueden existir dos tuplas iguales.

En el modelo relacional solo pueden existir tres tipos de interrelaciones: *Relación uno a muchos (1: N), Relación muchos a uno (N: 1) y Relación uno a uno (1:1).* Las relaciones del tipo *muchos a muchos (N: M)* no pueden existir, para la representación de este tipo de relaciones se crea una nueva tabla (relación) que contendrá como atributos obligatorios en forma de clave ajena, las claves primarias de las relaciones correspondientes. La figura 1.34 ejemplifica un diagrama genérico de este modelo de datos.

En el modelo relacional existe un álgebra relacional que permite analizar los cambios de estado que se especificaron mediante operaciones cuyos operandos son relaciones y su resultado es otra relación. Se puede definir al álgebra relacional como un lenguaje de consulta de procedimientos, que se basa en un sistema cerrado de operaciones definidas sobre relaciones. En el álgebra relacional existen cinco operaciones fundamentales; a continuación se listan las 5 operaciones fundamentales y una operación derivada que constituyen la base del álgebra relacional.

- *Elegir o Restricción:* Esta operación se representa con la letra griega sigma minúscula (σ) para señalar la selección. Mediante esta operación se obtiene un conjunto de tuplas que satisfacen con una expresión lógica o predicado de elección.
- *Proyección:* Esta operación se representa con la letra griega pi minúscula (π) y como subíndice se escriben los atributos que se desea que aparezcan en el resultado. Mediante esta operación se obtiene una relación definida sobre atributos de una relación eliminando las filas repetidas.
- *Producto Cartesiano:* Se representa mediante una cruz (X), es una operación binaria ya que involucra dos relaciones (tablas). Mediante esta operación se puede combinar información de varias relaciones.

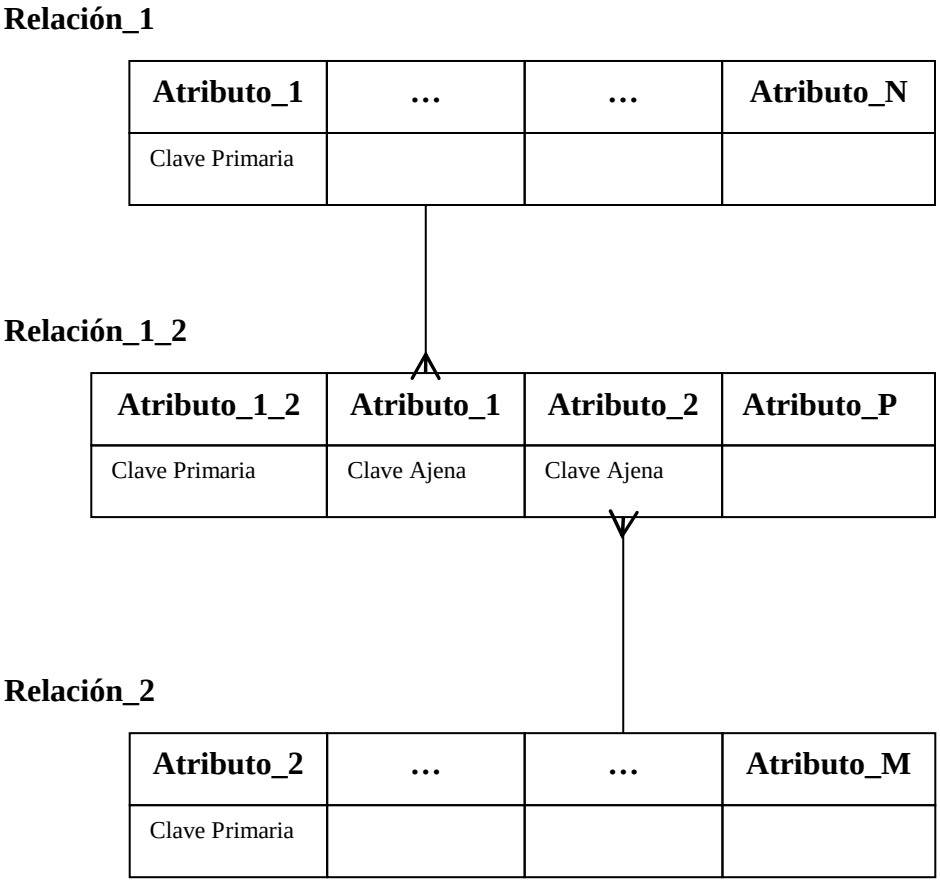


Figura 1. 34: Ejemplo de Diagrama de Estructura de Datos del Modelo Relacional

- *Unión:* Esta operación se representa mediante el símbolo de unión de la teoría de conjuntos (∪), siendo una operación del tipo binaria. Mediante la unión de dos relaciones que sean compatibles en su estructura se produce otra relación que tendrá las tuplas que aparecen en cualquiera de las dos relaciones o en ambas.
- *Diferencia:* Se representa mediante el signo menos (–), es una operación binaria que permite encontrar las tuplas que pertenecen a una relación pero no a la otra.
- *Intersección:* Esta operación es derivada (ya que se la puede obtener en base a las operaciones anteriormente descritas) y se la representa mediante el símbolo de intersección de la teoría de conjuntos (∩). Esta operación es de tipo binaria y devuelve como resultado una relación que contiene las tuplas que se encuentran en las dos relaciones.

1.3.3 SQL

El álgebra relacional permite representar las consultas de manera matemática y en forma concisa. Para el programador manejar este tipo de lenguaje para realizar consultas resulta complicado, por tal motivo se han creado lenguajes comerciales que facilitan la programación y desarrollo de las bases de datos. Existen tres lenguajes que son los más difundidos y éstos son: *Quel, QBE y SQL.* De estos tres, SQL (lenguaje de consultas estructurado) es el más importante ya que es el lenguaje base para la implementación de modelos relacionales.

SQL define ciertos tipos de datos propios y a partir de éstos el usuario puede crear sus propios tipos de datos. Los datos manejados por SQL son:

- Alfanuméricos: *Char*(n) y *VarChar*(n).
- Numéricos: *Int*, *Decimal*, *Real*.
- Fechas y Horas: *Date*, *Time*, *TimeStamp*.
- Otros: *Binary*, *VarBinary*.
- Definidos por el Usuario: Que se los define mediante los tipos de datos antes señalados y se los crea mediante la sentencia: *Create DataType.*

El lenguaje SQL está especificado de tal manera que cada sentencia se compone de cláusulas que comienzan con una palabra reservada y que generan una nueva relación que contiene el resultado de la consulta. Las tres palabras reservadas básicas para realizar una consulta son: *select, from y where.*

- *Select:* Corresponde a la operación proyección del álgebra relacional. Permite listar los atributos de las tablas involucradas de la consulta.
- *From:* Permite especificar las relaciones desde las cuales se extraerá la información a consultar.
- *Where:* Mediante esta cláusula se puede definir el predicado o las restricciones de la consulta.

En SQL se pueden encontrar otras operaciones del álgebra relacional como: *union (unión), intersect (intersección) y minus (diferencia).* Para aplicar estas operaciones se tienen que definir las relaciones involucradas mediante una

consulta (empleando select, from y where) y entre paréntesis. Para evitar confusiones por nombre de atributos similares SQL permite diferenciarlos de la siguiente manera: *nombre_relación.nombre_atributo*.

En SQL también se definen operadores lógicos para especificar los predicados de la cláusula where. Los operadores aceptados son *and, or y not.* Cuando un predicado o restricción tiene que ser analizado en un rango se utiliza el conector *in.* Las cláusulas *group by y order by* permiten al usuario ordenar la presentación de las tuplas resultado de la consulta. SQL adicionalmente permite calcular a través de ciertas funciones algunos valores: el promedio de un atributo se lo calcula mediante *avg*, el valor mínimo mediante *min,* el máximo con *max,* el valor total de todos los atributos con *sum,* y el número de tuplas afectadas con *count.* Para eliminar tuplas repetidas se emplea la cláusula *distinct* después de la cláusula *select*

La creación de relaciones en SQL se realiza mediante la sentencia: *Create Table nombre relación,* a continuación de esta sentencia entre paréntesis se define la relación especificando el nombre de cada atributo, el tipo de datos y si el atributo permite o no valores nulos. Es necesario que por cada relación creada se especifique cuál es la clave primaria (mediante la sentencia *primary key*) y en caso de que existan, las claves foráneas (mediante la sentencia *foreign key*). Para modificar una relación se emplea la sentencia *Alter Table nombre relación* y a continuación entre paréntesis se redefine la relación.

La inserción de datos en SQL se la realiza con la cláusula *insert,* y la información tiene que ser especificada según el siguiente esquema:

INSERT INTO nombre_relación [(lista columnas)] {**VALUES** (expresión [, expresión]…}

Para actualizar las tuplas en una relación se emplea la cláusula *update,* con el siguiente formato:

UPDATE nombre_relación **SET** atributo_x = {valor_x | NULL | sentencia_SELECT} [, atributo_y = {valor_y | NULL | sentencia_SELECT}],…

[**WHERE** predicados]

Para eliminar una o más tuplas se debe emplear la cláusula *delete,* de la siguiente manera:

DELETE [FROM] nombre_relación [**WHERE** predicados]

SQL ofrece varios niveles y tipos de seguridades; esta seguridad se la obtiene con el manejo de *vistas* que son relaciones (tablas) virtuales que se le presenta al usuario para que no acceda por completo al modelo conceptual. Para trabajar con vistas a todas las sentencias anteriores en vez de utilizar la palabra reservada *table* se emplea la también palabra reservada *view.* Otro tipo de seguridad que implementa SQL es el manejo de las cláusulas *grant (permitir)* y *revoke (denegar)* para asignar permisos de selección, inserción, actualización y eliminación en base a listas de usuarios.

1.3.4 SERVIDORES DE BASE DE DATOS

La arquitectura más común para el manejo de la información en una organización es la de Cliente – Servidor; en esta arquitectura las transacciones se dividen en procedimientos independientes que colaboran entre sí para el intercambio de información o datos. En esta arquitectura se distinguen dos actores: el *cliente* que es la aplicación o el proceso que inicia el diálogo para realizar una solicitud de información o recursos al *servidor,* que es el encargado de responder a dichas solicitudes.

Las principales características de esta arquitectura son:

- Los clientes acceden a un punto de servicio bien definido y único, que constituye la interfaz externa del servidor.
- Los clientes no tienen que preocuparse por la estructura interna, la lógica, la ubicación, el sistema operativo ni el hardware del servidor ya que los clientes acceden únicamente a la interfaz externa del mismo.
- Los cambios que se realicen en el servidor no influyen o cambian el comportamiento del cliente, ya que solo tiene que adaptarse la interfaz externa del servidor.
- Para el cliente es transparente si está accediendo a un servidor o a una granja de servidores para obtener la información que necesita.

En la actualidad los servidores de base de datos manejan dos modelos, el modelo Entidad - Relación y el modelo Relacional, de los cuales el más aceptado y difundido es el modelo relacional.

1.3.4.1 Arquitecturas del Esquema Cliente - Servidor

Esencialmente se tienen dos arquitecturas bajo el esquema de cliente - servidor. Estas arquitecturas se definen en base a los elementos del sistema que las conforman y éstas son:

- *Arquitectura de 2 Capas:* Ésta es la arquitectura común del esquema cliente - servidor, en donde se puede identificar tres elementos: la interfaz del usuario (cliente), la gestión del procesamiento (en donde se especifican las reglas del procesamiento de la información) y la gestión de la base de datos (en donde se realizan tareas de validación y se realiza el acceso a la base de datos). En la primera capa se encuentra el cliente y la gestión del procesamiento, y en la segunda capa se encuentra la gestión de la base de datos.

  En este tipo de arquitectura el cliente puede tener dos comportamientos: en el primer caso el cliente es el que intenta incluir la lógica del programa, la forma del procesamiento y el tipo de presentación de la información: mientras que en el segundo caso el cliente solo se encarga de la lógica de la presentación y el manejo de datos, dejando la lógica de la aplicación al servidor. Estos clientes se conocen como *thick clients* y *thin clients*, respectivamente. La figura 1.35 muestra un ejemplo de la arquitectura de 2 capas.

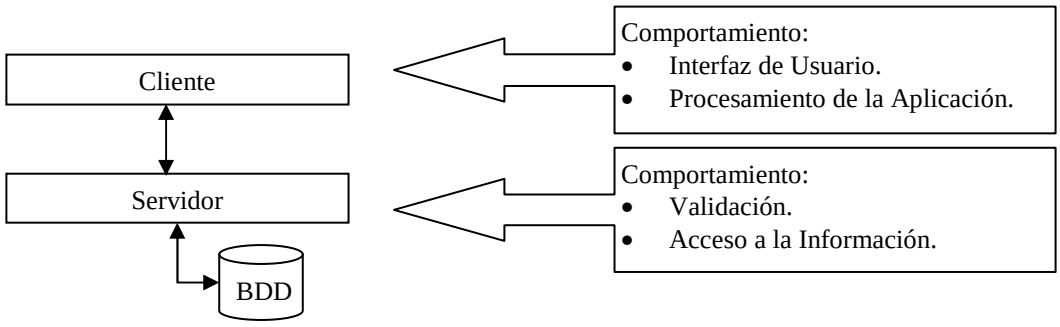


Figura 1. 35: Ejemplificación de la Arquitectura de 2 Capas

- *Arquitectura de 3 Capas:* En esta arquitectura se identifican tres componentes al igual que en la arquitectura de dos capas: la interfaz del usuario, el procesamiento de la lógica de la aplicación y la gestión de la información. Estos tres componentes se distribuyen en tres capas, la primera de ellas tiene al usuario en sí y se encarga de manejar la lógica de la presentación; la segunda capa incluye un servidor intermedio que maneja la lógica de la aplicación, y la tercera capa maneja el procesamiento de la información como se puede apreciar en la figura 1.36.

  Mediante el uso del servidor intermedio se consigue tener un *middleware*[6] capaz de permitir múltiples conexiones con clientes; ofrece flexibilidad, escalabilidad y seguridad al sistema, incrementa el rendimiento, y esconde la complejidad del procesamiento al usuario. Adicionalmente, el uso del *middleware* permite aligerar la carga al servidor de base de datos y sirve como un punto de protección de la información.

  Una implementación de la arquitectura de tres capas implica una complejidad mayor puesto que se tienen que controlar un mayor número de situaciones y elementos, pero brinda un mayor número de beneficios que la arquitectura de dos capas.

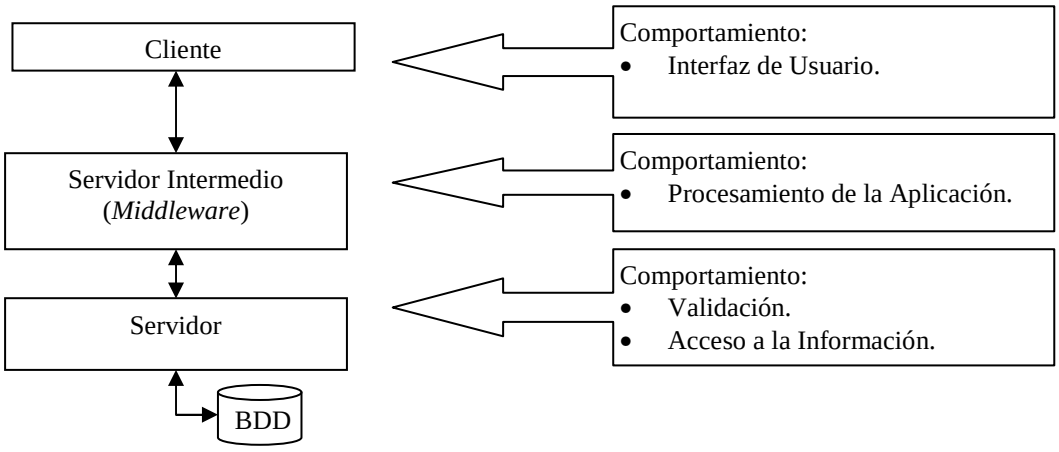


Figura 1. 36: Ejemplificación de la Arquitectura de 3 Capas

[6] *Middleware*: Módulo (software generalmente) intermedio que ofrece un punto de acceso transparente para la comunicación de dos procesos, adaptando la información y brindando seguridad.

1.3.4.2 MySQL [22]

MySQL es un gestor de base de datos diseñado en base a los principios de software libre; ésta es una de las principales características por las cuales ha tenido gran aceptación como uno de los motores de base de datos más usados en Internet. Además este gestor de base de datos es simple de usar y tiene un tiempo de respuesta rápido frente a una consulta. A continuación se describirán las principales características de MySQL:

- *Gestión de Base de Datos:* Es una aplicación que permite manejar un conjunto de datos eficientemente y de manera transparente. MySQL es un intérprete de SQL.
- *Base de Datos Relacional:* MySQL basa su funcionamiento en el modelo relacional, mientras que para la gestión de base de datos usa SQL estándar.
- *Open Source:* el código fuente se puede descargar de manera libre y la licencia que rige el uso de MySQL es GPL para aplicaciones no comerciales.
- *Sistema Multiplataforma:* MySQL puede ser instalado en cualquier distribución de UNIX, LINUX y adicionalmente existen versiones para sistemas Microsoft Windows ®.
- *Orientada a entornos de Cliente - Servidor.*
- *Rapidez en procesamiento de Consultas:* Debido a que es una base de datos de código abierto su velocidad ha sido optimizada por los usuarios.
- *Amplia compatibilidad con Aplicaciones:* MySQL es compatible con la mayoría de aplicaciones basadas en *open source*[7].
- *Manejo de Multithreading*[8]*:* Cuando MySQL se implementa sobre un servidor basado en Unix se puede implementar *multithreading* en el servidor aumentando de esta manera su desempeño.

[7] *Open Source*: Software de código liberado.
[8] *Multithreading*: Manejo de múltiples procesos simultáneamente.

El servidor de base de datos de MySQL permite realizar todas las operaciones de administración de bases de datos que incluye creación de base de datos y tablas; inserción, modificación, eliminación y consulta de datos; así como todas las operaciones que permite realizar SQL.

Las sentencias empleadas en MySQL se las ingresa mediante la línea de comandos o a su vez embebidas en un lenguaje de programación que se comunique con el servidor. Normalmente se emplea a PHP cuando se embeben las sentencias de instrucción en el ámbito de Internet; o se emplea JAVA en ámbitos de una intranet o en internet.

1.3.4.3 Oracle Database [23] [24]

Oracle Database es un gestor de base de datos creado por *Oracle Company* ® que ofrece una gran gama de opciones para potencializar las diferentes áreas que maneja este gestor. Originalmente Oracle diseñó su gestor para sistemas basados en Solaris (SPARC), pero en la actualidad existen diferentes versiones que se acoplan a las diferentes plataformas que hay en el mercado. Los beneficios ofrecidos por Oracle están orientados a *grid computing*[9] con una administración automatizada y automática. A continuación se describen las principales características de *Oracle Database*:

- *Gestión de Base de Datos.*
- *Base de Datos Relacional.*
- *Sistema Multiplataforma: Oracle Database* puede ser implementada sobre sistemas Microsoft ®, Linux, AIX5L, HP-UX (PA-RISC e Itanium), HP Tru64 UNIX, HP OpenVMS Alpha, OpenVMS Itanium, Solaris, Solaris (SPARC), OS/390 y Fujitsu Siemens BS2000/OSD.
- *Orientada a entornos de Cliente – Servidor.*
- *Administración de grandes volúmenes de datos con compresión y particionamiento:* Oracle permite reducir los costos de almacenamiento y

[9] *Grid Computing* término en inglés que hace referencia al incremento de la capacidad computacional bajo demanda mediante la compartición de recursos a través de un *middleware* sin la necesidad de aumentar equipos.

requisitos de espacio en disco mediante *Oracle Advanced Compression;* y aumenta el desempeño mediante el particionamiento de tablas e índices.

- *Seguridad en la Base de Datos:* Oracle implementa el módulo de *Oracle Database Security* para garantizar la privacidad y protección de los datos.
- *Diferentes Versiones: Oracle Company* ha desarrollado diferentes versiones de base de datos de acuerdo a los requerimientos de los diferentes sectores empresariales.

Oracle Database es un intérprete de SQL en donde las consultas y/o sentencias permiten realizar las operaciones de administración como: creación de base de datos y tablas; inserción, modificación, eliminación y consulta de datos; así como todas las operaciones que permite realizar SQL.

Oracle Company provee una serie de complementos modulares[10] que permiten ampliar la potencia de *Oracle Database* para ajustarse a los requerimientos específicos de una organización. Este tipo de características aumentan el rendimiento, disponibilidad, seguridad, *data warehousing*, y facilidad de uso; convirtiendo a *Oracle Database* en una base de datos bastante robusta.

1.3.4.4 Microsoft SQL Server [25]

SQL Server es un gestor de base de datos desarrollado por Microsoft ®. *SQL Server* puede ser implementado sobre sistemas personales así como servidores de multiproceso simétrico. Este gestor es un intérprete de comandos SQL que integra la administración de datos y el análisis de software de una manera confidencial considerando la criticidad de la información. Entre las principales características de este gestor se pueden considerar las siguientes:

- *Gestión de Base de Datos.*
- *Base de Datos Relacional.*
- *Orientada a entornos de Cliente – Servidor.*
- *Sistema Monoplataforma: SQL Server* solo puede ser instalado en distribuciones de Microsoft Windows.

[10] Tales como los módulos de: particionamiento, paquetes de administración, seguridad avanzada entre otros descritos en la página de Oracle.

- *Gama de Versiones:* Dependiendo de las características y necesidades de la organización se tienen diferentes versiones de *SQL Server* que pueden ser implementadas.
- *Herramientas Adicionales:* Además del gestor de base de datos, *SQL Server* implementa herramientas adicionales para generación de reportes, réplica de servidores, análisis de servicios, herramientas de clientes, SDKs, entre otras.
- *Gran Disponibilidad de la Información.*
- *Alto rendimiento.*
- *Sistema Escalable.*
- *Seguridad.*
- *Fácil Administración de la Información.*
- *Alta Productividad en el ciclo de desarrollo.*
- *Interoperabilidad:* El servidor de *SQL Server* mantiene una gran interoperabilidad con plataformas, aplicaciones y dispositivos a través del *Framework*[11] de .NET, los servicios Web y el soporte de estándares de la industria.

Gracias a estas características *SQL Server* se ha convertido en uno de los gestores de base de datos más difundidos a nivel global gozando de gran aceptación en el mercado [26] según *SD Times* y *BZ Research*.

SQL Server permite el acceso a los usuarios a través de una aplicación (Aplicación Cliente) basada en una API que contiene dos partes; la primera de ellas se encarga de pasar las instrucciones del lenguaje empleado (el lenguaje que debe estar basado en *Transact* – SQL) a la base de datos, mientras que la segunda se encarga de enviar instrucciones de un conjunto de funciones y métodos orientados a objetos a la base de datos y procesar los resultados devueltos por la base. Las principales API (API's nativas que no necesitan una

[11] *Framework:* Es una arquitectura de software que contiene las definiciones sobre las cuales se implementará otro software.

conversión y que se asocian directamente con el protocolo de red que se envía al servidor) manejadas por *SQL Server* son:

- *OLE DB:* Para aplicaciones que utilizan OLE DB como proveedor de servicios entre ellas *ActiveX Data Objects* (ADO).
- *ODBC:* Para aplicaciones o componente que necesitan un controlador ODBC nativo como DAO, RDO y clases de bases de datos de *Microsoft Foundation Classes* (MFC).

Adicionalmente, *SQL Server* también acepta *Bibliotecas de Bases de Datos* que es una API específica. Estas API pueden estar escritas en SQL, C o basadas en un componente llamado VBSQL (Microsoft Visual Basic). *SQL Server* también incluye un precompilador de C para una API *SQL incrustado*. Todas estas API's son implementadas como DLL[12] y se comunican con *SQL Server* a través de un componente llamado *biblioteca de red* tanto para el *cliente* como para el *servidor.*

SQL Server en su estructura funcional además de las bibliotecas de red del servidor maneja varios componentes de servidor tratados como un servicio, éstos son:

- Servicios abiertos de datos.
- Servicio MSSQLServer (*SQL Server*).
- Servicio SQLServerAgent (Agente *SQL Server*).
- Servicio Microsoft Search.
- Servicio MS DTC (Coordinador de transacciones distribuidas de Microsoft).

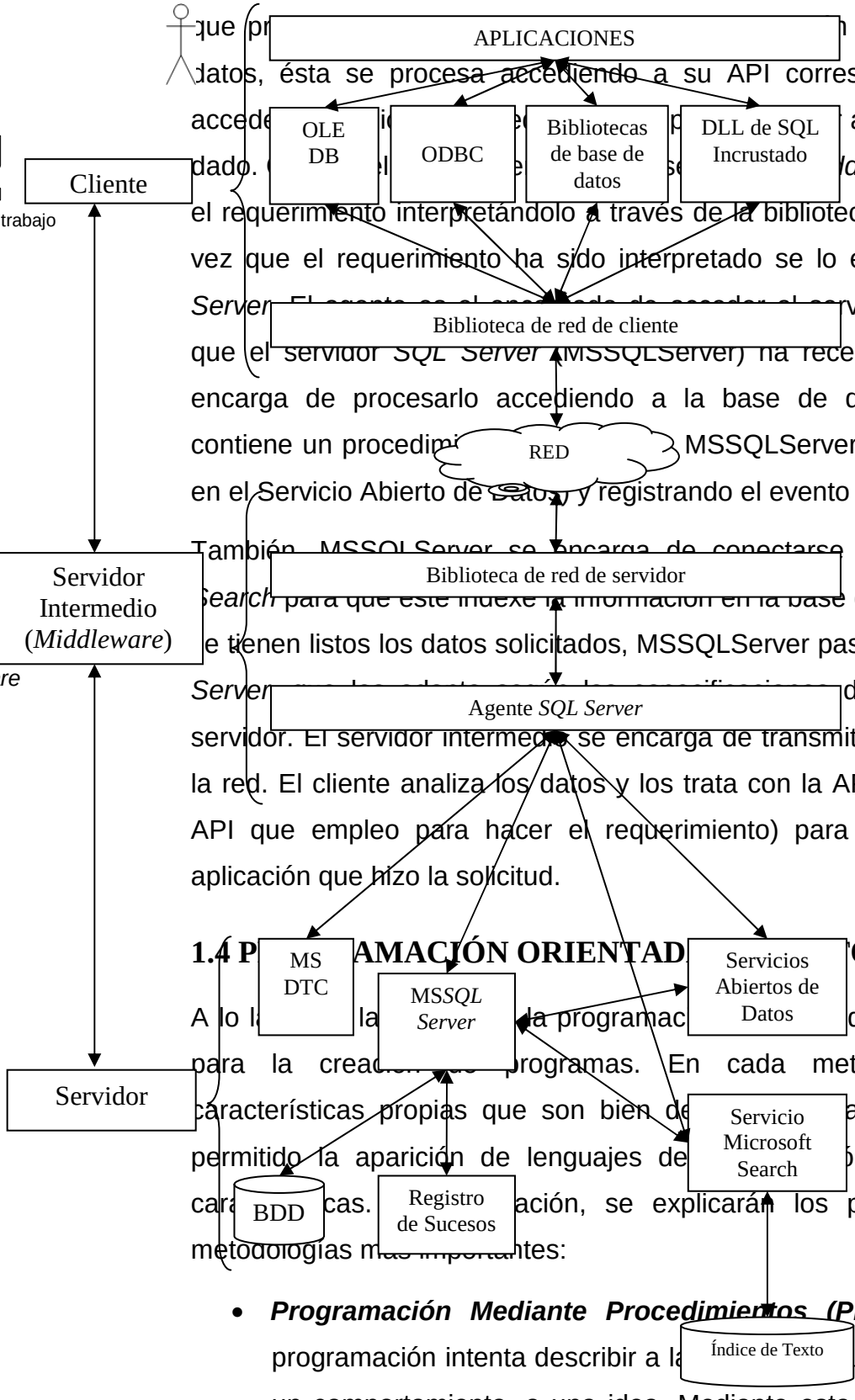
El componente de servicios abiertos de datos sirve como una interfaz entre bibliotecas de red del servidor y aplicaciones del servidor, permitiendo desarrollar procedimientos almacenados extendidos, y aplicaciones de servidor que acepten y procesen instrucciones Transact-SQL.

En la Figura 1.37 se esquematiza el funcionamiento de *SQL Server*, en la parte del cliente se indican las API's existentes y en el lado del servidor los servicios

[12] DLL (*Dinamic Link Library*): Archivos de código ejecutable que se cargan bajo demanda de un aplicativo. Es una librería compartida usada en los Sistemas Operativos que contiene definiciones de módulos, clases y espacios de nombres.

que pr… hace un requerimiento de datos, ésta se procesa accediendo a su API correspondiente, que a su vez accede… al servidor el requerimiento dado. … *Middleware* de servidor recepta el requerimiento interpretándolo a través de la biblioteca de red de servidor. Una vez que el requerimiento ha sido interpretado se lo ejecuta en el Agente *SQL Server*. El agente es el encargado de acceder al servidor *SQL Server*; una vez que el servidor *SQL Server* (MSSQLServer) ha receptado el requerimiento se encarga de procesarlo accediendo a la base de datos (si el requerimiento contiene un procedimiento… MSSQLServer consulta el procedimiento en el Servicio Abierto de Datos) y registrando el evento en el registro de sucesos.

También, MSSQLServer se encarga de conectarse con el Servicio *Microsoft Search* para que este indexe la información en la base de índice de texto. Cuando se tienen listos los datos solicitados, MSSQLServer pasa los datos al Agente *SQL Server* … de la biblioteca de red de servidor. El servidor intermedio se encarga de transmitir la respuesta a través de la red. El cliente analiza los datos y los trata con la API correspondiente (misma API que empleo para hacer el requerimiento) para luego entregárselos a la aplicación que hizo la solicitud.



1.4 P… AMACIÓN ORIENTAD… OS

A lo l… la… la programac… do diferentes metodologías para la creac… programas. En cada metodología se especifican características propias que son bien de… a metodología también ha permitido la aparición de lenguajes de … ón que se adaptan a sus cara… cas. … ación, se explicarán los principales rasgos de las metodologías más importantes:

- ***Programación Mediante Procedimientos (Procedural):*** Este tipo de programación intenta describir a la… un fenómeno, una acción, un comportamiento, o una idea. Mediante esta metodología, el programa intenta simular un procedimiento existente en el mundo real para imitar un

comportamiento o una acción humana. Esta metodología fue la primera en ser descrita y fue empleada en los años cuarenta y cincuenta, en donde los mecanismos de abstracción eran el lenguaje de máquina y el lenguaje de ensamblador (el lenguaje ensamblador ofrecía la posibilidad de usar mnemónicos que eran abstracciones de secuencias de bits que componían las instrucciones de un programa facilitando la programación). Posterior a este nivel apareció un siguiente nivel de abstracción en donde se agruparon instrucciones primitivas (instrucción de máquina que dependía de la arquitectura de la máquina en donde se compilaba la aplicación) para formar macroinstrucciones, de esta manera se podía realizar muchas acciones con una sola llamada. A finales de los años cincuenta aparecieron los *lenguajes de programación de alto nivel* que incluían un nuevo nivel de abstracción permitiendo a los programadores despreocuparse de las características físicas de las computadoras donde se ejecutaba la aplicación creando programas de propósito genérico. En un lenguaje de alto nivel las secuencias de sentencias se pueden agrupar en *procedimientos* que son invocados por una sentencia. Esta metodología basa la estructura de la programación en un algoritmo que describe el funcionamiento de un programa basado en procedimientos.

Una de las principales desventajas de los programas basados en esta metodología era el costo del software, tanto en relación al desarrollo inicial así como al mantenimiento del mismo. Otra inconveniente de esta metodología era que no podía manejar GUI's (interfaces gráficas).

- ***Programación Estructurada:*** Esta metodología apareció en los años sesenta, y su implementación mejoraba las aplicaciones basadas en procedimientos al emplear una programación estructurada; con esto se obtenían programas más claros, fáciles de probar y corregir, y más fáciles de modificar. Los nuevos conceptos que se incluyen en esta metodología son: *estructuras de control* que permiten marcar estados iníciales y finales, también indican puntos de entrada y salida de cada estructura; mediante las estructuras de control se pueden controlar los estados de acción y decisión. Otro concepto que implementa esta metodología es la *función,*

que promueve la reutilización del software y que permiten realizar tareas que se ejecutan repetidamente evitando así duplicar el código varias veces. Adicionalmente, las funciones permitieron el ocultamiento de la información ya que para la utilización de una función (o conjunto de funciones) solo es necesario conocer la interfaz a través de la cual se accede a su funcionalidad y no su implementación. Esta metodología también introduce el concepto de *módulos* que hace referencia a la división de un sistema complejo en sus componentes para reducir el tiempo de pruebas y las probabilidades de errores. Una aplicación es la combinación de sus componentes manejando de mejor manera la complejidad del sistema. La programación estructurada también permitió el manejo de interfaces gráficas con menús y ventanas.

Las principales desventajas de la programación estructurada son las dificultades que presenta su diseño ya que a través de sus componentes (funciones, estructuras de datos y control, y módulos) no se puede modelar bien el mundo real contemplando toda la gama de posibilidades existentes.

- ***Programación Orientada a Objetos:*** Esta metodología es la que se emplea desde mediados de los ochenta, se popularizó en los noventa, hasta la actualidad para el desarrollo de software. Mediante esta metodología se reduce la complejidad inherente al software (complejidad total del software y su comportamiento versus la sencillez de uso para el usuario) y se puede controlar de mejor manera los factores de calidad del software (algunos de estos factores se basan en los principios de la ingeniería de software). Existen numerosos factores de calidad, de los cuales se destacan:
    - *Fiabilidad:* El software debe funcionar correctamente y sin fallas.
    - *Eficiencia:* El software debe hacer un buen uso de los recursos computacionales (CPU, memoria, red, archivos, etc.) que tiene a disposición.

- *Portabilidad:* Este factor hace referencia a la facilidad con la que la aplicación puede ser usada sobre diferentes sistemas físicos o lógicos.
- *Comprensibilidad:* Refleja el hecho que el software sea fácil de comprender para su mantenimiento o modificación.
- *Verificabilidad:* Es la facilidad de verificación de un software para soportar procedimientos de validación y pruebas.
- *Integridad:* Es la capacidad de un programa para proteger sus componentes contra los procesos que no tenga derecho de acceso.
- *Facilidad de Uso:* Esta característica hace referencia a la facilidad que tiene un usuario para interactuar con la aplicación.
- *Robustez:* Este parámetro expresa la capacidad del producto para funcionar incluso en situaciones anormales.
- *Adaptabilidad:* Debido a la larga vida del software éste debe permitir cambios y añadiduras según las nuevas necesidades del mercado para mejorar su efectividad y competitividad. El diseño debe contemplar la posibilidad de aumentar nuevas características y capacidades en un futuro.
- *Reutilizabilidad*: Para minimizar los costos del software y el tiempo de desarrollo se deben diseñar sus componentes para que ofrezcan flexibilidad en su comportamiento, consiguiendo así la posibilidad para que éstos puedan ser reutilizados cuando se desarrollen nuevos componentes o nuevas aplicaciones.
- *Rentabilidad:* El costo de desarrollo y mantenimiento del software no debe exceder el beneficio esperado por la comercialización del mismo. Si se reduce el tiempo de desarrollo se reducen los costos de producción pero los costos más importante giran en torno al mantenimiento; por lo tanto un buen diseño, una buena implementación y el cumplimiento de los factores previamente descritos reducirán estos costos.

La programación orientada a objetos (POO) puede ser considerada como una metodología que revolucionó la forma de producir software ya que representa una nueva forma de pensar y hacer diseños en base a una abstracción de objetos del mundo real. La POO emplea objetos y no algoritmos como bloques de construcción lógicos en donde cada objeto representa una instancia de una clase, cada clase es una representación abstracta de un ítem (y de su comportamiento) del mundo real y las clases se relacionan entre sí por medio de relaciones de herencia.

Las razones fundamentales que han hecho que la POO sea la metodología principal para el desarrollo de una aplicación giran en torno a:

- Es una tecnología adecuada para la realización de aplicaciones.
- Los métodos de encapsulamiento (ocultación de la implementación) soporta un alto grado de reutilización de código que se incrementa por sus mecanismos de herencia y de instancias.
- En el entorno de las bases de datos, se adjunta bien a los modelos semánticos de datos para solucionar las limitaciones de modelos tradicionales incluyendo el modelo relacional.
- Existe una gran gama de lenguajes basados en la programación orientada a objetos.
- Permite desarrollar interfaces gráficas para los usuarios que facilitan su interacción sin eliminar la posibilidad de interacción por medio de la línea de comandos.
- Se ha logrado aumentar la capacidad para definir nuevos tipos de datos (clases).

Todas estas razones han hecho que la programación orientada a objetos provea el marco conceptual para la generación de nuevos programas.

1.4.1 CONCEPTOS PRINCIPALES

A continuación se presenta un glosario de términos y conceptos que son la base teórica para la programación orientada a objetos.

- *Abstracción:* Mecanismo de la mente humana fundamental para la comprensión de fenómenos o situaciones que involucren una gran cantidad de detalles.
- *Abstracción y Encapsulamiento:* La abstracción analiza el comportamiento observable de un objeto, mientras que el encapsulamiento analiza la implementación que da lugar a ese comportamiento.
- *Modelar:* Consiste en la identificación (abstracción) de objetos del universo del problema. Se identifica cómo son los objetos, su comportamiento y sus relaciones.
- *Objeto:* Es una abstracción de un elemento del contexto del problema, sirve para modelar entidades del mundo real. Un objeto también es una instancia de una clase.
- *Clase:* Describe un conjunto de objetos del contexto del problema que tienen las mismas características y el mismo comportamiento. La clase permite realizar la definición de la estructura y el comportamiento de objetos de un tipo determinado.
- *Jerarquía de Clases:* Determina la estructura de varias clases relacionadas, y que especifica qué clases heredan funcionalidad de otras clases.
- *Atributos:* Característica asignada a un elemento de la clase. Sirven para determinar propiedades o métodos y definen el nivel de accesibilidad para el código en otras partes del programa.
- *Tipo de Datos:* Tipo de información que puede ser almacenado en una variable (o constante) específica. En el contexto general de la POO, tipo de datos significa lo mismo que clase.
- *Constructor:* Método que se puede definir en una clase que sirve para crear una instancia de la misma. En un constructor se puede especificar valores predeterminados o valores pasados como argumentos. Normalmente no se puede utilizar un objeto sin antes haberlo instanciado (inicializado) a través de la llamada al constructor.
- *Instancia:* Objeto creado a lo largo de la ejecución de un programa.

- *Espacio de Nombres:* Atributo personalizado que ofrece un control más preciso sobre el código al que puede acceder otro código. En un espacio de nombres se puede definir clases que son relacionadas entre sí.
- *Sobrecarga:* Mecanismo de la POO que permite definir métodos o funciones con el mismo nombre pero con diferente funcionalidad diferenciándose entre sí por los argumentos que tienen. La sobrecarga se da en el mismo ámbito de la clase.
- *Agregación:* Técnica que permite abstraer relaciones entre clases en donde una clase tiene como atributos (está compuesta por) a otras clases.
- *Asociaciones:* Técnica que permite abstraer relaciones entre objetos que no pueden ser considerados como agregación.
- *Herencia:* Mecanismo de la POO que permite a una definición de clase incluir toda la funcionalidad de una definición de clase distinta y añadir nueva funcionalidad.
- *Polimorfismo:* Es la posibilidad de definir de forma distinta un método o función dependiendo del objeto. El polimorfismo se da cuando hay relaciones de herencia y redefine un método o función con una funcionalidad diferente para el objeto heredero.

Otro aspecto que se tiene que considerar cuando se implementa aplicaciones basadas en una programación orientada a objetos son las tareas comunes que se tienen que realizar, éstas son:

1. Definición de clases y nombres de espacio.
2. Creación de propiedades, métodos, funciones y descriptores de acceso (lectura y escritura según sea el caso).
3. Control de acceso a las clases, propiedades, métodos, funciones y descriptores de acceso dependiendo el ámbito en el cual serán empleadas. Normalmente se utilizan tres ámbitos: público, privado, compartido.
4. Creación de propiedades, métodos y funciones estáticas.
5. Creación de estructuras de enumeración.

6. Definición y utilización de interfaces.
7. Utilización de la herencia y la sustitución de elementos de clase.

1.4.2 UML

Unified Modeling Language (UML) es una metodología para el desarrollo de software, que permite a través de una notación crear un conjunto de diagramas normalizados que sirven para describir el comportamiento del sistema y los detalles de la arquitectura. Esta notación es independiente de los lenguajes de programación y está orientada al desarrollo de sistemas basados en la programación orientada a objetos. El resultado de emplear esta metodología es un modelo (que viene determinado por un conjunto de diagramas) que sirve como un plano para el desarrollo de la aplicación.

En el proceso de análisis y diseño basado en UML intervienen diferentes grupos de personas (usuarios, desarrolladores, y administradores) para generar un conjunto de solicitudes que permitirán definir el problema. Con el problema definido se procede a extraer los requisitos a partir de entrevistas con usuarios y con experiencias del mundo real; con los requisitos se construyen modelos que serán empleados en la implementación de la aplicación.

En el modelo del desarrollo orientado a objetos se pueden apreciar diferentes dimensiones del mismo; la primera de ellas es el *modelo estático* que define la estructura estática del sistema y que a su vez define el *modelo lógico* que permite apreciar la arquitectura de la aplicación desde el punto de vista de las abstracciones y emplea como herramientas los *diagramas de clases y diagramas de objetos*. El otro modelo contemplado en el modelo estático es el *modelo físico* que define la arquitectura desde el punto de vista del hardware y software; este modelo emplea como herramientas de diseño el *diagrama de módulos y el diagrama de procesos*. La segunda dimensión que se contempla en el desarrollo orientado a objetos es el *modelo dinámico* que describe la evolución dinámica y las interacciones entre objetos empleando como herramientas los *diagramas de transición de estados y los diagramas de interacción o de seguimiento de*

sucesos. La figura 1.38 esquematiza las partes del modelo del desarrollo orientado a objetos.

UML aunque define un conjunto de diagramas que son básicos para crear una aplicación, no define en sí la metodología ni los pasos que se deben seguir para poder generar dichos diagramas. En la actualidad existen diferentes metodologías para el desarrollo de sistemas computacionales que usan UML como lenguaje de modelado, aunque cada una de estas metodologías se las aplica en diferentes escenarios y bajo ciertas circunstancias se plantean los siguientes pasos como generales en la generación de un modelo con UML:

1. Descripción del Problema.
2. Recolección de Requerimientos.
3. Descripción de Escenarios y Selección de Actores.
4. Determinación de Casos de Uso.
5. Identificación de Objetos.
6. Diagramas de Secuencia (Diagramas de Interacción).
7. Diagramas de Colaboración.
8. Diagramas de Estados.
9. Diagramas de Actividad.
10. Diagramas Estáticos (Diagramas de Clases).
11. Diagramas de Implementación.

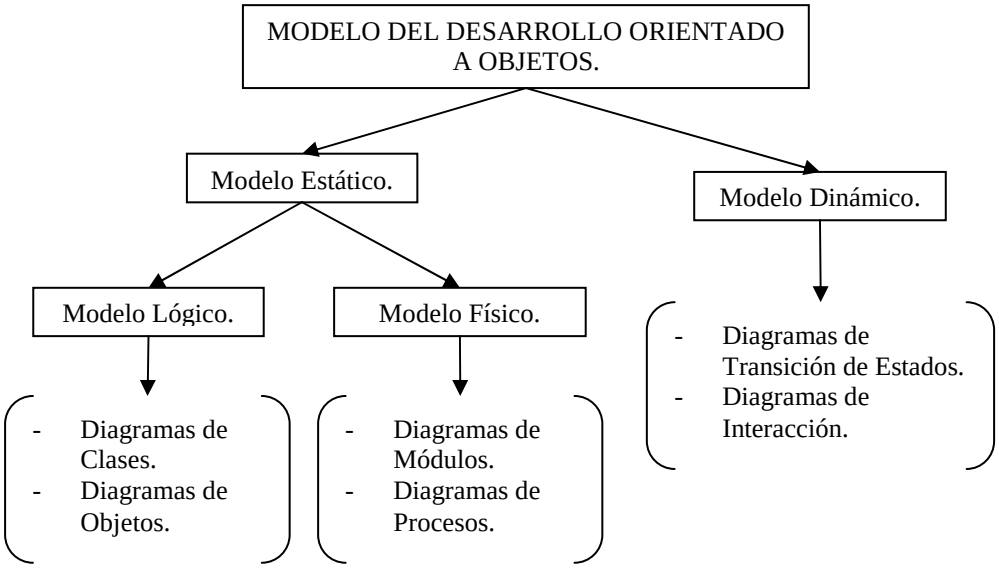


Figura 1. 38: Modelo del desarrollo orientado a objetos

Al finalizar estos pasos, se obtendrá un conjunto de diagramas que permitirán tener una noción completa de la aplicación a implementarse; es importante aclarar que en el proceso de diseño de una aplicación estos pasos son iterativos. A continuación se explicará más detalladamente cada uno de estos pasos.

1.4.2.1 Descripción del Problema

Para realizar la descripción del problema se deben reunir a todos los involucrados formando un equipo multidisciplinario (consiguiendo así la mayor parte de ópticas posibles) en donde se discute y plantea el problema que se intenta resolver con el software. En esta etapa se deben especificar todos los detalles del problema, además se debe adjuntar si existe documentación que ayude a esta descripción. Como resultado del análisis del problema se determina el título de la aplicación que debe describir brevemente la misma y el documento de análisis que contiene la descripción del problema escrito en narrativa.

1.4.2.2 Recolección de Requerimientos

A partir de la descripción del problema se empiezan a determinar todos los requerimientos para la aplicación, es decir se buscan las necesidades y funcionalidades con las que la aplicación debe cumplir. Este paso produce un documento que contiene los requerimientos debidamente enumerados.

1.4.2.3 Descripción de Escenarios y Selección de Actores

El siguiente paso en el diseño de sistemas es la descripción de escenarios, que no es más que narrar las posibles situaciones con las que la aplicación pudiera interactuar. En la figura 1.39 se muestra la información que un escenario debe tener. Se inicia con la *numeración* que es un número que permite distinguir el escenario con el que se trabaja; el *título* es una descripción corta del escenario y normalmente se lo define con un verbo; los *prerrequisitos* hacen referencia a las situaciones y/o condiciones previas necesarias para que el escenario se dé. En los siguientes dos puntos de una ficha CRC[13] se describe las personas (u otro sistema) que inicia y termina el escenario. *Post condiciones* indica el estado en la que se queda la aplicación después de haberse dado el escenario. Finalmente la *descripción* permite narrar todo lo que sucede en esta situación. Mientras más detalles se tienen en la descripción de un escenario posteriormente se podrá hacer una mejor abstracción en la determinación de los demás diagramas.

Numeración:
Título:
Prerrequisitos:
Quien lo comienza:
Quien lo termina:
Pos condiciones:
Descripción:

Figura 1. 39: Información de un escenario

En este paso también se realiza la identificación de los *actores* que representan todos los roles de personas u otros sistemas que interactúan con la aplicación que se está diseñando. Los actores se los representa con una persona (sin importar que sean sistemas) y se los enlista. Es bueno que en este punto se defina un perfil para cada rol del actor.

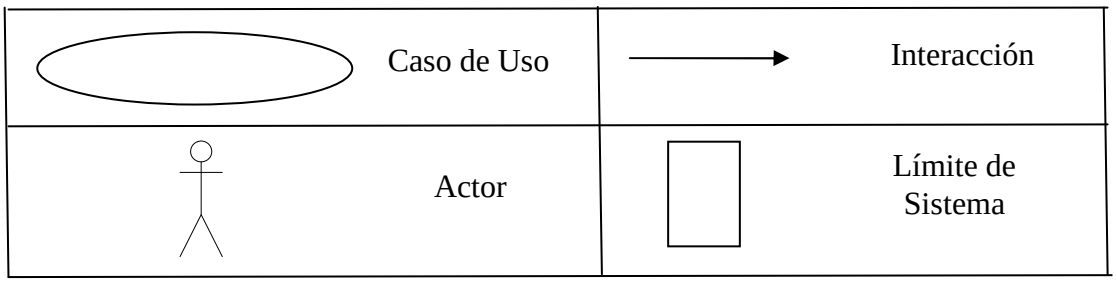
[13] Ficha CRC: Herramienta definida por UML que contienen diferente información y permite representar relaciones de colaboración o jerarquías de generalización/especialización mediante su ubicación espacial. Normalmente permite definir una clase, sus responsabilidades y sus colaboradores.

1.4.2.4 Determinación de Casos de Uso

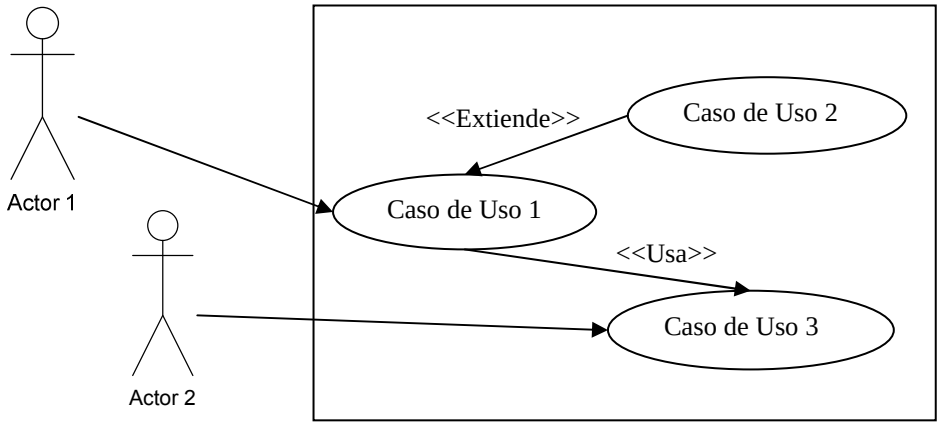
Los casos de uso se los realiza a partir de los escenarios y sirven como una interpretación gráfica de los mismos; con el caso de uso se puede representar la interacción entre un actor (o varios) y el sistema que se está diseñando.

Cada caso de uso tiene dos partes: la parte gráfica que modela un determinado escenario y una descripción. Se puede considerar que el diagrama de un caso de uso es la interpretación gráfica de un escenario y facilita su entendimiento. Se aconseja que cada escenario tenga su propio caso de uso y además se haga un diagrama que combine todos los escenarios. Existen dos relaciones que se pueden dar en un caso de uso; la primera se conoce como <<*extiende*>> y se emplea cuando el caso de uso es similar a otro ya descrito pero se le añade una característica (comportamiento) nueva; la segunda relación conocida como <<*usa*>> y es empleada cuando parte del comportamiento que se desea representar está en otro caso de uso y no se desea repetir dicho comportamiento en el caso de uso que se está representando.

En la figura 1.40 (a) se indican los elementos gráficos que se emplean para realizar un diagrama de caso de uso. Mientras que en la figura 1.40 (b) se ejemplifica el esquema de un caso de uso.



(a)



(b)

Figura 1. 40: Elementos y Ejemplificación de un Caso de Uso

1.4.2.5 Identificación de Objetos

En el siguiente paso se procede a realizar la primera abstracción de los objetos necesarios para el funcionamiento de la aplicación. Esta abstracción genera los objetos (definiendo atributos, propiedades, métodos y funciones) y las relaciones entre ellos (mensajes, herencia, agregación, etc.). Además de analizar los objetos y sus relaciones también es importante analizar la organización de los mismos.

En este punto se especifican las primeras características de las futuras clases (con los objetos) que sirven como sustento para la realización de los siguientes diagramas. Estas especificaciones se las puede realizar de manera narrativa y mediante una ficha CRC. En la figura 1.41 se esquematiza una ficha CRC.

Clase: **Superclases:** **Subclases:**	
Atributos	
Responsabilidades	Colaboradores

Figura 1. 41: Esquema de una ficha CRC

Las fichas CRC posteriormente se modificarán según los análisis de otros diagramas y darán lugar a los diagramas estáticos. Otra ventaja del uso de estas fichas es que enriquecen el diccionario de datos escrito al final del proceso.

1.4.2.6 Diagramas de Secuencia (Diagramas de Interacción)

Estos diagramas también son conocidos como diagramas de interacción, su función principal es analizar la comunicación entre objetos que se dan en cada uno de los escenarios. A través de estos diagramas se puede apreciar en detalle el comportamiento de un escenario y se puede considerar que el resultado de su aplicación es obtener los métodos de los objetos que intervienen en una clase. Los diagramas de secuencia se enfocan en el orden de los mensajes y si se requiere un mayor detalle del comportamiento de un escenario se procede a realizar o revisar (según sea la etapa en la que se encuentra) el diagrama de colaboración.

Para generar un diagrama de secuencia primero se deben enumerar los objetos pertenecientes al escenario de manera horizontal, debajo de cada objeto se traza una línea vertical que sirve como referencia para ver el momento en el que el objeto debe ser creado. A medida que se continúa con el análisis del escenario sobre la línea vertical, se dibuja un rectángulo que representa el tiempo que cada objeto está activo y existirá. Las relaciones que puede haber entre objetos se

generan por eventos o por operaciones y se las representa por medio de una flecha; cada una de estas relaciones (mensajes) mantienen un orden que se indica mediante su posición vertical en el diagrama. En la parte izquierda del diagrama se explica las instrucciones generales de cada mensaje del diagrama basadas en la descripción del escenario y del caso de uso, esta explicación se denomina *guión.* En la figura 1.42 se representa un diagrama de secuencia genérico.

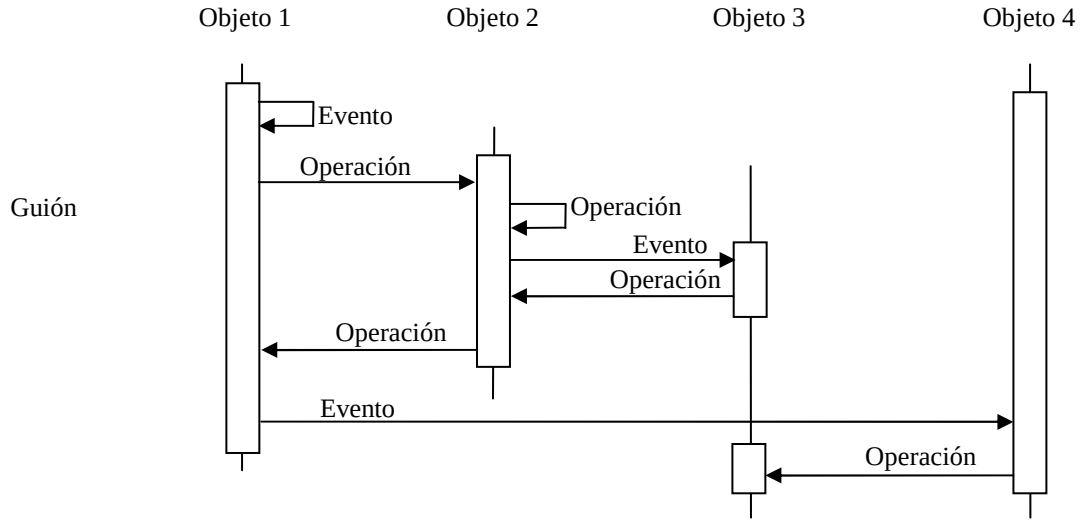


Figura 1. 42: Diagrama de Secuencia Genérico

1.4.2.7 Diagramas de Colaboración

La función principal de estos diagramas es analizar la distribución espacial de las interacciones de los objetos, es decir analizar el orden en los que se producen los mensajes (interacciones como eventos y operaciones) entre los objetos en tiempo de ejecución. Con el uso de los diagramas de colaboración se puede profundizar el nivel de detalle de un diagrama de secuencia.

En este tipo de diagramas se identifican los objetos mediante un rectángulo, cada objeto puede tener dos tipos de comportamientos: cuando invocan una operación se los conoce como *objetos clientes* y cuando suministran la operación se los conoce como *objetos servidores.* La dirección de la invocación viene determinada por una flecha junto con el número del mensaje que indica el orden del mismo y el

nombre de la operación a realizarse. Adicionalmente se pueden indicar los papeles o roles, las claves y restricciones presentes en los objetos.

Otra característica de los diagramas de colaboración es la facilidad que prestan para explicar un escenario mediante la dirección del flujo de datos a través de una flecha y un círculo, y para indicar el flujo se puede usar un objeto o un valor.

Existen escenarios en los que además de indicarse los valores anteriores también es importante señalar la visibilidad que un objeto tiene sobre otro, para esto se usan las marcas de visibilidad indicadas en la tabla 1.5.

Marca de Visibilidad	**Significado**
G	Indica que el objeto servidor es global al cliente.
P	Indica que el objeto servidor es parámetro de una operación del cliente.
F	Señala que el objeto servidor es parte del cliente.
L	Señala que el objeto servidor es un objeto declarado como local en el ámbito del cliente.

Tabla 1. 5: Marcas de Visibilidad usadas en el Diagrama de Colaboración

Si un objeto maneja su propio hilo de control (ejecución) se conoce como *objeto activo;* caso contrario los objetos son *secuenciales* (característica por defecto), se manejan con un único hilo de control y a esta característica se la conoce como *sincronismo simple.*

Finalmente, en los diagramas de colaboración existen cuatro niveles de concurrencia y éstos son:

- *Síncrono:* El cliente espera hasta que el servidor acepte el mensaje.
- *Paso de Mensajes con Abandono Inmediato:* El cliente abandona el mensaje si el servidor no lo atiende inmediatamente.
- *Sincronización de Intervalo (timeout):* El cliente abandona el mensaje si el servidor no lo atiende después de un lapso de tiempo determinado (*timeout*).

- *Asíncrono:* El cliente envía el evento al servidor para que lo procese, el servidor pone el mensaje en una cola de espera y el cliente continua su actividad hasta que el servidor tenga un resultado.

En la figura 1.43 se especifica la nomenclatura empleada para describir el sincronismo y la concurrencia en los diagramas de colaboración.

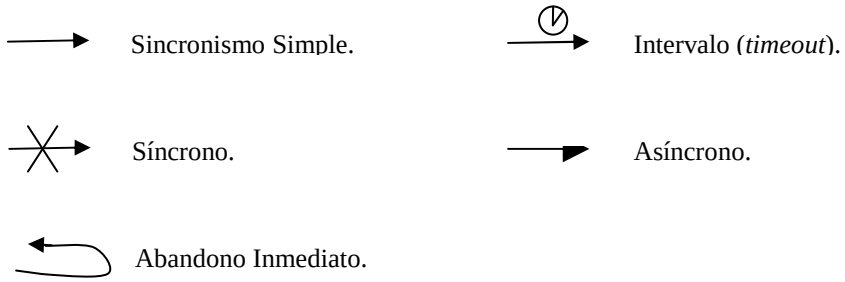


Figura 1. 43: Nomenclatura empleada en Sincronismo y Concurrencia en los diagramas de colaboración

1.4.2.8 Diagramas de Estados

Los diagramas de estados, también conocidos como diagramas de transición de estados, sirven para representar y analizar la secuencia de estados, las respuestas y acciones de un objeto durante su ciclo de existencia frente a los estímulos que recibe; estos estímulos son conocidos como *eventos* y causan cambios en el estado del objeto y del sistema.

En un diagrama de estados normalmente se parte de un estado por defecto representado por un círculo con una flecha. Éste es el punto de partida para la máquina de estados; al estado en si se lo representa con un rectángulo con las esquinas curvas y a los eventos se los representa con una flecha. Se puede *asociar acciones* a los estados indicando qué ocurre cuando se entra o sale del estado; del mismo modo se puede emplear *transiciones de estado condicionales* mediante el uso de expresiones booleanas entre llaves. Los Diagramas de Estados también permiten el *anidamiento de estados*, en donde el estado continente se llama *superestado* y los estados anidados se llaman *subestados*. En la figura 1.44 se puede encontrar la nomenclatura empleada en estos diagramas.

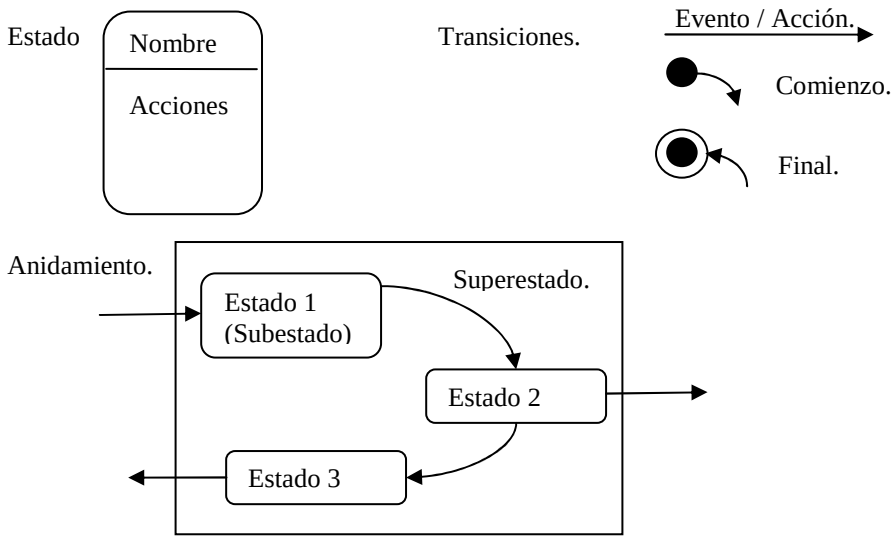


Figura 1. 44: Nomenclatura empleada en los Diagramas de Estados

1.4.2.9 Diagramas de Actividad

Los diagramas de actividad representan cómo se dirigen los flujos de los procesos internos en correspondencia a los flujos ocurridos en las clases, las operaciones de las clases y los casos de uso. Normalmente se emplean los diagramas de actividad en situaciones donde los eventos parten de acciones generadas internamente, en oposición a los diagramas de estados que son empleados cuando suceden eventos externos y en situaciones donde ocurren eventos asíncronos.

Estos diagramas emplean rectángulos redondeados para representar las actividades, para indicar que se debe tomar una decisión se emplea el rombo, y al igual que los diagramas de estados los diagramas de actividades parten de un punto inicial representado por un círculo y una flecha. Se puede organizar de mejor manera el diagrama mediante el uso de *Swinlane* (calles o pasillos) de actividades que no es más que una tabla de columnas en donde se dibujan las actividades. Existen dos íconos empleados para el control, uno de ellos sirve para enviar y recibir señales y el otro para denotar eventos aplazados. Adicionalmente, en los diagramas de actividades se puede representar relaciones de flujo entre acciones y objetos, en donde la acción se la representa con una flecha de línea entrecortada y los objetos mediante un rectángulo. En la figura 1.45 se especifica toda esta nomenclatura.

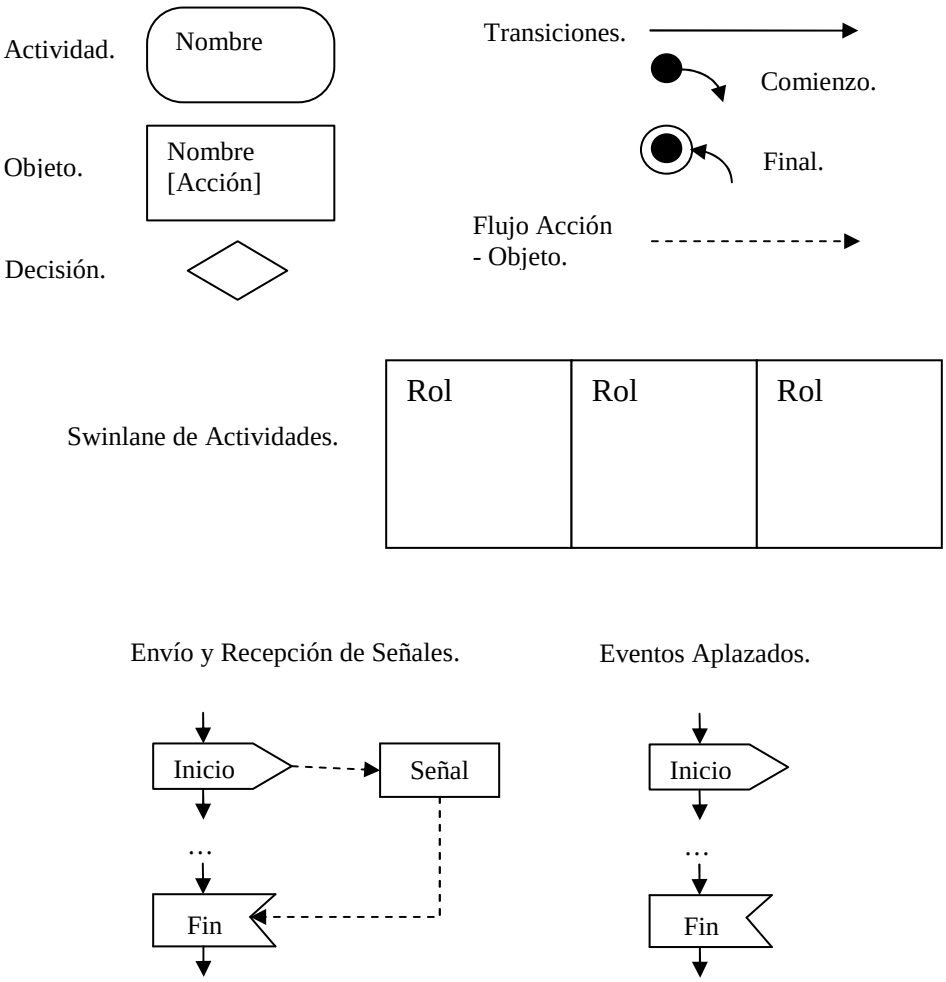


Figura 1. 45: Nomenclatura empleada en los Diagramas de Actividades

1.4.2.10 Diagramas Estáticos (Diagramas de Clases)

Los diagramas estáticos también conocidos como *Diagramas de Clases* son los más importantes del diseño orientado a objetos, puesto que éstos contienen toda la información de todas las clases y sus relaciones con otras clases de manera global, es decir sin particularizar en ningún escenario. Estos diagramas contienen un gran nivel de abstracción y son el producto de los diagramas anteriores, ya que a través de éstos se obtienen nombres de clases, se generan los atributos y operaciones de cada clase en base a los requerimientos planteados y analizados en los escenarios y casos de uso.

Se debe considerar que los clientes sólo operan sobre objetos y no sobre la clase; los atributos y las operaciones pueden ser visibles o no según el detalle deseado. Los atributos son datos que se almacenan en los objetos, los métodos y funciones

corresponden a la implementación de las operaciones y existen adornos de clase como las propiedades.

UML representa las clases de tres maneras:

- Sin Detalle
- Con Detalles a nivel de análisis y diseño.
- Con Detalles a nivel de implementación.

Aunque de éstas la forma más común de representar una clase es mediante el uso de detalles a nivel de análisis y diseño. Los atributos se pueden describir como:

Visibilidad: tipo = valor_inicial {propiedad}

En donde la visibilidad representa el grado de acceso de un cliente a la modificación de un atributo o método y se representa por los siguientes símbolos:

+ Para nivel de acceso público.

Para nivel de acceso protegido.

- Para nivel de acceso privado.

Las clases son representadas con un rectángulo en donde en la parte superior se escribe el nombre, el rectángulo se divide en compartimientos que contienen los atributos (descritos mediante el uso de la nomenclatura antes indicada) y las operaciones (definidas como métodos o funciones). Se pueden añadir compartimientos (en donde cada compartimiento puede tener su propio nombre) para ordenar los atributos y operaciones.

En el diseño basado en UML, los diagramas de clases pueden tener estereotipos que representan formas de clasificar las clases a alto nivel; un estereotipo puede ser definido como un ícono o mediante la siguiente nomenclatura:

<< Nombre estereotipo >>.

Cada clase se puede especializar usando estereotipos en *Tipos* y *Clases de Implementación.* Un *tipo* caracteriza un papel cambiante que un objeto puede adoptar y después abandonar, se representa mediante el uso del estereotipo: <<*type*>>. Una *clase de implementación* en cambio define físicamente la

estructura de datos y procedimientos que se implementará en un lenguaje de programación determinado; se representa mediante el uso del estereotipo <<*implementation class*>>. Se puede marcar una relación <<*realiza*>> que se indica con una flecha hueca con trazos discontinuos para indicar la implementación de un tipo por medio de una clase de implementación; es decir la relación implica una herencia de operaciones pero no de estructura.

Otro componente que se emplea en los diagramas de clases, son las *interfaces* que representa una especificación para las operaciones externas visibles de una clase, componente, u otra entidad pero sin especificar la estructura interna. Normalmente una interfaz solo especifica una parte limitada del comportamiento de una clase real, por lo tanto no tiene implementación y formalmente equivale a una clase abstracta que solo tiene operaciones abstractas (la clase no tiene atributos ni métodos). Existen dos formas de representar una interfaz:

- Mediante el uso del estereotipo <<*interface*>> en un rectángulo y la lista de las operaciones.
- Mediante el uso de un círculo unido por una línea a una clase con las operaciones de interfaz al lado del círculo.

En la sección 1.4.1 del presente capítulo se definieron las relaciones que pueden existir entre clases, estas relaciones en UML son representadas mediante la nomenclatura especificada en la figura 1.46.

La agregación y la composición son casos particulares de la asociación que denotan posesión (se maneja un todo y una parte); cuando se emplea la agregación se intenta dar una denotación de posesión basada en dirección o referencia indicando que sólo contiene un puntero, los tiempos de vida del todo y su parte son independientes. Por otro lado, cuando se emplea la composición y la denotación de posesión es por valor, indicando que se hace una copia del agregado y asegurando que los tiempos de vida del todo y su parte son iguales.

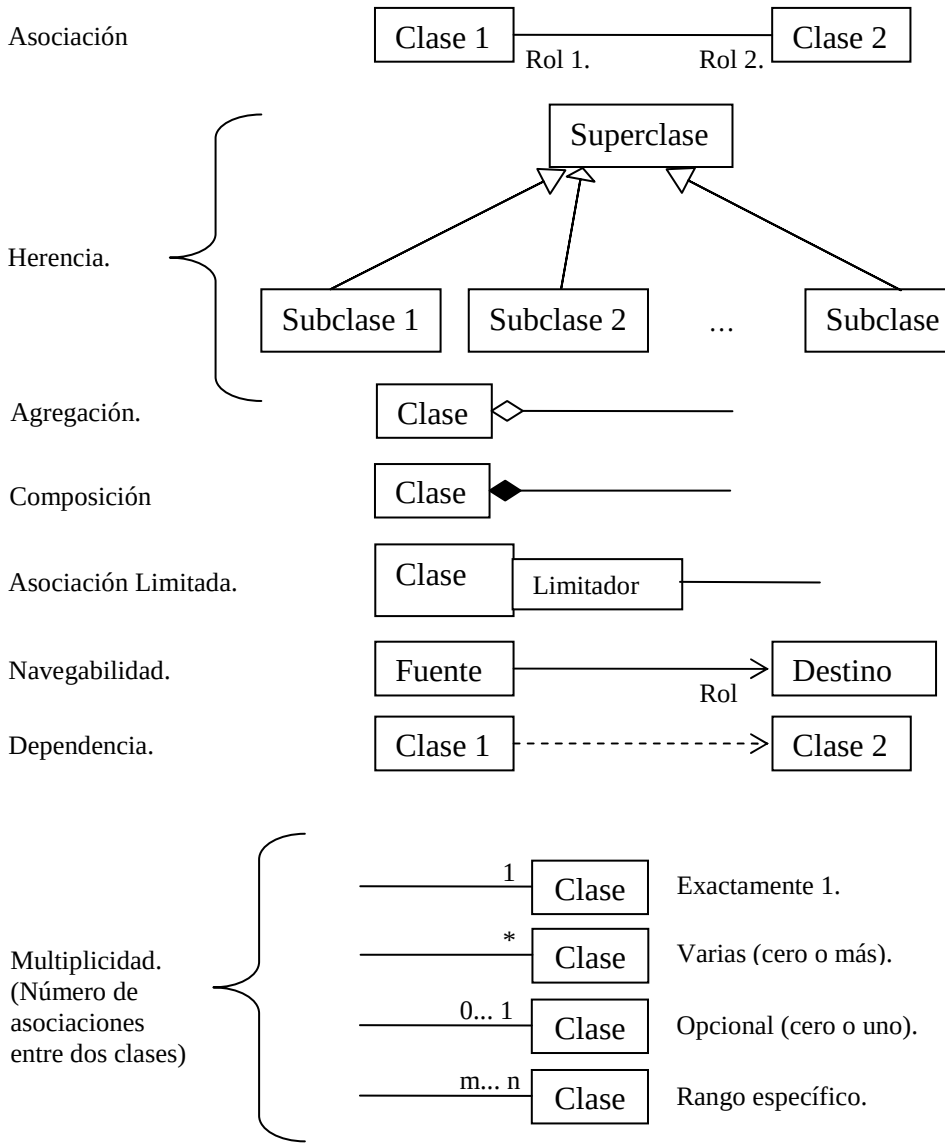


Figura 1. 46: Nomenclatura empleada para representar relaciones entre clases

Las utilidades de clase representan la encapsulación de abstracciones que proporcionan servicios algorítmicos que pueden estar constituidos por llamadas a funciones API de Windows, servicios para SGBD[14], los servicios de subprogramas entre otros servicios. Para declarar una utilidad de clase se emplea el estereotipo <<*utility*>> y es importante señalar que no se las puede heredar.

En los diagramas estáticos de UML, bajo ciertas circunstancias es importante señalar *restricciones* mediante el uso de condiciones semánticas que se deben

[14] SGBD: Sistema de Gestión de Base de Datos.

preservar (en donde se expresa la condición entre llaves y marcan las relaciones de multiplicidad que se deben cumplir entre las clases mediante el uso de una flecha entrecortada); estas restricciones se deben cumplir mientras el sistema esté en un estado estable. La notación empleada para marcar las restricciones se especifica en la figura 1.47. Existen también condiciones en las que se debe aclarar un diagrama, una clase, o simplemente hacer algún comentario sobre un elemento del diagrama; cuando esto sucede se emplean *notas* cuya simbología se encuentra en la figura 1.48.

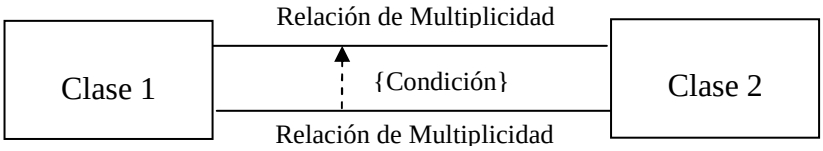


Figura 1. 47: Nomenclatura empleada para restricciones

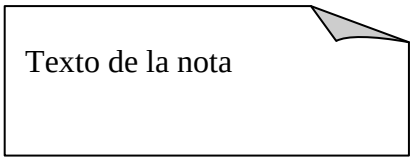


Figura 1. 48: Nomenclatura empleada para notas

Además de todos los elementos analizados, existe una forma no gráfica que se usa para proporcionar la definición completa de una entidad de la notación; en donde se considera que una entidad de la notación pueden ser clases, asociaciones, operaciones, o diagramas completos. Este elemento se conoce como una *especificación* y está compuesta por los siguientes ítems (que según el caso pueden o no ser empleados):

- ***Elementos Comunes:***
  - *Nombre:* identificador.
  - *Definición:* texto.
- ***Especificaciones de Clases:***
  - *Responsabilidades:* texto.
  - *Atributos:* lista de atributos.

- *Operaciones:* lista de operaciones.
- *Restricciones:* lista de restricciones.
- *Máquina de Estados:* referencia a la máquina de estados.
- *Control de Exportación:* [public | prívate | protected].
- *Parámetros:* lista de parámetros.
- *Persistencia:* [transitorio | persistente].
- *Concurrencia:* [secuencial | vigilada | síncrona | activa].
- *Complejidad Espacial*: expresión.

- ***Especificaciones de Operaciones:***
  - *Clase de Retorno:* referencia a una clase..
  - *Argumentos:* lista de argumentos.
  - *Calificación:* texto.
  - *Control de Exportación:* [public | prívate | protected].
  - *Protocolo:* texto.
  - *Precondiciones:* [texto | ref. a código fuente | ref. a diagrama].
  - *Post condiciones:* [texto | ref. a código fuente | ref. a diagrama].
  - *Excepciones:* lista de excepciones.
  - *Concurrencia:* [secuencial | vigilada | síncrona | activa].
  - *Complejidad Espacial*: expresión.
  - *Complejidad Temporall*: expresión.

1.4.2.11 Diagramas de Implementación

Mediante el uso de los Diagramas de Implementación se pueden indicar aspectos relativos a la implementación como estructura del código fuente y características propias del tiempo de ejecución. Existen dos tipos de diagramas de implementación: los *Diagramas de Componentes* que muestran la estructura del código fuente y muestran el orden de los paquetes; y los *Diagramas Desplegables* que muestran la estructura del sistema en tiempo de ejecución.

En los diagramas de componentes se pueden identificar tres tipos de diagramas:

- *Categorías de Clases:* Este diagrama es un agregado que contiene clases y otras categorías de clases; sirven para dividir el modelo lógico de un sistema bajo un criterio o criterios dados permitiendo clasificar, organizar o trabajar mejor en la implementación de las clases. Cada categoría de clases tiene su propio nombre y no es necesario indicar todas las clases contenidas sino solo las más representativas. En la figura 1.49 se esquematiza una categoría de clases genérica.

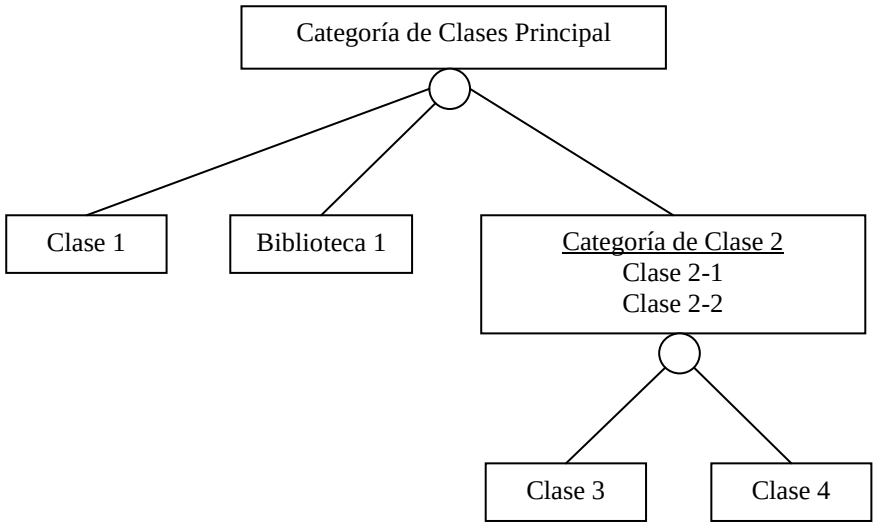


Figura 1. 49: Esquematización de una Categoría de Clases genérica

- *Diagrama de Paquetes:* Mediante estos diagramas se representan paquetes (que son grupos de elementos del modelo) del sistema. Los elementos que pueden ser contenidos son paquetes subordinados, subsistemas o modelos. En la figura 1.50 se ejemplifica un diagrama de paquetes.

- *Diagrama de Módulos:* Este tipo de diagrama sirve para indicar la asignación de clases y objetos a módulos en la vista física de un sistema. Se denotan tres tipos de ficheros: *programa principal* que contiene la raíz del programa, *especificación* que indica el fichero cabecera del módulo y *cuerpo* que denota el fichero cuerpo de cada módulo. Las dependencias entre módulos son representadas con una flecha que apunta al módulo respecto al cual existe la dependencia y hacen referencia a las

dependencias de compilación. Las agrupaciones de módulos relacionados lógicamente se representan con un rectángulo sombreado con bordes redondos y se conocen como subsistemas. En la figura 1.51 se indica la simbología empleada en los Diagramas de Módulos.

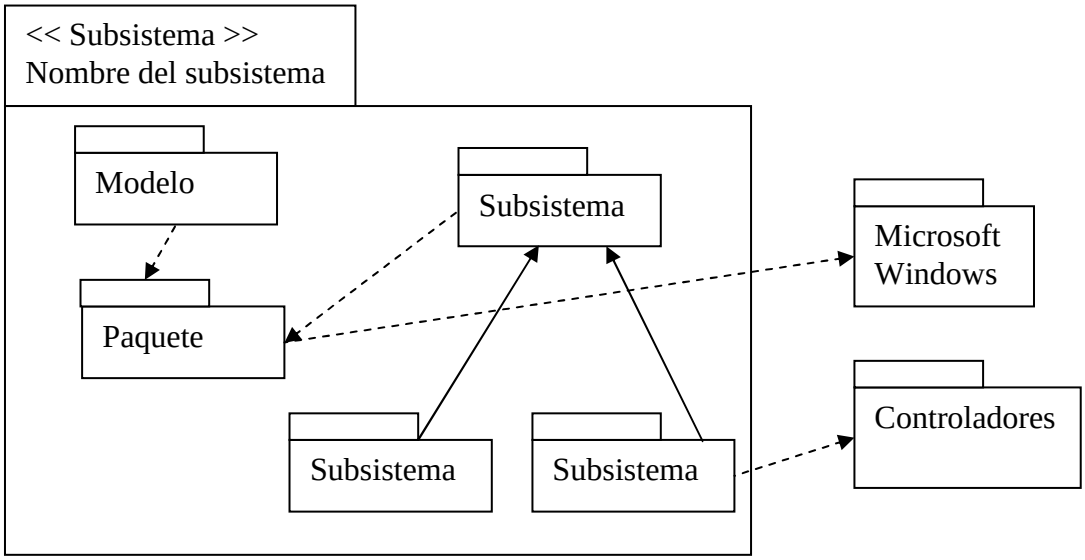


Figura 1. 50: Ejemplificación de un Diagrama de Paquetes

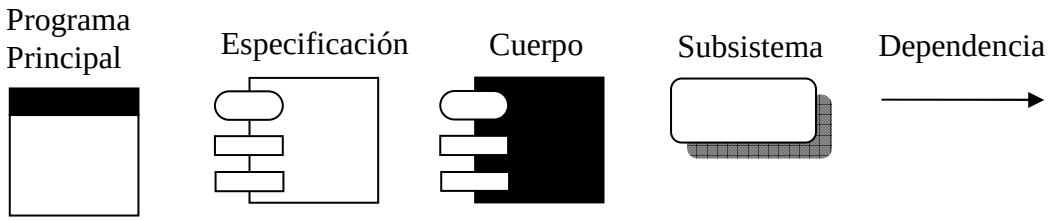


Figura 1. 51: Simbología empleada en los Diagramas de Módulos

Por otro lado los Diagramas Desplegables normalmente se definen a través de los *Diagramas de Procesos* que son los encargados de mostrar las relaciones entre los procesos y los procesadores a través del diseño físico de un sistema. Gracias a estos diagramas se podría definir políticas de planificación para la ejecución de procesos en un procesador. En este tipo de diagramas se definen tres elementos básicos: *procesadores* que representan a los dispositivos de hardware capaces de ejecutar programas, *dispositivos* que son los elementos de hardware incapaces de ejecutar programas y las *conexiones* que indican la comunicación entre procesadores y dispositivos. Además de estos elementos (representados

por los símbolos de la figura 1.52) se pueden definir íconos específicos de acuerdo a las necesidades presentes en el diagrama. En este tipo de diagramas también se puede dar el anidamiento (un elemento que está al interior de otro) de sus elementos y se pueden añadir especificaciones en modo texto.

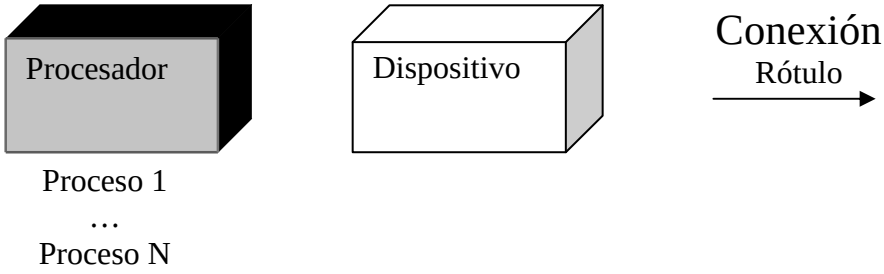


Figura 1. 52: Símbolos específicos empleados en los Diagramas de Procesos

1.4.3 VISUAL STUDIO 2005

Visual Studio es un producto de Microsoft que implementa un IDE (*Integrated Development Environment*), que no es más que un programa compuesto por un conjunto de herramientas orientadas al desarrollo de software para sistemas Microsoft Windows. El IDE de Visual Studio se puede utilizar con diferentes lenguajes de programación e incluye en forma de aplicación diferentes componentes que han sido empaquetados. Los principales componentes que se tienen son editores de código, compiladores, intérpretes, herramientas de automatización, depuradores y constructores de GUI's. Los lenguajes soportados por Visual Studio son:

- *Visual C++:* Es un IDE para Microsoft C, C++ y C++/CLI (*Command line interface*). Trabaja con las API's de Microsoft Windows, DirectX y Microsoft .NET Framework.
- *Visual C#:* Es un lenguaje de POO creado por Microsoft basado en C/C++ que utiliza la plataforma .NET (utiliza el modelo de objetos de dicha plataforma). Este lenguaje es muy similar a Java.
- *Visual J#:* Es un lenguaje de programación creado por Microsoft que permite usar la sintaxis de JAVA para crear aplicaciones y servicios.

- *ASP.NET:* Es un IDE/*Framework* creado para desarrollar aplicaciones web, sitios web dinámicos y servicios web XML. Este *Framework* está basado en ASP y utiliza como lenguaje base para la programación a Visual C++, Visual C#, Visual J# o Visual Basic .NET.
- *Visual Basic .NET:* Es un lenguaje de POO que emplea la sintaxis básica Visual Basic implementada sobre el *framework* .NET.

En la actualidad existen diferentes extensiones que se pueden instalar en Visual Studio para aceptar y trabajar con otros lenguajes de programación. Visual Studio 2005 incluye en sus entornos diseñadores de implantación (que permite validar el diseño de la aplicación antes de ser implantado). Adicionalmente, tiene un entorno para publicación web y pruebas de carga. Otra de las características importantes de Visual Studio 2005 es el soporte para compilación y librerías de 64-bits.

1.4.3.1 Visual Basic 2005

Visual Basic 2005 es un lenguaje programación orientado a objetos implementado sobre el IDE de Visual Studio 2005 y su sintaxis está basada en el lenguaje *BASIC* (*Beginners All - Purpose Symbolic Instruction Code*). El término *Visual* hace referencia a la posibilidad de crear interfaces gráficas (GUI's). Los elementos de programación (reglas de sintaxis) que emplea Visual Basic (VB) incluyen instrucciones, declaraciones, métodos, operadores y palabras clave.

Visual Basic .NET (VB .NET) al igual que todo lenguaje de programación maneja un conjunto de tipos de datos, los mismos que son tratados como clases, motivo por el cual además de ser declarados tienen que ser inicializados; estos tipos de datos se los puede clasificar en dos tipos: de *valor* que son los que llevan información en sí y los de *referencia* que contienen la dirección de memoria en donde reside un dato. Los principales tipos de datos nativos de VB .NET son: *byte, sbyte, int16, int32, int64, integer, uint16, uint32, uint64, single, double, boolean, char, decimal, y string (que en si es un arreglo de char).* Además de estos tipos de datos, se manejan los tipos de datos definidos por el programador, las clases definidas en el lenguaje y las clases creadas por el programador.

En la sintaxis empleada por VB .NET se pueden definir los siguientes conceptos y palabras clave:

- *Nombre de espacio:* Es el medio proporcionado por la plataforma de .NET a través del cual se pueden organizar las clases dentro del entorno, agrupándolas de un modo lógico y jerárquico. Para iniciar la región de un nombre de espacios se emplea la palabra clave y reservada *Namespace* y para marcar la parte final se usa *End Namespace.*

- *Ensamblados:* Conjunto de tipos y recursos reunidos para formar la unidad más elemental de código que se puede ejecutar en el entorno de .NET.

- *Class:* Término reservado para marcar el inicio de una clase. Para indicar el final de una clase se emplea el término *End Class.*

- *Módulos:* Similar a las clases ya que proveen un espacio en donde se pueden incluir declaraciones de variables, procedimientos, funciones, etc. La diferencia radica en el hecho que el código contenido en un módulo puede ser usado de forma directa sin necesidad de crear una instancia de dicho módulo. Para iniciar y finalizar un módulo se emplean los términos reservados *Module* y *End Module.*

- *Procedimientos:* Es una rutina de código que puede ser invocado desde cualquier punto del programa y ejecuta ciertas líneas de código. Un procedimiento puede tener como argumento una variable pasada por referencia o por valor. Si un procedimiento devuelve un valor entonces se lo conoce como función y usa las palabras reservadas *Function y End Function* para iniciar y terminar su declaración. En cambio si el procedimiento no devuelve un resultado entonces se lo conoce como un método y se utilizan las palabras clave *Sub y End Sub* para declararlo.

- *Propiedades:* Las propiedades se usan para configurar la forma que tendrá la clase. Una propiedad puede ser de sólo escritura, de sólo lectura o de escritura y lectura. Se limita el código de una propiedad mediante el uso de las palabras claves *Property y End Property*. Para asignar un valor a la propiedad se emplea la palabra *Set,* mientras que para obtener el valor de la propiedad se usa el término *Get*. Si se desea declarar una propiedad de solo lectura se emplea la palabra clave *ReadOnly* acompañada de Get o Set según se desee*.*

- *Evento:* Los eventos son métodos (Sub) que se ejecutan cuando el sistema operativo los provoca. Normalmente están asociados a un control.
- *Manejadores*: Un manejador permite controlar el evento indicando el procedimiento que se va a emplear. La palabra clave que se emplea es *Handle*.
- *Me:* Palabra clave que hace referencia a la clase actual.
- *MyBase:* Palabra clave que hace referencia a la clase desde la cual se heredó.
- *Variables:* Son espacios de memoria en los que se almacenan un valor y vienen definidos por un tipo de datos. Para declarar una variable en la parte interna de un método o función se emplea la palabra clave *Dim.* Una variable siempre tiene que ser declarada indicando el tipo de datos que contendrá.
- *Constantes:* Son datos que permanecen constantes a lo largo de la ejecución de la aplicación. Para declarar una constante se emplea el término clave *Const*.
- *Arreglos:* Los arreglos o matrices son un tipo de variable que permiten tener más de un elemento a los que se pueden acceder mediante un índice. Un arreglo se lo declara mediante la palabra clave *Array.*
- *Privilegio de Acceso:* VB .NET permite decidir quién puede acceder e instanciar a cada uno de sus componentes como las funciones, métodos, propiedades, tipos de datos, variables, constantes, arreglos y clases mediante el uso de tres términos claves y reservados. Estos tres términos hacen referencia a los tres niveles de acceso que son *private* que solo permite el acceso a nivel interno de la clase o espacio, *public* que permite total acceso y *friend* que permite el acceso solo a los miembros del mismo ámbito.

Otro aspecto importante de analizar además de los conceptos y las palabras clave de la sintaxis básica de VB son los operadores aritméticos, operadores lógicos y las estructuras de control que son permitidas.

Los operadores son aquellos elementos que permiten combinar variables, constantes, valores literales, instrucciones, etc.; adicionalmente estos operadores permiten comparar valores aritméticos y literales, permiten también evaluar expresiones devolviendo valores booleanos (verdadero o falso).

Los operadores aritméticos son aquellos que permiten realizar operaciones aritméticas y éstos son:

- *Suma (+):* Suma dos números o concatena una cadena de caracteres.
- *Resta (-):* Resta dos números o cambia el signo de un número.
- *Multiplicación (*):* Multiplica dos números.
- *División Real (/):* Divide dos números devolviendo el resultado con precisión decimal.
- *División Entera (\):* Divide dos números devolviendo como resultado un valor numérico entero.
- *Módulo (Mod):* Divide dos números y devuelve el módulo de la división.
- *Potenciación (^):* Eleva un número a una potencia.
- *Concatenación (&, +):* Permite unir dos o más cadenas de caracteres para formar una sola cadena.

Los operadores de comparación sirven para comprobar el nivel de igualdad o diferencia existente entre un operando y otro de una expresión; el valor que se obtiene como resultado es un valor booleano. Estos operadores son:

- Igual (=).
- Distinto (<>).
- Menor que (<).
- Menor o igual que (<=)
- Mayor que (>).
- Mayor o igual que (>=).
- Comparación de Cadenas (*Like*).

- Comparación de Objetos (*Is*).

Otro grupo importante de operadores son los operadores lógicos que permiten evaluar expresiones, estos operadores son:

- *And:* Realiza una operación de conjunción, si las dos expresiones son verdaderas entonces se devuelve un valor verdadero como respuestas caso contrario se devuelve falso.
- *Or:* Realiza una operación de disyunción, si una de las dos expresiones es verdadera el resultado es verdadero caso contrario es falso.
- *Not:* Realiza una operación de negación de una expresión.

VB .NET maneja algunas estructuras de control y bucles que son la herencia que ha recibido de la programación estructurada. VB .NET al ser un lenguaje orientado a objetos emplea estas estructuras en sus procedimientos como mecanismos de control y para facilitar ciertas operaciones. Estas estructuras son condicionales y emplean expresiones booleanas que usan los operadores antes descritos. En el anexo B se especifica la descripción y la forma de uso de los operadores.

- *If… End If.*
- *Select Case… End Select.*
- *While… End While.*
- *Do… Loop.*
- *For… Next.*
- *For Each… Next.*

Otra de las características importantes de VB .NET es el manejo de errores a través de las excepciones. Una excepción es un objeto generado por un error que contiene información sobre las características del error que se ha producido. VB .NET tiene una estructura que permite controlar las excepciones; en esta estructura se debe colocar el código que podría generar el error para después a través de una instrucción poder manipularlo.

Esta estructura es conocida como la estructura *Try… Catch* y su forma de uso es la siguiente:

Try

Código que podría generar un error

Catch nombreExcepción1 *As* tipoExcepción1

Código para el tipoExcepción1

[Catch nombreExcepción1 *As* tipoExcepción1

Código para el tipoExcepción1]

Finally

Código que debe ejecutarse haya o no error.

End Try

Un aspecto también importante que se controla en Visual Basic 2005 es el de las interfaces gráficas; las herramientas que permiten desarrollar aplicaciones gráficas compuestas por formularios y controles Windows, están contenidas en el espacio de nombres *System.Windows.Forms*; algunos de estos controles son implementados como objetos y se encuentran disponibles como componentes muchos de ellos *ActiveX*. A continuación se describirán las principales clases (controles de Windows) y su función.

- *Clase Form:* Esta clase contiene todos los miembros para la creación y manipulación de formularios. En una instancia de esta clase se colocan los controles de Windows.
- *Control Label:* Este control representa una etiqueta de texto.
- *Control Text:* Representa un cuadro de texto.
- *Control Button:* Representa un botón de pulsación.
- *Control ListBox:* Este control representa una lista de valores.
- *Control ComboBox:* Este control combina un cuadro de texto y una lista de valores desplegable.
- *Control CheckBox:* Representa una casilla de verificación.

- *Control RadioButton:* Representa un botón de opción auto excluyente.
- *Control GroupBox:* Este control es una caja de agrupación de controles.

Los controles descritos anteriormente son los más comunes por su alta frecuencia de utilización en aplicaciones de Windows, cada control representa una clase que puede ser instanciada, pero gracias al diseñador de formularios incluido en el IDE de VS el programador sólo se tiene que preocupar por la configuración de los controles y el manejo de los diferentes eventos asociados a cada uno de ellos.

El manejo de datos en Visual Basic 2005 se lo hace a través del modelo de acceso a datos incluido en la plataforma .NET Framework conocido como ADO .NET (*ActiveX Data Objects*). Este modelo ofrece un conjunto de herramientas para hacer consultas y posibilita también el trabajo en modo desconectado (lo que reduce el tráfico de red) empleando XML como formato universal de transmisión de datos. ADO .NET provee un conjunto de interfaces, clases, estructuras y enumeraciones que permiten el acceso a datos desde la plataforma .NET de Microsoft; entre estas características provee clases para realizar conexiones con diferentes proveedores de bases de datos, permitiendo ejecutar consultas y comandos basados en SQL o en OleDB. El espacio de nombres y clases de ADO .NET se encuentran contenidos en *System.Data*.

1.5 FINGERPRINT SDK [13]

En el ámbito de la programación existen muchas herramientas para aplicaciones específicas que no son incluidas en el IDE de VS 2005. Este conjunto de herramientas, que no son más que clases y componentes (algunos de ellos *ActiveX* y DLL's) vienen empaquetados por los diferentes proveedores en *kits* de desarrollo de software (SDK por sus siglas en inglés). La principal finalidad de estos SDK's es facilitar al programador la creación de aplicaciones ya que proporciona nuevas herramientas para la generación de aplicaciones específicas.

En el ámbito del manejo de huellas dactilares existen diferentes *kits* de desarrollo, pero uno de los más reconocidos e importantes por las características que ofrece, por las certificaciones que tiene y por los premios que ha obtenido es el *kit* de desarrollo *Fingerprint SDK* de Griaule Biometrics ©.

Este SDK maneja diferentes componentes según el ámbito en donde se lo emplea, de tal suerte, provee DLL's, componentes *ActiveX* y componentes para aplicaciones o *applets* de JAVA. *Fingerprint SDK* es una librería orientada al reconocimiento de huellas dactilares que puede ser integrada en diferentes aplicaciones biométricas.

Fingerprint SDK cumple con la normativa ANSI / NIST – ITL 1-2000. Además de cumplir con este estándar, ha obtenido la certificación de NIST y un certificado de reconocimiento otorgado por el FBI. *Fingerprint SDK* es el ganador de la competencia de verificación de huellas dactilares del 2006 (FVC por sus siglas en inglés). Por estos y otros méritos este SDK es una buena opción para el manejo de huellas dactilares.

1.5.1 CARACTERÍSTICAS

1.5.1.1 Especificaciones Técnicas

Las especificaciones técnicas de acuerdo al manual de *Fingerprint SDK* están divididas en tres grupos y son:

- *Captura*
    - Detección de la conexión o desconexión de un lector de huellas.
    - Detección automática del dedo.
    - Soporte de carga y almacenamiento de archivos BMP de huellas dactilares.
    - Tamaño máximo de imagen de 1280 x 1280 pixeles.
    - Tamaño mínimo de imagen de 50 x 50 pixeles.
    - Resolución máxima de 1000 DPI[15] y mínima de 125 DPI.
- *Extracción*
    - Tiempo promedio de extracción de una huella de 100 ms.
    - Tamaño máximo de imagen de 500 x 500 pixeles.

[15] DPI: Abreviatura para puntos por pulgada (*DOTS PER INCH*). Ésta es una medida de resolución de impresión, video o imágenes que representa el número de puntos individuales que pueden estar en un espacio lineal de una pulgada.

- Tamaño mínimo de imagen de 50 x 50 pixeles.
- Tamaño promedio del arreglo (arreglo que contiene la huella dactilar) de 400 bytes.

- *Comparación*
  - Velocidad de Identificación (Versión del SDK de Identificación): Hasta 35000 huellas por segundo.
  - Velocidad de Identificación (Versión del SDK de Verificación): Hasta 100 huellas por segundo.
  - Velocidad de Verificación (Versión del SDK de Identificación y Verificación): Hasta 100 huellas por segundo.

1.5.1.2 Lectores de Huellas Soportados

El SDK de *Griaule Biometrics* soporta diferentes lectores de huellas dactilares, para esto se tiene que instalar el software de control de cada uno de ellos. Dependiendo del lector, en cada máquina pueden estar uno o varios lectores funcionando al mismo tiempo. A continuación se indica la lista de lectores soportados:

- *Microsoft Fingerprint Reader*.
- *Microsoft Wireless IntelliMouse Explorer with Fingerprint Reader*.
- *Microsoft Optical Desktop with Fingerprint Reader*.
- *DigitalPersona U.Are.U 4000 y 4000B*.
- *Testech Bio-I CYTE*.
- *Secugen Hamster III*.
- *M2SYS M2-S Fingerprint Reader*.
- *Biotouch / Futronic FS80*.
- *Nitgen Hamster I / Hamster II*.
- *Certis Image Orcanthus*.
- *Crossmatch V250 / V300 / V300 LC / V300 LC2 / V500*.

1.5.1.3 Lenguajes de Programación Soportados

Otra característica importante de *Fingerprint SDK* es la compatibilidad con diferentes lenguajes de programación. Los lenguajes que soporta las DLL's, *ActiveX* y los componentes JAVA de *Fingerprint SDK* son:

- Java (Windows).
- Delphi.
- Visual Basic 6.0 y .NET.
- C++ y Visual C++.NET.
- Visual C#.
- VBA.
- Visual FoxPro 8.

Hay que destacar que cualesquiera que sea el lenguaje de programación éste debe funcionar sobre Windows, aunque esto no quita la posibilidad que mediante el uso de *applets* la aplicación final pueda funcionar sobre Linux.

1.5.1.4 Otras Características

A continuación se describirán otras características de este SDK:

- Este SDK tiene dos versiones la de *Identificación* y la de *Verificación*. Estas dos versiones solamente difieren de la velocidad de identificación según lo descrito en las características técnicas.
- La identificación se realiza de uno a muchos.
- Para que el SDK funcione apropiadamente se necesita un procesador mínimo de 200 MHz con 64 MB o más y espacio en disco de 20 MB.

1.5.2 LICENCIAMIENTO

Fingerprint SDK provee tres versiones de su SDK; la versión Trial, la de Identificación y la de Verificación. El Trial de este SDK dura 90 días y no tiene limitante alguno en lo que respecta a la funcionalidad. La versión de Identificación es útil cuando se va a trabajar con un gran número de huellas dactilares mientras

que la versión de Verificación es buena en ambientes donde lo principal no es identificar huellas sino verificarlas (no se tiene un gran número de huellas dactilares).

La ventaja del proceso de licenciamiento de *Fingerprint SDK* es que la licencia viene contenida en un archivo de texto y que solo tiene que ser copiado en la carpeta de la aplicación. En la tabla 1.6 se presentan los costos de licenciamiento de *Fingerprint SDK* tomados de la página web de Griaule [29]:

Licenciamiento / Versión	Versión de Identificación	Versión de Verificación
Single Computer	$ 36	$ 20
SDK Integrator 150 (Licencia hasta para 150 máquinas)	$ 3200	$ 1600
SDK Integrator 400 (Licencia hasta para 400 máquinas)	$ 6000	$ 3200
SDK Integrator 1500 (Licencia hasta para 1500 máquinas)	$ 10000	$ 6000
SDK Integrator 5000 (Licencia hasta para 5000 máquinas)	$ 16000	$ 10000
SDK Integrator 15000 (Licencia hasta para 15000 máquinas)	$ 25000	$ 16000

Tabla 1. 6: Tabla de Costos de Licenciamiento de *Fingerprint SDK*

En el anexo C se presenta un resumen de los métodos principales que maneja este SDK. Estos métodos son necesarios en la implementación de aplicaciones de verificación de huellas dactilares y están disponibles a través de un archivo *dll*.

1.6 SERVIDORES

Los servidores aparecen debido a la necesidad de las organizaciones de manejar más eficientemente sus operaciones, al igual que sus recursos, para de esta

manera reducir los costos y gastos de operación. Esto se lo logra mediante la centralización de los servicios en común que demandan los usuarios de dicha organización, estos servicios son provistos por uno o varios servidores.

1.6.1 CONCEPTOS

- *Servidor:* Un servidor es un ente (hardware o software) que ha sido designado para proveer un servicio. Los servicios más comunes son los de: almacenar, manejar, distribuir o procesar información. Estos servicios deben ser rápidos y eficientes, además de incorporar seguridad para proteger los respectivos procesos de ataques y salvaguardar la información que se maneja.

  Normalmente, un servidor implementa su servicio a través de un interfaz (por ejemplo un puerto TCP o UDP) que da a conocer al cliente; el servidor permanece en estado de espera hasta que llegue una petición o después de una transacción realizada por parte del cliente (host u otro servidor).

- *Dimensionamiento:* Es un proceso que consiste en encontrar un sistema (recursos de hardware y software) en base a los requerimientos de las aplicaciones que van a correr sobre el mismo para conseguir el desempeño esperado de las mismas.

- *Redimensionamiento:* Proceso de dimensionamiento aplicado en un sistema existente.

- *Planificación de capacidad:* Es un proceso que permite determinar los recursos de hardware y software necesarios para soportar futuros incrementos en la carga de trabajo de un sistema (producido por el incremento de usuarios, el aumento del uso de los recursos del sistema, o por la mejora de aplicaciones sea mediante funciones adicionales o nuevas aplicaciones), este proceso está basado en el monitoreo del sistema.

1.6.2 TIPOS DE SERVIDORES

Los servidores se clasifican de acuerdo a la función que desempeñan; en la tabla 1.7 se resumen los principales tipos de servidores.

Tipo de Servidor	Función

Archivo e impresión	Comparte y controla archivos e impresoras.
Aplicación	Comparte aplicaciones, por ejemplo: un servidor de una aplicación contable.
Correo Electrónico	Almacena correo electrónico. Ofrece interconexión para el cliente a través de protocolos como POP3 o SMTP y correo electrónico a través de Internet.
Conectividad de redes	Ofrece servicios de red, tales como: DHCP, enrutamiento de paquetes, cifrado/descifrado, VPN, entre otros.
WWW	Ofrece servicios Web y de noticias
Acceso Remoto	Permite a los usuarios acceder a una red local de manera remota.
Base de datos	Servidor que permite el acceso a la información contenida en una base de datos.

Tabla 1. 7: Tipos de Servidores y sus Funciones

1.6.3 ETAPAS DE UN SERVIDOR

Las etapas de un servidor corresponden a los procesos que se ejecutan a lo largo de la vida útil del mismo. Estas etapas son: *la selección del servidor* (diseño y dimensionamiento), *adquisición y producción del servidor*, y *el redimensionamiento* del mismo (en caso de ser necesario). Dentro de estas etapas se realizan una serie de tareas, representadas en la figura 1.53; para describir estas tareas se ha empleado como referencia la guía utilizada por IBM para dimensionamiento y planificación de capacidad de servidores, la misma que sirve para cualquier sistema [11]. Estas tareas se explican a continuación.

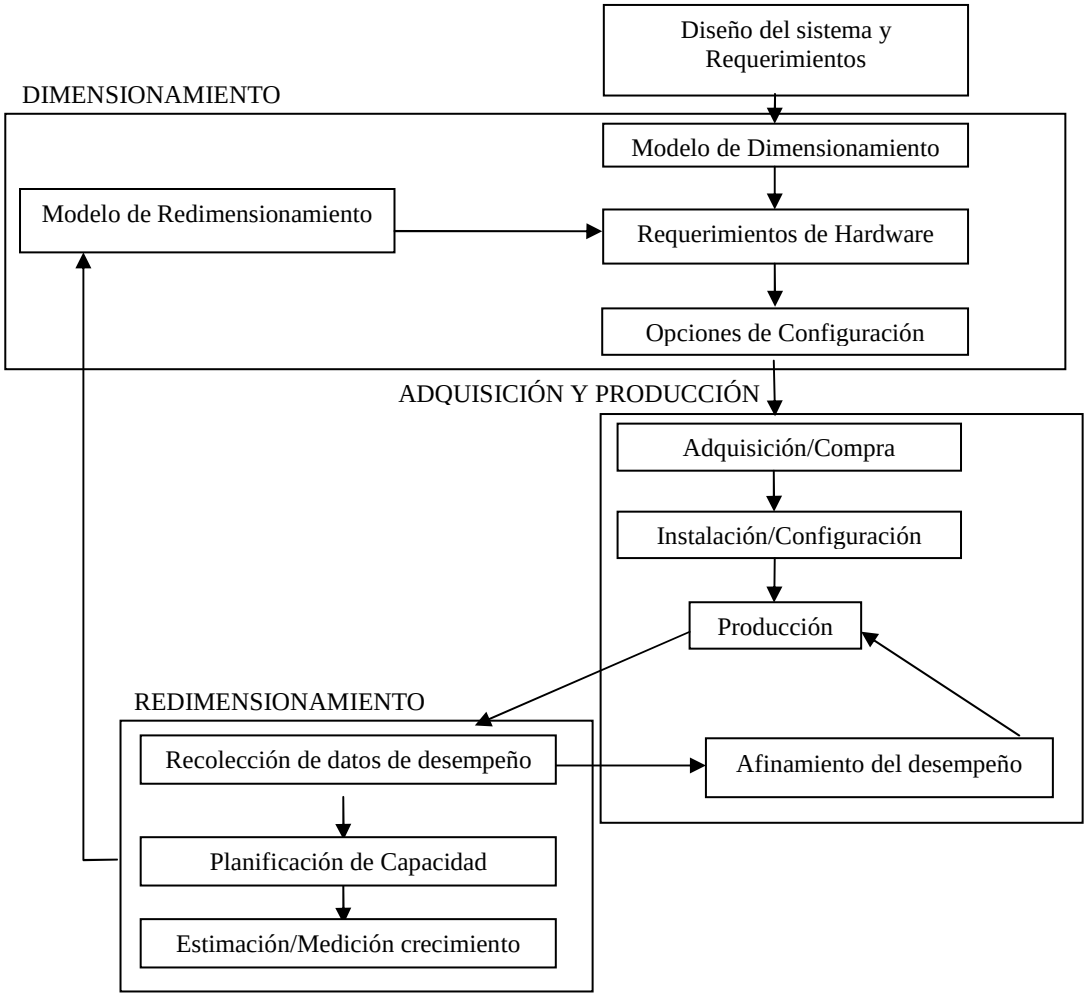


Figura 1. 53: Etapas del dimensionamiento de servidores

1.6.3.1 Diseño del Sistema y Requerimientos

La primera tarea a realizar previo al dimensionamiento de servidores es el diseño del sistema y el planteamiento de los requerimientos del mismo; esta tarea involucra determinar las reglas del negocio, construir un modelo de datos, seleccionar el software de aplicación y del *middleware*, determinar la infraestructura del departamento de tecnología de la información (IT), determinar las interfaces y comunicaciones con otros sistemas, entre otros puntos que permiten establecer las primeras bases para posteriormente realizar un proceso de dimensionamiento.

1.6.3.2 Modelo de Dimensionamiento

Esta tarea es la primera dentro del proceso de dimensionamiento y se basa en la recopilación de información del número de usuarios, transacciones y los volúmenes de datos (presentes o históricos) para estimarlos y proyectarlos conociendo así probables volúmenes de información que se manejarán; en función de estos valores y a medida que se avance en las tareas del dimensionamiento se determinarán los requerimientos de hardware, software y de la aplicación.

Un método práctico empleado en el modelo de dimensionamiento es *la segmentación*, que consiste en realizar preguntas que permiten determinar de manera rápida el tipo de dimensionamiento requerido. Estas preguntas pueden ser las siguientes:

- *¿Cuál es el tipo de aplicación?:* Es decir cuál es el trabajo que llevará a cabo el servidor, de esta manera se determina qué información debe estar disponible.
- *¿Qué hará el usuario?:* Permite clasificar a los usuarios en función del tipo de servicio que requieren, y así estimar la carga de trabajo que se generará por parte de cada uno.
- *¿Cuál es el tipo de interfaz de usuario?:* Se debe responder si la interfaz es un *web browser*, telnet, línea de comando, gráfica u otro tipo de interfaz. A través de esta pregunta se determina si la lógica de la aplicación del usuario está ejecutándose en un servidor o en la propia estación de trabajo.
- *¿Cuánta información está siendo procesada?*: Esta información se puede obtener de dos formas: con datos basados en la experiencia que el usuario ha tenido en sistemas similares, o con la estimación del número de usuarios por la cantidad de información de cada uno de éstos. Esta pregunta permite realizar una primera estimación de la capacidad de procesamiento que debería tener el servidor.
- *¿Cuál es el número de usuarios?:* Para responder esta pregunta principalmente se debe tomar en cuenta el número de usuarios que se espera tener cada día; esta estimación se la puede hacer en base a datos

históricos o estimaciones mediante estudios de mercado. Esta pregunta permitirá estimar la carga que tendrá el servidor.

- *¿Cuál es la frecuencia y el tamaño de las transacciones?:* Es un factor importante de analizar para determinar la carga de trabajo. Esta información puede obtenerse del tipo de usuarios y de la interfaz que será implementada; en caso de no tener valores exactos será necesario realizar suposiciones. Esta información es necesaria para determinar el tipo de procesador ya que una variación en la frecuencia o tamaño de las transacciones puede requerir mayores recursos del procesador y otros elementos.

1.6.3.3 Requerimientos de Hardware

Dependiendo de los factores previamente analizados y los requerimientos computacionales debidamente expresados, se procede a determinar los requerimientos de hardware basados principalmente por: el procesador, la memoria, el subsistema de discos, y los adaptadores.

- *La capacidad del procesador*: El procesador es el cerebro de la computadora, se encarga de traer las instrucciones y los datos desde la memoria para procesarlos, ejecutarlos y devolver los resultados a la misma; por esta razón es uno de los componentes más importantes a ser seleccionados en el dimensionamiento. Otra razón por la cual se debe hacer una buena elección del procesador es la económica ya que reemplazar un procesador y adaptarlo al sistema implica un costo elevado.

  La elección de un procesador principalmente debe estar basada en dos parámetros: *el número de núcleos de procesamiento* y *la frecuencia de trabajo de los mismos.* La cantidad de procesadores presentes en un servidor depende de algunos factores como el sistema operativo y el trabajo que va a realizar el servidor.

  Además de los factores descritos, también es importante tomar en cuenta las opciones que ofrecen las distintas compañías fabricantes de procesadores en el mercado. Los procesadores que normalmente se encuentran presentes en un servidor común por sus buenas prestaciones y

relativo bajo costo son los fabricados por AMD o los fabricados por Intel; aunque no se debe dejar de lado a los procesadores de Motorola, IBM o Apple también comunes.

- *Memoria:* La memoria es el segundo componente que se debe seleccionar cuando se dimensiona un servidor. El principal reto en la selección de la memoria es el escoger la mejor combinación entre velocidad y el tamaño, ya que una mala elección puede producir limitaciones computacionales en el servidor y como consecuencia (según el problema que se presente) se necesitaría reemplazar o añadir módulos de memoria. Idealmente se busca tener una memoria de gran capacidad y en donde el acceso a la misma sea inmediato, aunque en la práctica esto no es un hecho por motivos tecnológicos (en la práctica la velocidad es inversamente proporcional al tamaño). Para compensar esta situación se ha establecido una jerarquía de memoria que está esquematizada en la figura 1.54 y en donde sus elementos se explican en los párrafos subsiguientes.
- *Registros:* Los registros son las memorias más rápidas y su acceso es prácticamente inmediato ya que se encuentran dentro de la estructura del procesador. Esta memoria típicamente es menor a 1 KB y el CPU la emplea para almacenar información crítica y resultados temporales. Los registros se encuentran dentro del procesador para reducir el tiempo de acceso, ya que esta información se busca en pequeños espacios de memoria y no se pagan penalizaciones de tiempo por tener que acceder a otro tipo de memorias (de mayor jerarquía).
- *Caché:* Las memorias caché son volátiles, contienen una réplica de una parte de la memoria principal y su tamaño en la actualidad está en el orden de los MB. En función del tamaño de la memoria caché y de su cercanía (determinada por su orden de acceso) a la CPU se pueden clasificar en tres niveles: nivel 1 (L1 – on chip), nivel 2 (L2) y nivel 3 (L3); la caché de nivel 1 corresponde a la más pequeña y más rápida y se encuentra integrada en la CPU.

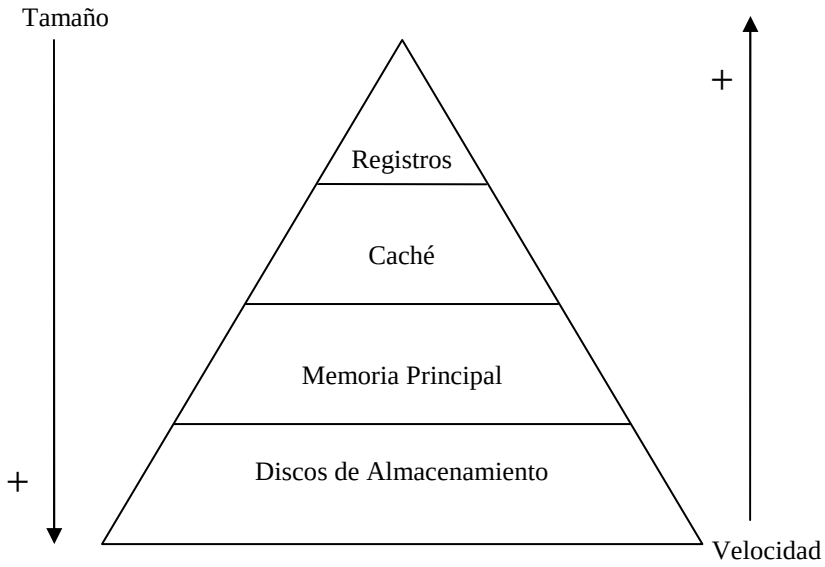


Figura 1. 54: Jerarquía de memoria

Es importante tomar en cuenta la estructura de los datos en las memorias caché, los cuales son organizados en líneas para que el procesador pueda encontrarlos de manera más rápida. Cada línea trae una etiqueta que hace referencia a la dirección en la memoria principal del primer byte correspondiente a la información deseada, acompañado por algunos datos adicionales de control. A continuación de la etiqueta viene la información en sí conformada por palabras (una palabra es un arreglo de 8 bits). Por tanto, si se tiene una memoria caché con líneas pequeñas la mayor parte de su capacidad será ocupada por etiquetas, reduciendo el tiempo de transferencia de datos entre la memoria principal y la memoria caché. Por otro lado el tener un mayor tamaño para cada línea incrementa el espacio para los datos, pero también aumenta el tiempo de transferencia de información con la memoria principal.

Otros factores que se deben considerar, que influencian los tiempos de acceso, son los *aciertos (hits)* y *desaciertos (miss)*; cuando los datos requeridos se encuentran en caché se produce un acierto, caso contrario se producirá un desacierto y será necesario acceder a la memoria principal en busca de dicha información, lo que involucra un costo de tiempo adicional; es por esta razón que

algunos sistemas implementan caché internas y externas para reducir el acceso a memoria principal, porque mientras mayor sea el número de aciertos mayor será el rendimiento del sistema.

- *Memoria Principal:* La memoria principal normalmente está constituida por una (o varias) RAM del tipo volátil. En cuanto a su desempeño esta memoria es más lenta que la caché y el acceso a la misma tarda mucho más, puesto que su tamaño es mayor. En la memoria principal se almacenan los datos que el ordenador está usando en un instante determinado, y son borrados de la misma cuando se apaga al ordenador, cuando dichos datos ya no necesitan ser utilizados o cuando se reemplazan por otros (ya sean de la misma aplicación o de otras aplicaciones).
- *Disco de almacenamiento:* El disco de almacenamiento (típicamente el disco duro) es el último nivel en la jerarquía de memorias. El acceso a esta memoria es el más lento de todos debido a ser el último nivel en donde se busca la información y al mayor espacio que se tiene en el mismo. Es importante destacar que no todo el disco duro es empleado como memoria sino tan solo la parte que fue destinada para paginación o memoria virtual.

- *Subsistema de disco:* En la selección del subsistema de disco se toman en cuenta aspectos como el *número de unidades de disco* y la *arquitectura de almacenamiento*. El número de unidades de disco hace referencia al número de dispositivos de almacenamiento a utilizar, pudiendo tener un arreglo de discos o simplemente un dispositivo de almacenamiento. La arquitectura de almacenamiento explica la forma en que se accederá a las unidades de disco para almacenar la información; dentro de estas arquitecturas las más populares en el mercado son las siguientes:
  - *DAS (Directly Attached disk Storage):* Es el método de almacenamiento más básico y su configuración es la más simple. En esta arquitectura un solo disco es conectado físicamente al servidor u ordenador como se puede observar en la figura 1.55; existen variantes de la arquitectura en donde también se puede tener un

subsistema de discos (configurados separada e independientemente) o un arreglo de discos (RAID).

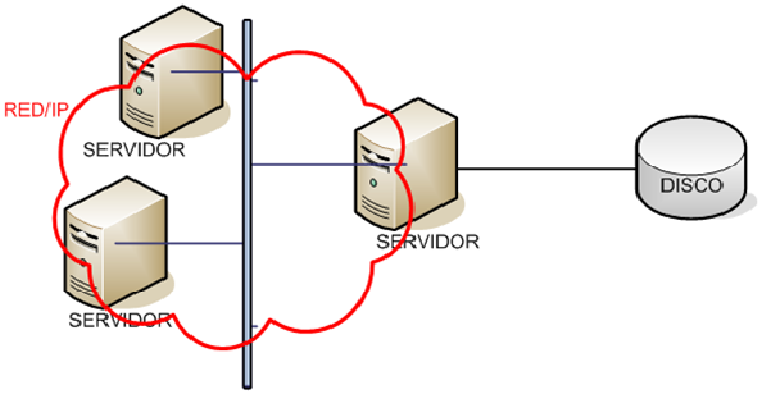


Figura 1. 55: Configuración Básica de Arquitectura DAS

Los protocolos y modos de almacenamiento más usados para DAS son: SCSI (*Small Computer System Interface*), *Fiber Channel* (FC) y *Serial Storage Architecture* (SSA).

El bus SCSI como medio de almacenamiento provee un canal de alto rendimiento y confiabilidad para los datos entre el servidor y la unidad de almacenamiento. Presenta limitaciones de escalabilidad, conectividad y ubicación (por la distancia). SCSI como un protocolo especifica los comandos para leer y escribir los bloques de datos entre el servidor y el dispositivo de almacenamiento; estos comandos son emitidos por el sistema operativo en respuesta a las peticiones de datos por parte del usuario.

Fiber Channel como modo de almacenamiento es el estándar utilizado para redes; este estándar combina las características de un bus con las de una red (distancia y conectividad) gracias al uso de la fibra óptica que es el medio por el cual el dispositivo de almacenamiento está unido al servidor con una tecnología punto a punto. *Fiber Channel* ha sido estandarizado por el grupo T11 del Comité Internacional para Estándares de Tecnologías de la Información y maneja su propio formato de trama, pero con la

ventaja de soportar diferentes protocolos de transporte, tanto de almacenamiento (DAS) como de paquetes (IP).

SSA especifica una interfaz serial que permite conectar al servidor con diferentes dispositivos de entrada y salida como: controladores de cinta controladores de disco, controladores ópticos, CDRoms y otros subsistemas de almacenamiento.

- *NAS (Network Attached disk Storage):* Esta arquitectura transfiere los datos entre los dispositivos por medio de la red usando TCP/IP. El dispositivo de almacenamiento no está unido directamente a la LAN sino que lo hace mediante un servidor dedicado o un NAS *appliance*[16]. En la figura 1.56 se representa un esquema de la configuración de un NAS.

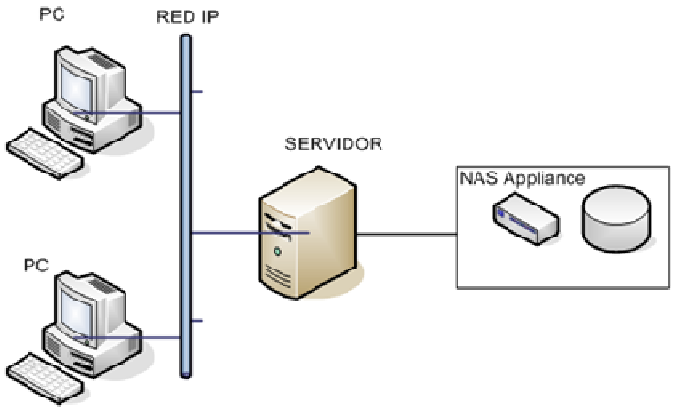


Figura 1. 56: Configuración NAS

- *SAN (Storage Area Network):* Esta arquitectura trabaja como una LAN porque permite la comunicación entre todas las unidades de almacenamiento dentro de una red como se muestra en la figura 1.57, para lo cual hace uso de elementos de interconexión como *routers*, *gateways*, *hubs* y *switches*. El modo de almacenamiento por defecto con el que trabaja esta arquitectura es *Fiber Channel* (FC).

  Otra característica importante de SAN es que permite que servidores heterogéneos[17] compartan herramientas comunes de

[16] NAS *appliance* es un dispositivo que funciona como servidor, tiene un sistema operativo reducido (o embebido) y su función es abastecer la información a ser compartida por múltiples servidores.
[17] Servidores con características diferentes de hardware y software

almacenamiento (disco, cinta) las cuales pueden estar a kilómetros de distancia.

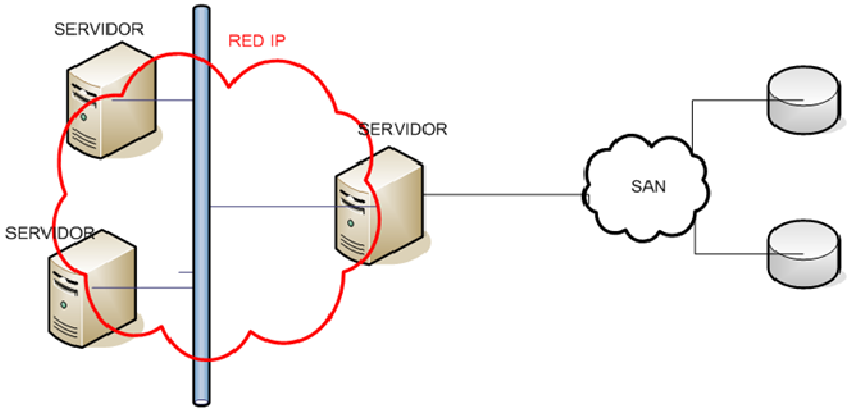


Figura 1. 57: Configuración SAN

En la tabla 1.8 se presenta un cuadro comparativo de las características principales de las arquitecturas de almacenamiento explicadas anteriormente; esta tabla permite en base a los requerimientos escoger la arquitectura más adecuada para el servidor que se está dimensionando. Por ejemplo, si se va a tener una configuración de varios servidores que comparten varios recursos y además se encuentren a grandes distancias, la configuración SAN resultará más adecuada. Mientras que, para un sistema con un solo servidor que manejará gran capacidad de almacenamiento la arquitectura DAS es la mejor.

- *Adaptadores:* El último componente que se especifica en los requerimientos de hardware son los adaptadores. Un adaptador se encarga de recibir la señal que llega de un dispositivo y lo convierte en un bloque de bytes que se copiarán a la memoria; del mismo modo leerá de la memoria los bytes requeridos por el dispositivo y serán convertidos a la señal que maneja el mismo. Siempre deben especificarse los adaptadores de los dispositivos que conformarán el servidor, de tal suerte se tienen que especificar los adaptadores de red, de disco, de impresora, entre otros (dependiendo del tipo de dispositivos adicionales que se tengan). La selección dependerá del tipo de dispositivo, así por ejemplo un adaptador de red dependerá del tipo de conexión que se dispondrá en el servidor (por

ejemplo Ethernet, WLAN, fibra óptica) o un adaptador de disco dependerá de la arquitectura seleccionada.

Arquitectura / Característica	DAS	NAS	SAN
COSTO	Bajo	Medio	Alto
IMPLEMENTACIÓN	Fácil	Fácil	Compleja
EFICIENCIA	Alto	Alta	Alta
ESCALABILIDAD	Baja	Alta	Alta
CONEXIÓN	Dedicada	A múltiples redes	Varios servidores
RESPALDOS Y PROTECCIÓN	Si (costosos)	-	Si
RECURSOS COMPARTIDOS	No	Si (Limitada por NAS Appliance)	Si
COMPATIBILIDAD	Alta	Alta	Baja
INTEGRIDAD	Alta	No es prioridad	Alta
DISTANCIA	Cortas	Limitadas por el medio de transmisión de la red	Largas

Tabla 1. 8: Características de Arquitecturas de almacenamiento

1.6.3.4 Opciones de Configuración de Servidores

Una vez planteados los requerimientos de hardware, dentro del dimensionamiento se procede a determinar la configuración más adecuada del servidor; es decir si será por ejemplo un solo sistema, o una granja de varios sistemas pequeños que cumplan con los requerimientos anteriormente establecidos (*cluster*[18]). La selección de una configuración depende de las necesidades de la organización en base a las características que ofrece cada configuración; los principales parámetros a tomar en cuenta son la flexibilidad de crecimiento (en *clusters*), la minimización de costos (supercomputador) y la disponibilidad del servicio.

Hay que recalcar que esta etapa sólo sirve para determinar la configuración y no se debe implementar aún la opción seleccionada.

[18] Un *cluster* corresponde a un grupo de computadores interconectados mediante alguna tecnología de red, que en conjunto son percibidos como un solo computador. Ofrecen disponibilidad, eficiencia, rendimiento y escalabilidad.

1.6.3.5 Adquisición, Instalación y Producción del Sistema

Estas tareas son posteriores al proceso de dimensionamiento; en base a los requerimientos generados en dicho proceso se puede ejecutar las tareas de selección del proveedor de equipos (mediante licitación por ejemplo), adquisición de todos los componentes necesarios (Hardware y software), instalación de los mismos y puesta en operación del sistema.

1.6.3.6 Recolección de Datos de Desempeño

La recolección de datos sobre el desempeño del sistema se lo hace a través del monitoreo de sus componentes y en base a los resultados se realiza un análisis del comportamiento de los mismos.

Después del análisis del comportamiento del sistema, si el resultado indica que uno o más componentes están trabajando a su máxima capacidad provocando que el servicio se vuelva deficiente o inestable; dependiendo del proveedor del servicio, del impacto que tenga esta situación sobre los usuarios y/o de la relación costo beneficio que esto implique entre otros factores, se podría realizar un afinamiento del sistema o una planificación de capacidad.

1.6.3.7 Afinamiento del Sistema

En esta etapa se trata de mejorar el desempeño del sistema mediante una utilización más eficiente de los recursos existentes.

1.6.3.8 Planificación de Capacidad

Esta tarea es parte del redimensionamiento del sistema y permite determinar los recursos de hardware y software adicionales que necesita el sistema actual, debido a que uno o más componentes del mismo están alcanzando su máxima capacidad y no es posible solucionarlo mediante un “afinamiento”.

1.6.3.9. Modelo de Redimensionamiento

Esta tarea corresponde a la última dentro de la etapa de redimensionamiento; el modelo de redimensionamiento se lo emplea en un sistema que se encuentra en producción y usa técnicas similares a las empleadas en el dimensionamiento de un servidor. Para proceder a realizar un modelo de redimensionamiento se

necesita medir el uso de los recursos, el número de usuarios, y el tamaño y velocidad de las transacciones del sistema actual. Normalmente el modelo de redimensionamiento intenta aprovechar lo existente como las configuraciones del sistema o las arquitecturas implementadas.

1.6.4 SELECCIÓN DEL SERVIDOR

La selección del servidor consiste en una etapa del servidor en la que se elige el sistema más adecuado en función de los requerimientos determinados en el proceso de dimensionamiento, el cual involucra las siguientes tareas: *modelo de dimensionamiento, requerimientos de hardware y configuración del servidor,* las cuales fueron definidas anteriormente.

1.7 SEGURIDAD

La seguridad es un proceso que consiste en garantizar que un elemento está libre de peligro, riesgo o daño en un ambiente dado y bajo ciertas condiciones. En el mundo de la informática, la seguridad de los datos está orientada a garantizar que la información de un individuo u organización no sea dañada o alterada por ataques de personas no autorizadas o elementos como virus, gusanos, troyanos, entre otros. Por esta razón se han buscado diferentes mecanismos de seguridad para proteger los datos en las computadoras y en los medios de almacenamiento (esta protección es conocida como seguridad informática), así como al momento de ser transmitidos en una red de comunicaciones (como por ejemplo sistemas distribuidos, redes LAN o internet). La seguridad informática y la seguridad orientada a redes están enfocadas a la protección de diferentes elementos; pero solo la combinación de las dos permite brindar una seguridad integral a la información al proteger sus principales propiedades; las mismas se enumeran a continuación:

- *Confidencialidad:* Esta propiedad hace referencia al hecho que la información sólo puede ser accedida por quien esté autorizado a hacerlo.
- *Disponibilidad:* Se procura que en un sistema informático la información esté disponible para los usuarios legítimos siempre que se la necesite.

- *Integridad:* Asegura que la información solo puede ser modificada por partes (personas o sistemas) autorizadas.
- *Autenticación:* Asegura que la persona que va a acceder a la información o que va a establecer (iniciar o aceptar) una comunicación es quien dice ser.

Estas propiedades pueden verse comprometidas debido a la presencia de ataques; definiendo a un ataque como cualquier acción que compromete a la seguridad de la información y pueden ser:

- *Ataques pasivos:* Estos ataques se presentan cuando personas no autorizadas capturan o monitorizan la información sin alterar su contenido atentando a la confidencialidad de la misma. Un ejemplo de este tipo de ataque es el análisis de tráfico usando *sniffers*[19].
- *Ataques activos:* Ocurre cuando personas no autorizadas interactúan directamente con el sistema o la información alterando su contenido, inhibiendo su disposición o creando información falsa. Estos ataques pueden ser: denegación de servicio, enmascaramiento, reenvío (reproducción) o modificación de mensajes.

Existen tres aspectos importantes que se deben considerar cuando se desea implementar políticas de seguridad a nivel organizacional y/o tecnológico, y éstos son:

- *Ataque de seguridad:* Estos ataques generalmente se pueden presentar de cuatro formas distintas.
    - *Interrupción:* Es un ataque a la disponibilidad de la información debido al daño de un elemento que almacena o transporta la misma.
    - *Intercepción:* Ataque a la confidencialidad de la información debido a que un tercero no autorizado accede a la información o parte de la misma.

[19] Un *sniffer* es un programa que captura paquetes que circulan en la red a la que pertenece el equipo, luego de capturarlos estos paquetes pueden ser analizados y obtener cierta información.

- *Modificación:* Este hecho se da cuando una tercera persona no autorizada accede a la información y la altera, este ataque afecta la integridad de la información.
- *Fabricación:* Sucede cuando una persona no autorizada crea información personificando a un ente autorizado. Este ataque corresponde a un ataque de autenticidad.

- *Mecanismo de seguridad*: Otro aspecto a considerar son los mecanismos de seguridad; éstos son diseñados para detectar ataques de seguridad en cualquiera de sus formas, para prevenir de ataques que pueden comprometer las características de la información y para recuperar la información en caso de verse afectada por algún tipo de ataque. Uno de los mecanismos más destacados es la *criptología* el cual se encarga del estudio de sistemas que manejan cifrado de mensajes por medio de claves. Este mecanismo involucra dos áreas: la *criptografía* y el *criptoanálisis*; la criptografía consiste en cifrar mensajes para brindar seguridad a la información y el criptoanálisis trata de romper dicha seguridad por medio del descifrado de los mismos.
- *Servicio de seguridad:* Finalmente un servicio permite fortalecer la seguridad de la información almacenada o transmitida de una organización. Los servicios de seguridad están basados en uno o varios mecanismos de seguridad que permiten proteger las propiedades de la información enumeradas anteriormente.

1.7.1 CRIPTOGRAFÍA

Uno de los mecanismos de seguridad más importantes que se tienen es el de la criptografía, que consiste en cifrar y descifrar la información mediante el uso de algoritmos y claves. Existen diferentes algoritmos de encripción y se usará el más adecuado o una combinación de ellos en función del tipo de aplicación y el nivel de seguridad deseado. Estos algoritmos se encuentran clasificados en dos grupos: los algoritmos *de encripción convencional y* los de *encripción con llave pública*.

1.7.1.1 Conceptos

- *Llave:* Conjunto de bits que sirven para cifrar o descifrar información mediante el uso de un algoritmo.
- *Texto plano:* Representa los datos en su estado natural a los que se les aplicará un algoritmo de encripción.
- *Texto cifrado:* Representa los datos después de haber aplicado el algoritmo de encripción al texto plano.
- *Sustitución:* Mecanismo en el que se reemplazan uno o varios datos del texto plano en base a una tabla o patrón dado.
- *Permutación:* Mecanismo mediante el cual se intercambian uno o varios datos de un texto plano en base a una tabla o patrón dado.
- *Operaciones modulares:* Operaciones aritméticas basadas en el módulo de un número con respecto a otro (el módulo corresponde al residuo de la división entre dichos números). Así se tiene que: *a mod b* es el residuo de a para b.
- *Cajas S (S-Boxes):* Son matrices de números enteros que permiten reducir el tamaño de un bloque de datos.
- *Swapping (intercambio):* Consiste en tomar la mitad de los bits del lado derecho de un bloque de datos e intercambiarlos con la mitad de los bits del lado izquierdo del mismo.
- *Shift circular:* Es un cambio circular de posición en el que una cadena de *n* datos se divide en una mitad derecha y otra izquierda. En caso de ser un *shift* circular hacia la izquierda para cada mitad se toma uno o más bits del inicio (pero siempre un número menor a *n/2*) y se trasladan al final de la respectiva mitad. En cambio si es un *shift* circular hacia la derecha para cada mitad se toma uno o más bits del final (pero siempre un número menor a *n/2*) y se trasladan al inicio de la respectiva mitad.

1.7.1.2 Encripción Convencional

También conocida como encripción simétrica; los algoritmos basados en este tipo de encripción usan una misma llave para encriptar y desencriptar. La fortaleza de estos algoritmos se encuentra en mantener secreta la llave que será compartida por el emisor y el receptor, esta llave debe ser intercambiada entre ambos mediante un canal seguro.

El esquema de la figura 1.58 es una representación básica de la encripción convencional en donde el mensaje M o texto plano es encriptado usando una llave compartida K; luego el mensaje encriptado Ek(M) es enviado al receptor, éste desencripta el mensaje con la llave K mediante un proceso inverso a la encripción y obtiene el mensaje original o texto plano M.

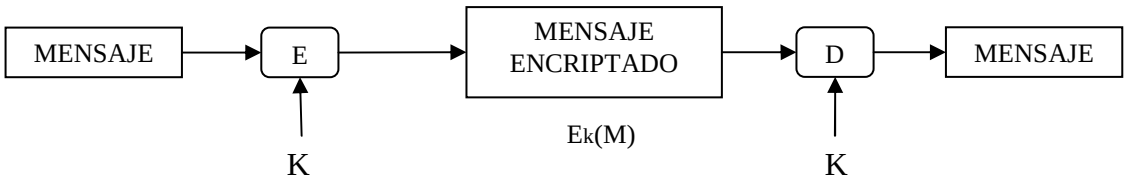


Figura 1. 58: Encripción convencional

Los algoritmos de encripción convencional pueden cifrar bloques de datos (*block cipher*) o un flujo de datos (*stream cipher*). Para el cifrado de bloques se cifra un bloque a la vez, mientras que para el cifrado de flujo de datos se cifra un bit (o byte) a la vez. De estos dos métodos el cifrado de bloque es el más usado y dependiendo del modo de operación, puede lograr el mismo efecto que el cifrado de flujo.

Los algoritmos de cifrado de bloque simétricos más destacados por su fortaleza o gran difusión en diferentes aplicaciones existentes son: 3DES, AES, IDEA, Blowfish, RC5 y RC2. Estos algoritmos surgieron después de la introducción de DES, uno de los algoritmos más importantes y aún empleado, el cual se describirá brevemente a continuación junto con los anteriormente numerados.

- *DES (Data Encryption Standard)*: Es un algoritmo de encripción convencional de bloque que recibe como entrada bloques de 64 bits y usa llaves de 56 bits; la figura 1.59 es una representación general del algoritmo DES.

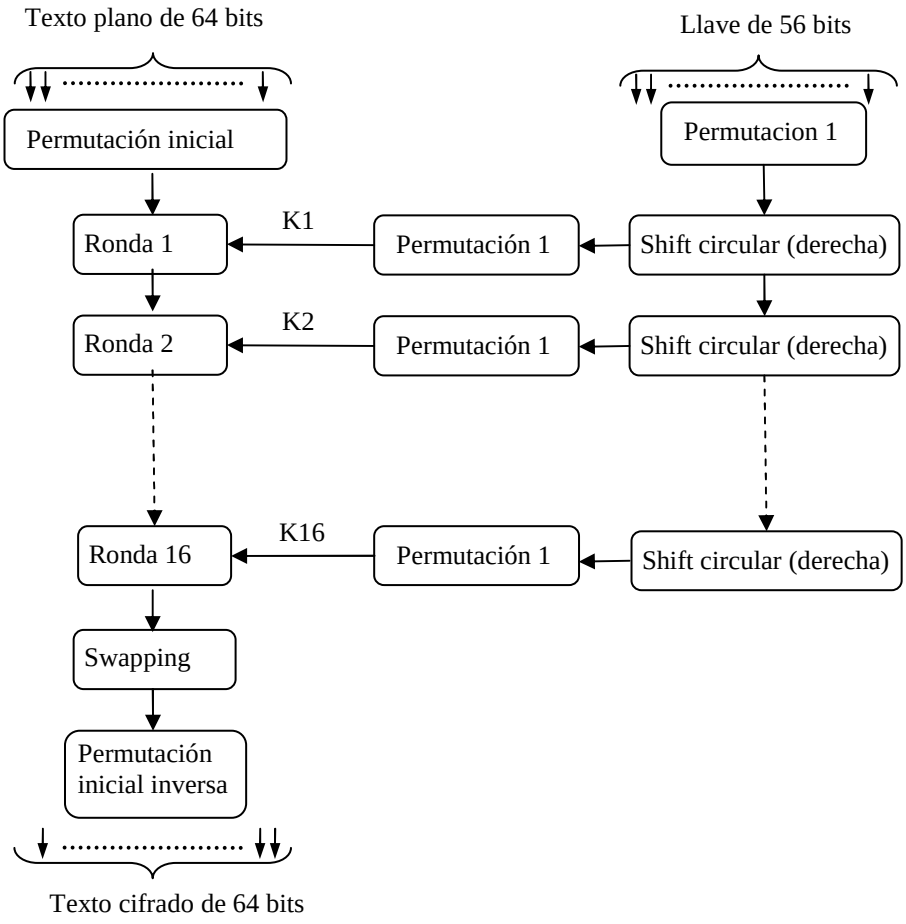


Figura 1. 59: Algoritmo DES[8]

En la parte izquierda del diagrama puede observarse las operaciones aplicadas al bloque de texto plano de 64 bits, mientras que en el lado derecho se observa las operaciones aplicadas a la llave de 56 bits que cifrará el correspondiente bloque de datos.

Para cada ronda se realiza el procedimiento esquematizado en la figura 1.60. Como se puede observar los pasos que se siguen son los siguientes:

1. El bloque de 64 bits recibido a la entrada de la ronda se divide en dos mitades, una derecha y una izquierda (Mitad D-1, Mitad I-1 respectivamente).
2. La mitad Derecha de 32 bits se somete a una expansión basada en una tabla dada, obteniéndose 48 bits a la salida.

3. La salida de la expansión se somete a un OR exclusivo con la llave correspondiente a la ronda.
4. Se sustituye el bloque obtenido mediante las S-boxes, obteniéndose a la salida un bloque de 32 bits.
5. Se permutan los bits según una tabla.
6. El bloque resultante de la permutación es sometido a un OR exclusivo con la mitad izquierda del bloque de entrada, el resultado corresponde a la mitad derecha del bloque de entrada a la siguiente ronda.
7. La mitad derecha del presente bloque antes de someterlo a toda la serie de operaciones anteriormente indicadas, se lo toma para conformar la mitad izquierda del bloque de entrada a la siguiente ronda.

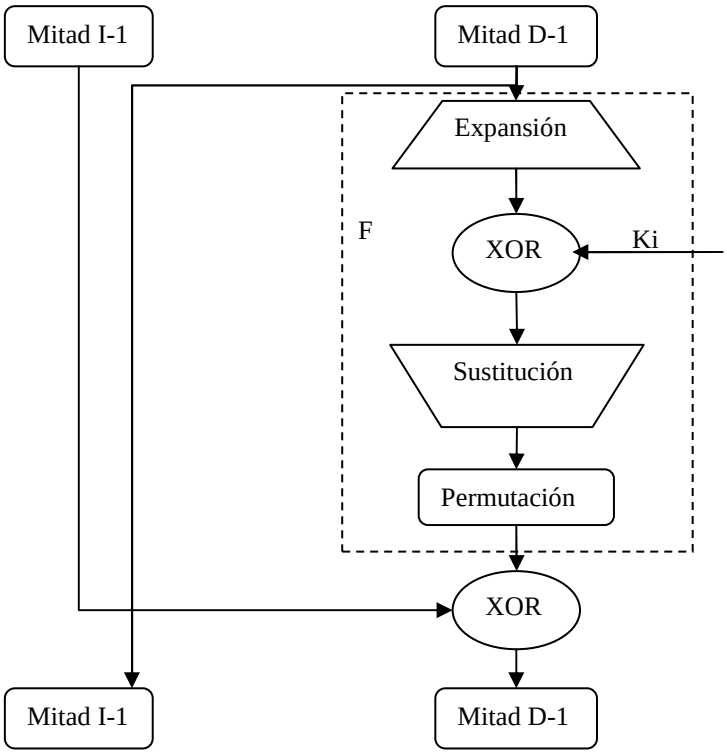


Figura 1. 60: Procedimientos de una ronda de DES[8]

DES ha sido muy usado desde 1976 ya que fue aprobado como estándar federal, y a pesar de las vulnerabilidades que presenta a los ataques de fuerza bruta, debido a que la longitud de su llave es tan solo de 56 bits, sigue siendo

utilizado como un mecanismo de seguridad básica en ambientes donde la confidencialidad de la información no es un factor crítico, donde es muy poco probable que se presenten ataques de fuerza bruta para recuperar la llave secreta o como complemento en aplicaciones que manejan otros mecanismos de seguridad adicionales (por ejemplo comercio electrónico).

- *3DES:* Debido a las vulnerabilidades de DES por la longitud de su llave (56 bits) se buscó la forma de incrementar el tamaño de la llave de manera efectiva, para lo cual se diseñó una versión modificada al algoritmo DES que consiste en aplicar tres veces DES para la encripción de un texto plano y para lo cual se han desarrollado dos posibilidades: Triple DES con dos llaves o triple DES con tres llaves.

  Para el primer caso se duplicó el tamaño efectivo de la llave a 112 bits, para entender mejor su implementación. En la figura 1.61 se pueden observar las tres etapas de 3DES con dos llaves: la primera corresponde a la encripción del texto plano con una llave compartida K1 cuya salida será la entrada a la siguiente etapa, en la segunda se desencripta el bloque recibido con una llave K2 y finalmente en la tercera etapa se recibe la salida de la segunda etapa y se vuelve a encriptar con la llave K1. El objetivo de la encripción, desencripción y encripción con dos llaves diferentes es mantener la compatibilidad con sistemas que trabajen solo con DES, para lo cual se iguala K1 a K2.

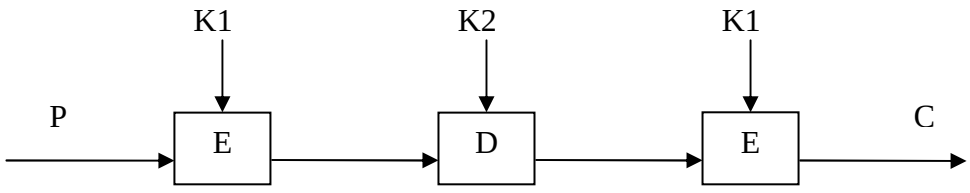


Figura 1. 61: Algoritmo Triple DES

Para el segundo caso de tres llaves el tamaño efectivo de la llave corresponde a 168 bits (con lo que se dificulta aún más el trabajo del criptoanalista[20]) y se aplican las tres etapas mencionadas anteriormente (EDE - Encripción Desencripción Encripción) pero cada etapa emplea su propia llave.

[20] Persona que práctica el Criptoanálisis.

La fortaleza de este algoritmo radica en el tamaño efectivo de su llave ya que esto dificulta el criptoanálisis, es por esta razón que se mantiene como la segunda mejor alternativa para encripción convencional.

- *AES:* Debido a las debilidades que comenzó a presentar DES apareció la necesidad de buscar un nuevo estándar, para lo cual el gobierno de Estados Unidos a través de NIST (Instituto Nacional de Estándares y Tecnología) convocó a un concurso a nivel mundial para escoger el nuevo algoritmo estándar de encripción simétrica que sería llamado AES (*Advanced Encryption Standard*). Los requisitos que debía cumplir el algoritmo eran:
    - Encripción simétrica de bloques.
    - El diseño debe ser público.
    - Soportar llaves de longitud de 128, 192 y 256 bits.
    - Debe poder implementarse en hardware y software.
    - El algoritmo será público o con licencias de carácter no discriminatorio.

  La propuesta ganadora fue el algoritmo *Rijndael*, que usa permutaciones, sustituciones y varias rondas que dependen del tamaño de la clave y el bloque. Trabaja a nivel de bytes y no de bits, de donde la entrada es un arreglo de 16 bytes (128 bits) correspondientes al texto plano que se almacenarán en un arreglo llamado *state* de 4 por 4 bytes. Este arreglo va cambiando en cada ronda, y finalmente su contenido tendrá el texto cifrado. La llave es expandida en 11 arreglos de llaves del mismo tamaño que el arreglo *state* y cada uno de éstos representa una llave de ronda, cuyo valor es un proceso complicado de rotaciones repetidas con OR exclusivo.

  Este algoritmo además de proveer una gran seguridad trabaja a una gran velocidad y es la mejor opción de encripción convencional en la actualidad.

- *IDEA:* Este algoritmo se ha considerado como una buena propuesta para reemplazar a DES debido a su forma de trabajo ya que se basa en dos operaciones sencillas de adición modular y XOR, y una muy compleja de multiplicación modular la cual le da mayor fortaleza a IDEA. Este algoritmo es usado en PGP para seguridad de correo electrónico.

- *Blowfish:* La ventaja de este algoritmo es su rápida ejecución y necesita de muy pocos recursos de memoria debido al uso de dos operaciones primitivas para su encripción.
- *RC5:* Entre las características destacadas de este algoritmo es que trabaja con número de rondas variable y longitud de llave variable las mismas que dependen de los datos de entrada. Estas particularidades representan una ventaja ya que dificultan el criptoanálisis diferencial que consiste en determinar qué modificación produce en el texto cifrado las modificaciones en el texto plano.
- *RC2:* Al igual que RC5 trabaja también con longitud de llave variable y es utilizado para la seguridad de correo electrónico en S/MIME.

En la actualidad el algoritmo más robusto y recomendado es AES que en Estados Unidos es el algoritmo oficial de encripción convencional a utilizarse en cualquier institución pública y privada de su nación, pero debido a sus políticas restrictivas de exportación no es asequible para la mayoría de países, sobre todo para los países en desarrollo, por esta razón la segunda opción recomendada a utilizar es el algoritmo Triple DES.

Otro punto importante a destacar en la encripción convencional es la manera en que será distribuida la llave compartida por las dos partes que van a intercambiar la información para lo cual puede hacérselo manualmente entre ambas o mediante un centro de distribución de llaves KDC (*Key Distribution Center*).

1.7.1.3 Encripción con llave pública

En el caso de la encripción convencional el principal problema es la distribución segura de las llaves, por lo que se propuso esta forma de encripción también conocida como encripción asimétrica, ya que se usa una llave para encripción y otra distinta para la desencripción. Dentro de los algoritmos más destacados están RSA (*The Rivest-Shamir-Adleman*) y ECC (*Elliptic Curve Cryptography*), los cuales no son clave de estudio de este proyecto por tanto no serán detallados. A continuación solo se presentará una breve definición de cómo trabaja la encripción con llave pública y los servicios que provee.

Dentro de los servicios que presta este tipo de encripción están: sólo la protección a la confidencialidad, sólo la protección a la autenticación o la protección a ambas propiedades. Si sólo se quiere garantizar confidencialidad con encripción con llave pública, en la figura 1.62 (a) se indica el procedimiento correspondiente. A encripta un mensaje M con una llave pública KUb y lo envía a B, quien publicó dicha llave (en una página Web por ejemplo) y B es la única que puede desencriptar dicho mensaje ya que es la única que contiene la correspondiente llave privada KRb, pero cualquier persona puede emitir un mensaje a B personificando a A y B no puede demostrarlo.

Si solo se quiere garantizar autenticación el procedimiento se muestra en la figura 1.62 (b); la diferencia con el procedimiento anterior es que el emisor A encripta con su llave privada mientras que B o cualquier otra persona puede desencriptar con la respectiva llave pública de A.

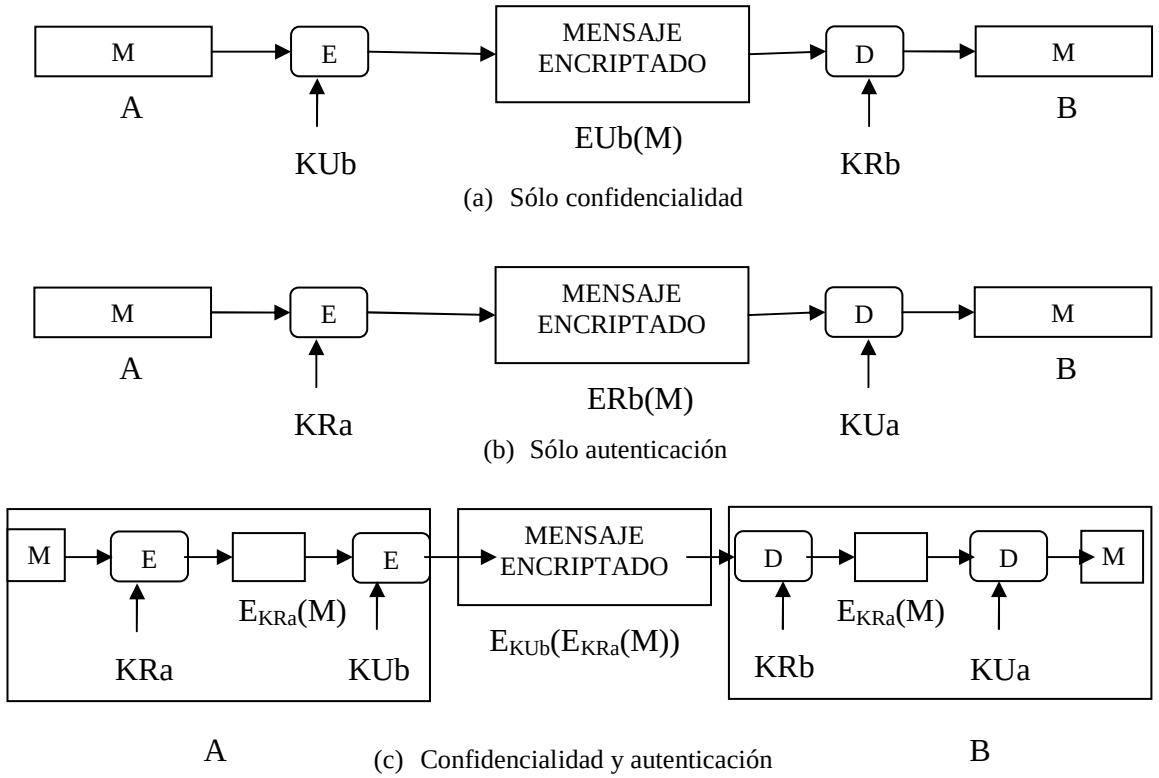


Figura 1. 62: Encripción con llave pública: Confidencialidad[8]

Para garantizar autenticación y confidencialidad se aplica una combinación de los dos procedimientos anteriores como se puede observar en la figura 1.62 (c).

Los últimos dos esquemas corresponden a la *autenticación de mensajes* en los que se protege a las dos partes A y B, que están intercambiando información, de terceras partes no autorizadas que quieran personalizar a una de las dos.

1.7.2 APLICACIONES DE LA CRIPTOGRAFÍA

1.7.2.1 Firma digital

Así como la firma autógrafa, la firma digital garantiza la correspondencia entre una persona y la información que ésta ha emitido en un documento o en un mensaje.

A diferencia de la autenticación de mensajes este mecanismo es utilizado para proteger a ambas partes que desean comunicarse entre ellas, es decir que quien recibe el mensaje pueda verificar que el transmisor es quien dice ser, que el transmisor no pueda negar el haber emitido un mensaje que si lo emitió y que el receptor no se envíe un mensaje a el mismo personificando a otra persona.

1.7.2.2 Certificados Digitales

Un certificado digital corresponde a un fichero que garantiza la asociación entre una persona y su respectiva clave pública; este documento digital debe ser emitido por una autoridad certificadora que a su vez debe ser reconocida como tal por otra institución encargada de la acreditación de estas entidades de certificación. Los certificados digitales contienen la información de la autoridad certificadora y del titular de dicho certificado, la clave pública del titular del certificado, fecha de validez, número de serie e identificación del emisor del certificado (firma digital).

1.7.2.3 Redes Privadas Virtuales

Las Redes Privadas Virtuales mejor conocidas como VPNs por sus siglas en inglés (*Virtual Private Networks*), permiten establecer una conexión segura como si se estuviera utilizando la estructura de la red LAN, pero la información viaja sobre una red pública. La información es encriptada y encapsulada para garantizar las propiedades fundamentales de la misma. Los algoritmos de encripción generalmente utilizados en VPNs son DES y 3DES.

1.7.2.4 Seguridad Informática en el Sector Bancario

La seguridad Informática dentro de este sector consiste en proteger la información de las personas (clientes, administradores y propietarios) que forman parte de una institución bancaria o afín ante posibles amenazas como negación del servicio, ingeniería social, *spyware* entre otras que atentan contra las propiedades fundamentales de la información. Por este motivo es importante aplicar una serie de mecanismos que permitan alcanzar los niveles de seguridad requeridos entre los cuales están: adoptar un estándar de seguridad internacional para sistemas informáticos, documentar todas las actividades operacionales relacionadas la seguridad de la información, utilizar mecanismos robustos de control de acceso, monitorear las transacciones entre otros mecanismos que dependerán de las políticas de la institución.

1.7.2.5 Comercio Electrónico

El comercio electrónico es una aplicación que permite la venta o compra de productos a través de medios electrónicos como por ejemplo Internet que es el más difundido, por ende las partes involucradas en la transacción deben garantizar su identidad y es necesario establecer mecanismos seguros de autenticación y de cifrado lo cual se consigue con la encripción de llave pública y la encripción de llave privada respectivamente.

CAPÍTULO 2

DISEÑO, DESARROLLO E IMPLEMENTACIÓN DEL SISTEMA

2.1 DISEÑO DE LA APLICACIÓN

En esta sección se diseñará la aplicación en base a los conceptos de la programación orientada a objetos descrita en el subcapítulo 1.4; el modelo de diseño y de desarrollo usa como lenguaje de modelado UML versión 2.

2.1.1 DESCRIPCIÓN DEL PROBLEMA

- **Título de la Aplicación:**

ATM DACTILAR: Interfaz para autenticar usuarios de un cajero automático mediante el uso de huellas dactilares.

- **Documento de Análisis:**

Se requiere un software que permita autenticar a una persona mediante la verificación de su huella dactilar, esta persona en adelante será denominada cliente o usuario indistintamente. El sistema obtendrá un número de identificación del usuario en el momento que éste ingrese la tarjeta al cajero; después se solicitará la huella dactilar que junto al número de identificación del usuario (número de tarjeta) serán transmitidas a un servidor que procederá a verificar a la persona, autenticándola en el sistema y dándole paso a otra aplicación (que ya está en producción) que le permitirá al cliente hacer transacciones con el cajero. En caso que la persona sea autenticada afirmativamente, el servidor enviará un mensaje de aceptación al cajero para que el usuario pueda realizar la transacción deseada. En caso que la persona sea rechazada en la autenticación, el servidor enviará un mensaje de negación de autenticidad al cajero y junto a ese mensaje también se le indicará qué dedo es que el que está registrado en el sistema, dándole la oportunidad al cliente de nuevamente de iniciar el proceso de autenticación.

La comunicación que se da entre el host (cajero) y el servidor estará basada en el protocolo TCP/IP por lo que se debe poder especificar la dirección IP del servidor así como el puerto en el que dicho servidor está aceptando las transacciones; adicionalmente aunque se emplee un túnel VPN previamente establecido entre el host y el servidor, el mensaje debe ser encriptado con una llave conocida por los dos extremos y especificada como parámetro de la aplicación.

El servidor y el cajero deben llevar registros para monitorear qué personas han intentado usar la aplicación, registrando el número de identificación, así como debe también registrar el puntaje de verificación, si el usuario fue o no aceptado y la dirección IP del cajero que envió el mensaje (solo para el servidor).

Para determinar si un cliente desea autenticarse en un cajero automático, la aplicación leerá periódicamente un archivo que contiene un *timestamp* (fecha y hora) seguido del número de identificación del cliente que busca la autenticación. La aplicación externa ya en producción en el cajero es la encargada de cambiar la información de este archivo una vez que el cliente ha ingresado su tarjeta; si se ha producido este cambio y además el *timestamp* coincide con la hora local del equipo, se solicitará la huella al cliente para la respectiva autenticación. El resultado de la autenticación es almacenado en otro archivo que será leído por la aplicación externa la cual le permitirá al usuario seleccionar la transacción a realizar en caso de haber sido autenticado, caso contrario pasará a un estado de error si falló la autenticación.

Para realizar el enrolamiento de los clientes y sus huellas dactilares en el sistema, existirán personas (debidamente registradas) que a través de una interfaz gráfica ingresarán el número de identificación del usuario, el lenguaje de preferencia del mismo, la cooperativa a la que pertenece dicho usuario, el dedo que ha sido registrado, así como el número de identificación de la persona que está enrolando al cliente. En este proceso también se puede dar la actualización de cualquier dato del cliente.

El sistema debe estar en la posibilidad, de manejar temporizadores de espera para cada una de las etapas de verificación (temporizador de espera de la huella dactilar después de ingresada la tarjeta, temporizador de espera para recibir la confirmación de autenticidad). Del mismo modo la aplicación debe estar en la

capacidad de generar reportes de estado e históricos de uso; estos reportes son: número de clientes aceptados o no por la aplicación filtrado por fecha, número de identificación de los clientes que han sido rechazados por el sistema filtrado por fecha, identificación de los cajeros que se han conectado al servidor y que han realizado alguna transacción, y número de clientes nuevos que han sido enrolados filtrado por fecha.

2.1.2 RECOLECCIÓN DE REQUERIMIENTOS

R.1. Verificar la autenticidad de un usuario mediante su número de identificación y su huella dactilar.

R.2. La comunicación entre el cajero y el servidor debe emplear los protocolos IP y TCP.

R.3. Los mensajes que se transmiten entre el cajero y el servidor deben estar encriptados con una llave conocida por los dos.

R.4. El servidor y el cajero deben manejar un registro de monitorización de uso de la aplicación.

R.4.1. El servidor registra el número de identificación del usuario, el puntaje de verificación, si el usuario fue o no aceptado y la dirección IP del cajero que envió el mensaje.

R.4.2. El cajero registra el número de identificación del usuario, el puntaje de verificación y si el usuario fue o no aceptado.

R.5. Solo personas debidamente autorizadas pueden enrolar clientes nuevos o actualizar datos de un cliente en el sistema a través de una interfaz gráfica.

R.6. La información que se debe ingresar por cada cliente es: número de identificación, nombre, lenguaje de preferencia, cooperativa a la que pertenece, dedo que ha sido registrado e identificación de la persona que enroló al cliente.

R.7. La aplicación manejará temporizadores de espera para cada una de las transacciones y estado del cajero. Estos temporizadores son:

R.7.1. Temporizador de espera de la huella dactilar después de ingresada la tarjeta.

R.7.2. Temporizador de espera para recibir la confirmación de autenticidad.

R.8. El software permitirá generar reportes del estado de la aplicación y de históricos de uso. Estos reportes son:

R.8.1. Número de clientes aceptados o no por la aplicación, filtrado por fecha.

R.8.2. Número de identificación de los clientes que han sido rechazados por el sistema filtrado por fecha.

R.8.3. Identificación de los cajeros que se han conectado al servidor y que han realizado alguna transacción

R.8.4. Número de clientes nuevos que han sido enrolados, filtrados por fecha.

2.1.3 DESCRIPCIÓN DE ESCENARIOS Y SELECCIÓN DE ACTORES

A continuación se describirán los escenarios presentes en este problema.

- **Numeración:** 1

  Título: Autenticar personas para acceso administrativo.

  Prerrequisitos: Huella dactilar de la persona.

  Quien lo comienza: Persona en busca de autorización administrativa.

  Quien lo termina: Aplicación.

  Post condiciones: Autenticación aceptada → Persona autorizada asignada a su respectivo rol. Caso contrario Persona no autorizada.

  Descripción: Son todos los procedimientos necesarios para extraer la huella dactilar de una persona que desea acceder al sistema para interactuar con él. Una vez extraída la huella dactilar la aplicación procederá a buscar la identidad de la persona (identificación). Si la persona identificada cumple con las credenciales (permisos) necesarias, la interfaz

gráfica se iniciará en el modo respectivo y con el perfil adecuado, caso contrario mostrará el correspondiente mensaje de error.

- **Numeración:** 2

  Título: Enrolar Clientes.

  Prerrequisitos: Interfaz de enrolamiento activa, Escenario 1.

  Quien lo comienza: Persona autorizada para enrolamiento.

  Quien lo termina: Persona autorizada para enrolamiento.

  Post condiciones: Cambio del registro de enrolamiento.

  Descripción: Se ingresa todos los datos del cliente al sistema por parte de la persona encargada del enrolamiento. En caso de que el cliente ya haya sido registrado previamente, se producirá un mensaje de error.

- **Numeración:** 3

  Título: Actualizar Clientes.

  Prerrequisitos: Cliente ingresado en el sistema, Número de identificación del cliente disponible, Escenario 1.

  Quien lo comienza: Persona autorizada para actualizar clientes.

  Quien lo termina: Persona autorizada para actualizar clientes.

  Post condiciones: Cambio de la información del cliente.

  Descripción: La persona que desea actualizar los datos del cliente, primero debe ser autenticada por el sistema, para lo cual tiene que registrarse. Posterior a eso la aplicación solicita al servidor de base de datos toda la información del cliente usando el número de identificación del mismo. Los datos son cargados en la interfaz gráfica para que la persona autorizada actualice la información del cliente. Los cambios son almacenados en el servidor de base de datos.

- **Numeración:** 4

  Título: Verificar Clientes.

Prerrequisitos: Cliente ingresado en el sistema, Número de identificación del cliente disponible.

Quien lo comienza: Aplicación externa

Quien lo termina: Aplicación.

Post condiciones: Archivos con información del cliente serán modificados.

Descripción: La aplicación empaqueta la huella dactilar extraída del cliente, y procede a armar un paquete con la información necesaria para ser transmitido al servidor y verificar la autenticidad del cliente. En caso que la autenticidad sea comprobada como positiva el servidor envía un paquete que se escribe en un archivo en el cajero, el cual le indica que la verificación ha sido positiva y le permite al cliente realizar la transacción deseada. En caso que la autenticidad sea comprobada como negativa el servidor envía un paquete que se escribe en un archivo en el cual se le indica al cajero que el cliente ha sido rechazado.

- **Numeración:** 5

  Título: Administrar el sistema.

  Prerrequisitos: Escenario 1.

  Quien lo comienza: Administrador.

  Quien lo termina: Administrador.

  Post condiciones: Ninguna.

  Descripción: El administrador debidamente identificado y autorizado puede realizar cambios en los parámetros de la aplicación, del mismo modo puede revisar los registros de la aplicación y generar reportes.

A partir de los escenarios previamente descritos se pueden seleccionar los siguientes actores:

- Persona en busca de autorización administrativa.
- Persona autorizada para enrolar clientes.
- Aplicación externa que controla el cajero.

- Persona autorizada para actualizar clientes.
- Cliente o usuario del cajero.
- Administrador.

2.1.4 DETERMINACIÓN DE CASOS DE USO

Una vez descritos los escenarios y seleccionados los actores del sistema el siguiente paso será plantear los casos de uso. En esta sección se planteará el diagrama de uso general de la aplicación así como los diagramas de uso de cada uno de los escenarios, acompañados de una descripción y la referencia del escenario que representan. Para realizar los diferentes diagramas de UML se utilizará como herramienta principal de diagramación el programa Microsoft Visio.

2.1.4.1 Caso de Uso 1: Diagrama General

- *Escenario:* ninguno.
- *Descripción:* En este caso de uso se plantean los límites del sistema y se recopilan todos los casos de uso a diagramarse.

2.1.4.2 Caso de Uso 2: Autenticar Personas para Acceso Administrativo

- *Escenario:* 1.
- *Descripción:* Una persona que desea realizar una tarea administrativa (enrolar clientes, actualizar clientes y/o administrar el sistema) debe autenticarse en el sistema mediante su huella dactilar. El sistema intenta identificarla consultando en la base de datos, si el usuario se encuentra registrado (ingresado en el sistema) se buscan los permisos asignados y se le comunica a la aplicación del éxito en el ingreso y los permisos; caso contrario, el sistema emite un mensaje de error.

2.1.4.3 Caso de Uso 3: Enrolar Clientes

- *Escenario:* 2.
- *Descripción:* Para que un usuario pueda acceder a este caso de uso, previamente se debe autenticar según el caso de uso 2. La persona autorizada para el enrolamiento solicita todos los datos al cliente, captura la

huella dactilar del mismo e ingresa la información al sistema. Para guardar la información, la aplicación verifica y valida el formato de la misma. Después de la verificación, se establece una conexión con el servidor de base de datos para almacenar la información, si ésta se almacena correctamente, se indicará el éxito de la operación a través del despliegue de un mensaje, caso contrario, se mostrará el correspondiente mensaje de error indicando la causa. Se puede producir un error cuando el cliente ya ha sido ingresado o si la información no tiene el formato adecuado. La información que se almacena por cada cliente es la especificada en el requerimiento 6 (Sección 2.1.2 – R.6).

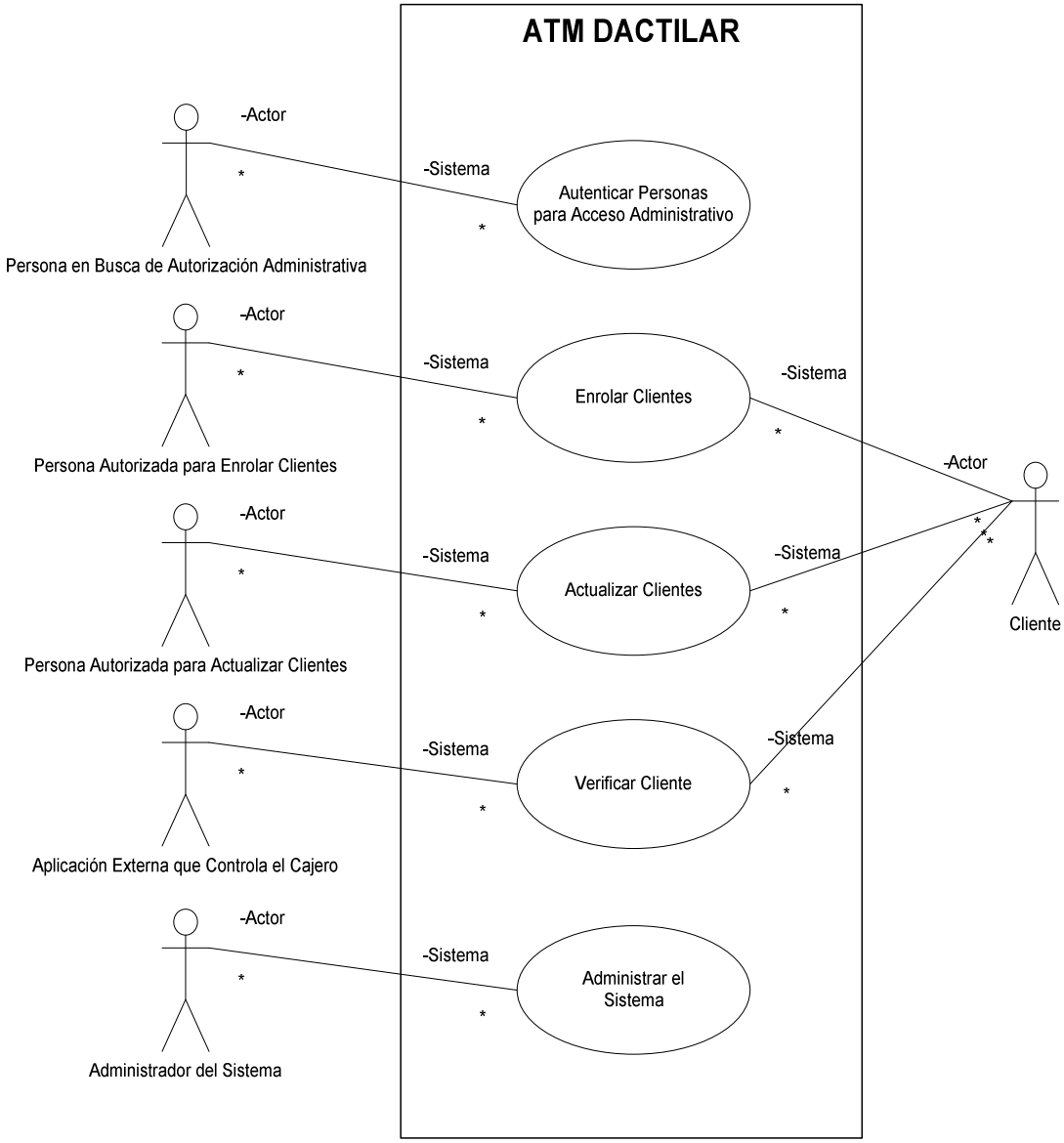


Figura 2. 1: Caso de Uso 1 (Diagrama General)

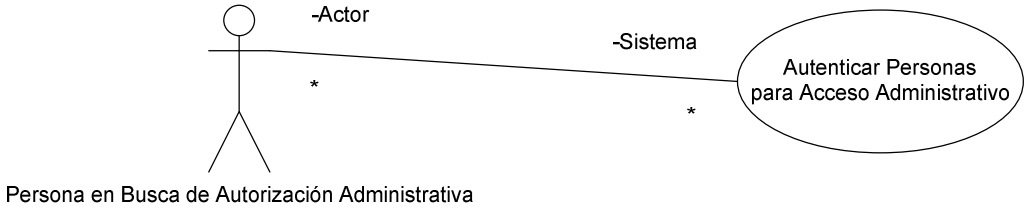


Figura 2. 2: Caso de Uso 2 (Autenticar Personas para Acceso Administrativo)

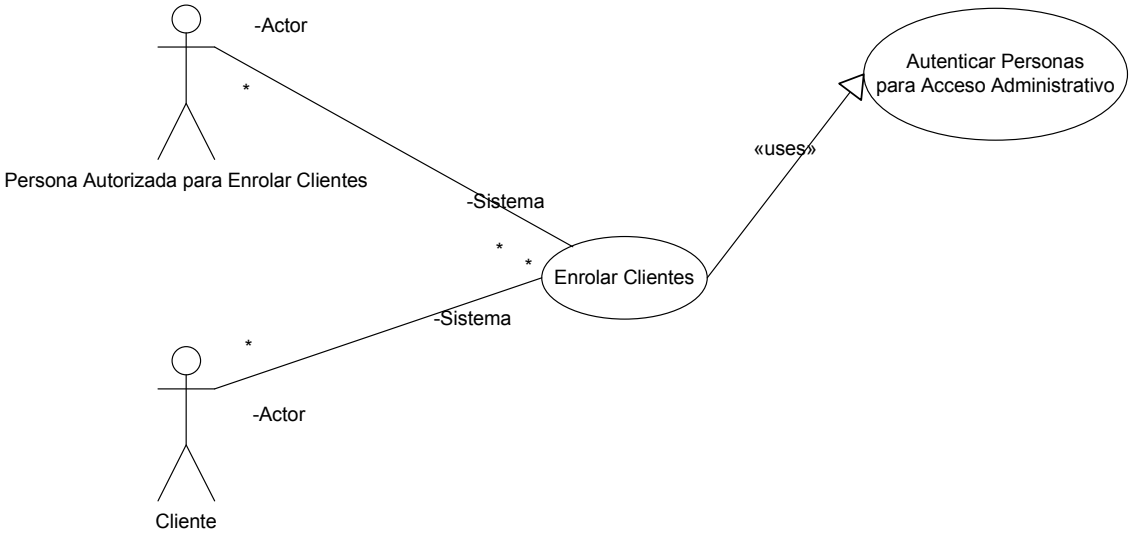


Figura 2. 3: Caso de Uso 3 (Enrolar Clientes)

2.1.4.4 Caso de Uso 4: Actualizar Clientes

- *Escenario:* 3.
- *Descripción:* Para que un usuario pueda acceder a este caso de uso, previamente se debe autenticar según el Caso de Uso 2. Este caso de uso extiende el Caso de Uso 3, salvo el hecho que no se produce un mensaje de error cuando el cliente ya se encuentra registrado, lo que permite es actualizar la información del cliente.

2.1.4.5 Caso de Uso 5: Verificar Clientes

- *Escenario:* 4.
- *Descripción:* La aplicación externa requiere la autenticación de un cliente y escribe en un archivo la fecha, hora actual y el número de identificación del cliente; si el sistema detecta el cambio de hora y/o fecha se procede a

extraer la información de dicho archivo y solicita la huella del cliente. El cliente coloca su dedo (el que ha sido registrado en el enrolamiento) sobre el lector y se extrae su huella dactilar; la aplicación arma un mensaje que contiene el número de identificación y la huella dactilar, lo encripta y lo transmite al servidor quien valida la información. Después de validar la información, el servidor devuelve un mensaje también encriptado que contiene la aceptación o el rechazo del cliente.

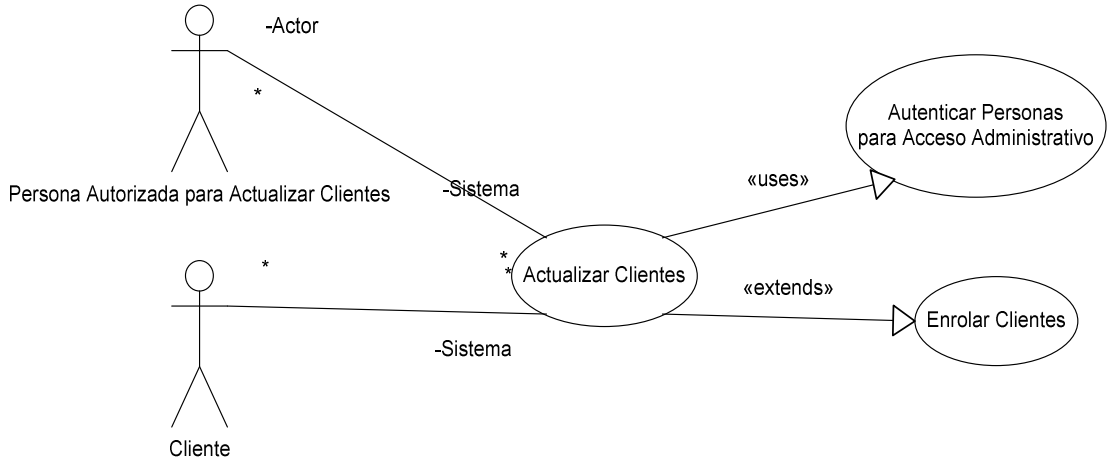


Figura 2. 4: Caso de Uso 4 (Actualizar Clientes)

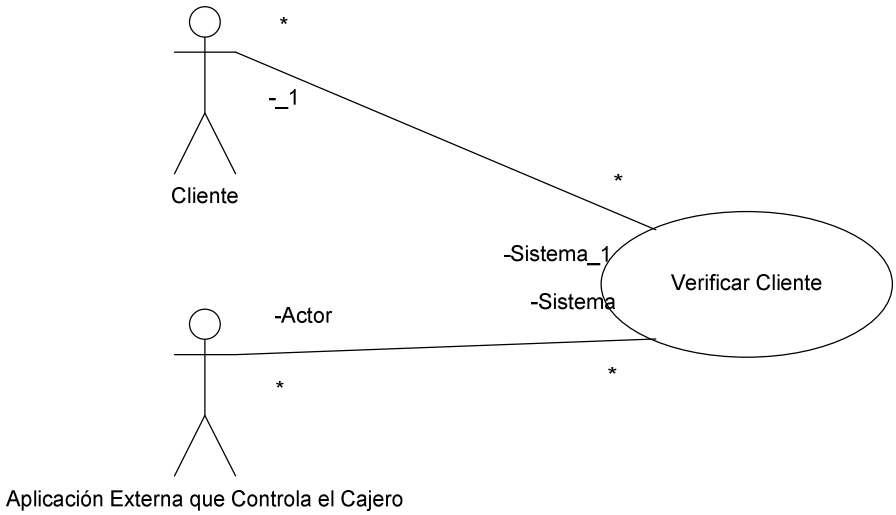


Figura 2. 5: Caso de Uso 5 (Verificar Clientes)

2.1.4.6 Caso de Uso 6: Administrar el Sistema

- *Escenario:* 5.

Descripción: Para que un usuario pueda acceder a este caso de uso, previamente se debe autenticar según el caso de uso 2. El administrador debidamente verificado puede cambiar parámetros de la aplicación, como son los relacionados a la conexión con el servidor de base de datos. El administrador también puede revisar los reportes indicados en el requisito número 8.

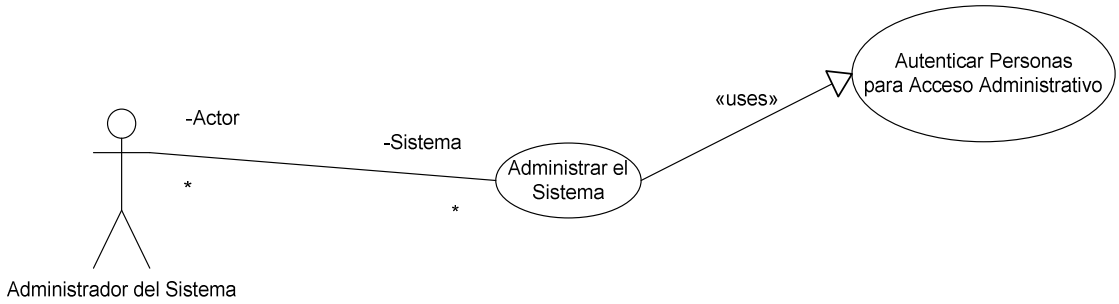


Figura 2. 6: Caso de Uso 6 (Administrar el Sistema)

2.1.5 IDENTIFICACIÓN DE OBJETOS

Del análisis de los requerimientos, escenarios y casos de uso planteados en los subcapítulos 2.1.2, 2.1.3 y 2.1.4, empleando como método de descripción fichas CRC, se pueden extraer los siguientes objetos básicos:

Clase: Usuario Administrativo.	
Atributos: Nombre del Usuario, Número de Identificación, Huella Dactilar, Lista de Roles.	
Responsabilidades: Autenticar Usuarios, Proveer el o los roles asignados a un usuario, Proveer la huella a ser identificada.	**Colaboradores:** Huella.

Figura 2. 7: Ficha CRC - Clase Usuario Administrativo

Clase: Cliente.	
Atributos: Nombre del Usuario, Número de Identificación, Lenguaje de Preferencia, Cooperativa a la que Pertenece, Huella Dactilar, Número de Identificación de la persona que enrolo al cliente.	
Responsabilidades: Representación de un Cliente (actor), Proveer la huella a ser verificada.	**Colaboradores:** Huella.

Figura 2. 8: Ficha CRC - Clase Cliente

Clase: Huella.	
Atributos: Plantilla, Tamaño, Dedo.	
Responsabilidades: Proveer la plantilla de una huella dactilar.	**Colaboradores:**

Figura 2. 9: Ficha CRC - Clase Huella

Clase: TxRX (Transmitir Recibir).	
Atributos: Dirección IP, Puerto, Codificación, Protocolo.	
Responsabilidades: Proveer todos los métodos para la comunicación entre dos objetos a nivel de red.	**Colaboradores:**

Figura 2. 10: Ficha CRC - Clase TxRx (Transmitir Recibir)

Clase: Servidor.	
Atributos: TxRx, BDD, Encripción, Lista de Huellas, Lista de Usuarios.	
Responsabilidades: Autenticar clientes, esperar mensajes para procesarlos, extraer y almacenar información en la base de datos.	**Colaboradores:** Huella, BDD, Cliente, TxRx, Encripción.

Figura 2. 11: Ficha CRC - Clase Servidor

Clase: Host Cajero.	
Atributos: TxRx, Encripción, Huella, Cliente.	
Responsabilidades: Extraer la huella del cliente, encriptar mensajes, enviar información mediante mensajes al servidor.	**Colaboradores:** Huella, Cliente, TxRx, Encripción.

Figura 2. 12: Ficha CRC - Clase Host Cajero

Clase: Host Administrativo.	
Atributos: BDD, Huella, Usuario Administrativo.	
Responsabilidades: Autenticar Usuarios Administrativos y Clientes, extraer la huella del usuario administrativo, cambiar parámetros de la aplicación, generar reportes.	**Colaboradores:** Huella, Usuario Administrativo, BDD.

Figura 2. 13: Ficha CRC - Clase Host Administrativo

Clase: BDD.	
Atributos: Nombre de Base de Datos, Nombre de Servidor de BDD, Usuario DBB, Password, Conexión, Estado de conexión.	
Responsabilidades: Enlazar el Servidor o Host Administrativo con la BDD.	**Colaboradores:**

Figura 2. 14: Ficha CRC - Clase BDD (Base de Datos)

Clase: Encripción.	
Atributos: Mensaje, Clave.	
Responsabilidades: Proveer todos los métodos para la encripción y desencripción entre dos objetos.	**Colaboradores:**

Figura 2. 15: Ficha CRC - Encripción

2.1.6 DIAGRAMAS DE SECUENCIA

Una vez definidos los objetos, se puede determinar cómo se comunican los mismos mediante los siguientes diagramas de secuencias:

2.1.6.1 Diagrama de secuencia 1: Autenticar Personas (Usuario Administrativo)

- Escenario: 1.
- Caso de Uso: 2.
- Descripción: Si un *Usuario Administrativo* desea enrolar clientes, actualizar clientes o administrar el sistema, es necesario que sea autenticado, para lo cual su huella debe ser capturada por el lector como se ilustra en el diagrama de la figura 2.16; una vez generado este evento, el *Host Administrativo* crea el objeto *Huella* y con la imagen capturada extrae la huella del usuario a autenticar. Este objeto notifica el resultado de la extracción al *Host Administrativo*, el mismo que crea un objeto *Base de Datos* para establecer la conexión con el servidor de Base de Datos y llama al procedimiento almacenado para consultar todas las huellas de Usuarios Administrativos de la Base de Datos; el objeto *Base de Datos* retorna el resultado de la consulta al objeto *Host Administrativo,* el mismo que

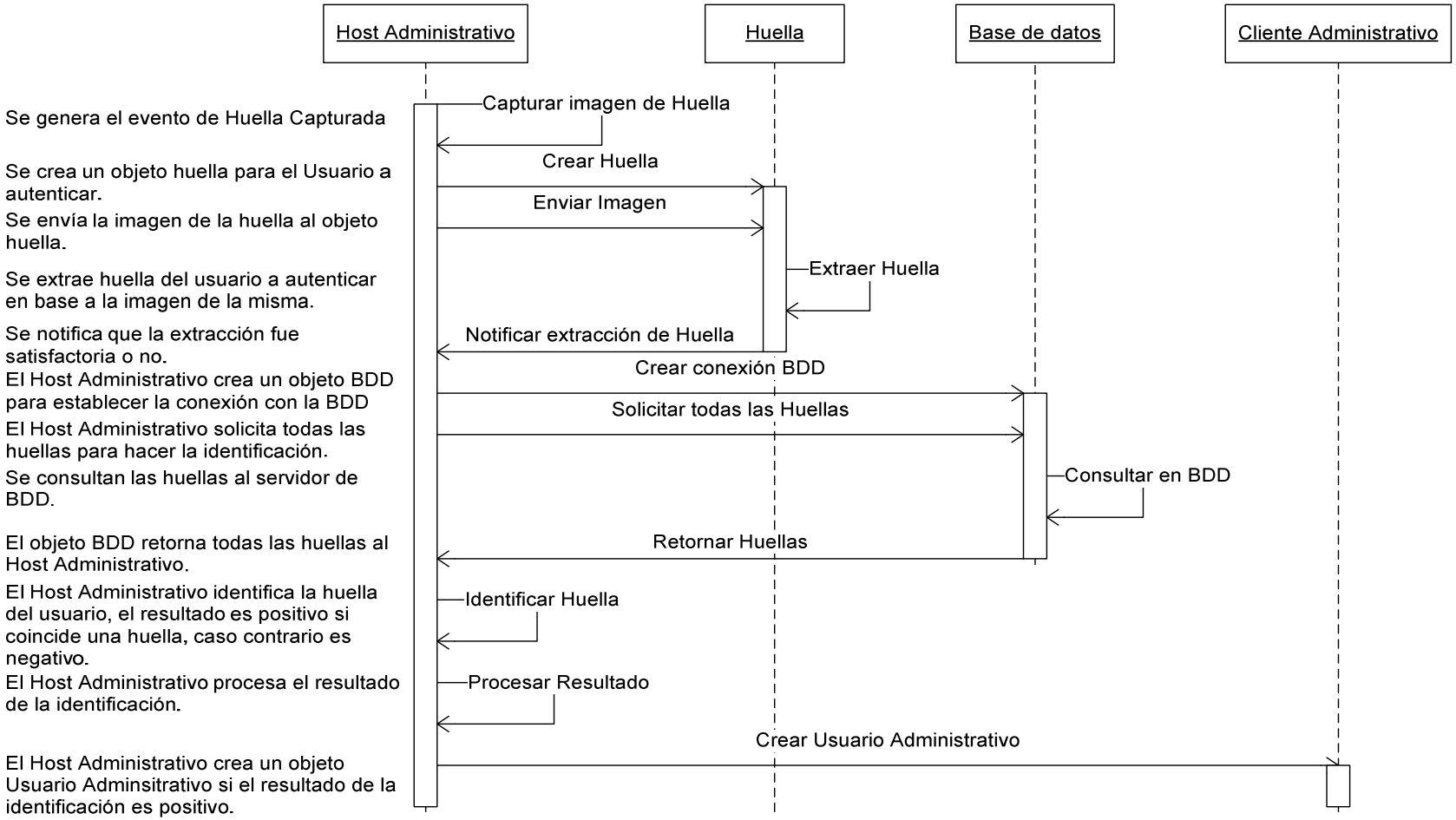


Figura 2. 16: Diagrama de Secuencia 1 - Autenticar Personas (Usuarios Administrativos)

identifica al Usuario Administrativo y si el resultado de la identificación es positivo crea un nuevo objeto *Usuario Administrativo.*

2.1.6.2 Diagrama de secuencia 2: Enrolar Clientes

- Escenario: 2.
- Caso de Uso: 3.

  Descripción: Para que un *Usuario Administrativo* pueda enrolar clientes el objeto *Host Administrativo,* debe primeramente verificar que está autorizado para hacerlo como se puede ver en el diagrama de la figura 2.17; con ello se genera el evento *cargar la interfaz* para enrolamiento si el usuario efectivamente está autorizado. A continuación el *Host Administrativo* crea un objeto *Cliente* donde se carga toda la información correspondiente al usuario a enrolar; después de un proceso de verificación del formato de la información según su tipo de datos y formato, el *Host Administrativo* se encarga de establecer la conexión con el servidor de Base de Datos y solicitar el almacenamiento de la información mediante el respectivo procedimiento almacenado. El objeto *Base de Datos* envía el resultado del almacenamiento (positivo o negativo) al *Host Administrativo*, el mismo que lo procesa y despliega la información correspondiente al resultado.

2.1.6.3 Diagrama de secuencia 3: Actualizar Clientes

- Escenario: 3.
- Caso de Uso: 4.

  Descripción: Para que un *Usuario Administrativo* pueda actualizar clientes, el objeto *Host Administrativo* debe primeramente verificar que está autorizado para hacerlo, con lo cual se genera el evento de *cargar la interfaz* para actualización si el usuario efectivamente está autorizado como se muestra en el diagrama de la figura 2.18. A continuación el *Host Administrativo* crea un objeto *Huella* para extraer la huella del cliente a actualizar en base a la imagen de la misma.

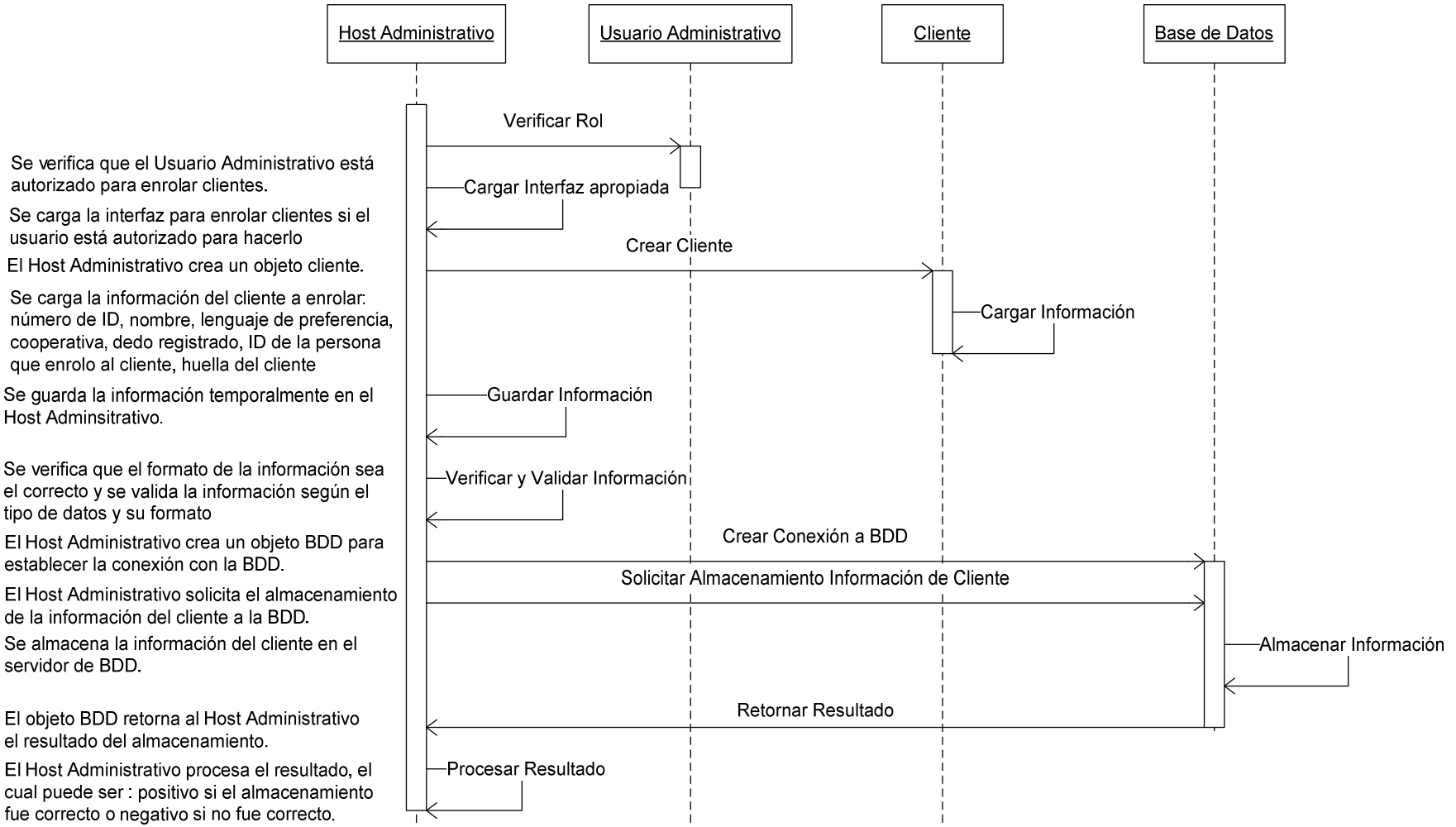


Figura 2. 17: Diagrama de Secuencia 2 - Enrolar Clientes

El *Host Administrativo* crea un objeto *Base de Datos* el cual establece la conexión con el servidor de Base de Datos y solicita las huellas correspondientes al cliente que desea verificar en base a su identificador, mediante la llamada al correspondiente procedimiento almacenado. El *Host Administrativo* procede a verificar que el cliente se encuentra registrado, si el resultado es positivo crea un objeto *Cliente* para cargar la información a actualizar. El *Host Administrativo* crea el objeto *Base de Datos,* el mismo que se encarga de llamar al procedimiento almacenado que permite la actualización de la información del cliente y retorna el resultado al *Host Administrativo*, éste lo procesa y despliega el resultado correspondiente.

2.1.6.4 Diagrama de secuencia 4: Verificar Clientes

- Escenario: 4.
- Caso de Uso: 5.

Descripción: Si la aplicación externa requiere la verificación del cliente, el objeto *Host Cajero* debe determinarlo mediante la lectura de un archivo que contiene la fecha y hora de la petición de autenticación y el número de identificación del cliente a verificar; posteriormente captura la imagen de la huella del cliente y el Host Cajero crea un objeto *Huella*, encargado del procesamiento de la imagen capturada como se indica en el diagrama de la figura 2.19. Un mensaje encriptado con la información de la huella y del identificador del cliente es transmitido al *Servidor,* quien crea un objeto *Base de Datos* el cual establece la conexión con el servidor de Base de Datos y llama al procedimiento almacenado que permite consultar las huellas correspondientes al cliente que se desea verificar; el resultado de la verificación se envía al objeto *Servidor* y éste lo transmite en un mensaje encriptado al objeto *Host Cajero* que en función del resultado desplegará la información correspondiente.

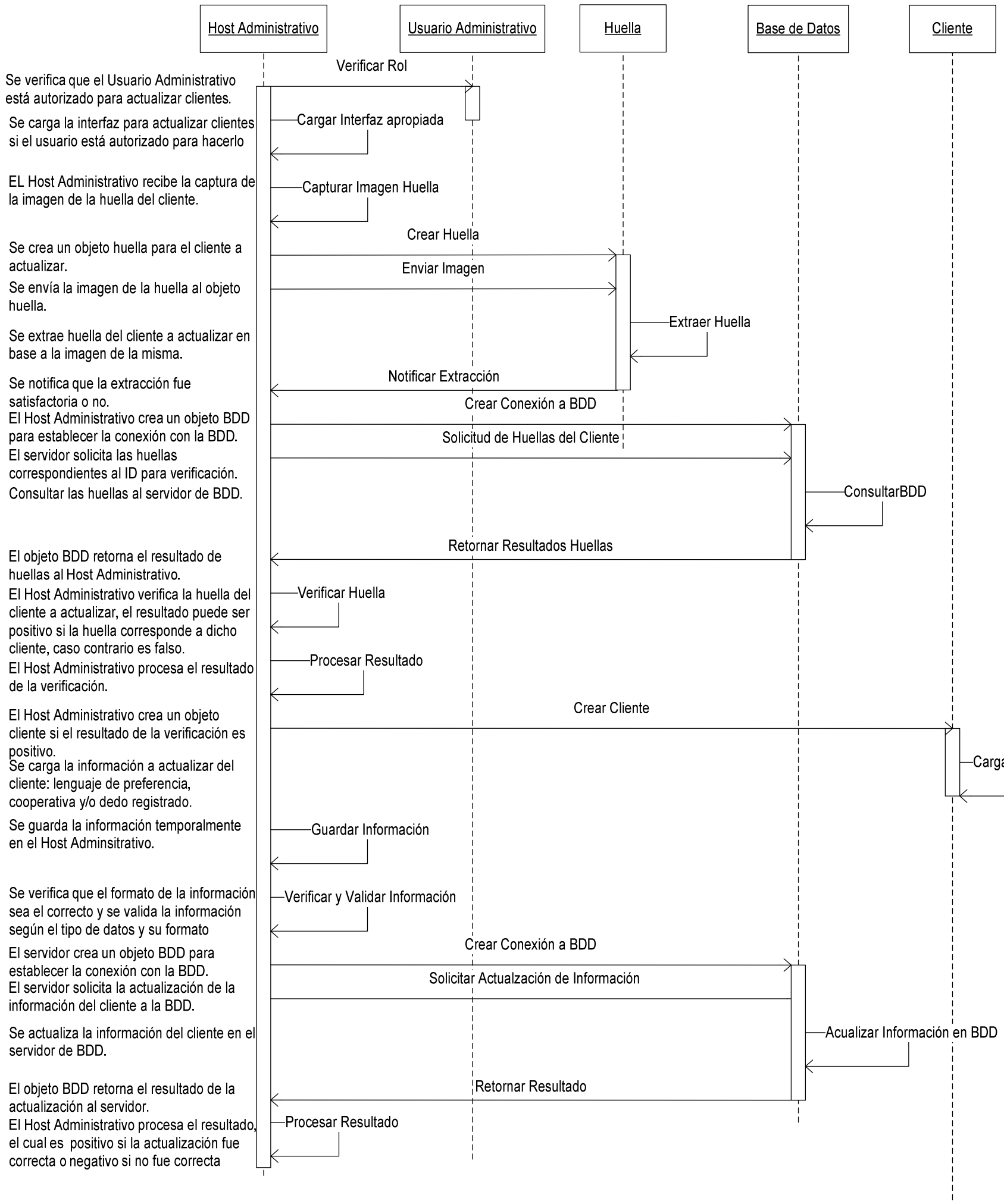


Figura 2. 18: Diagrama de Secuencia 3 - Actualizar Clientes

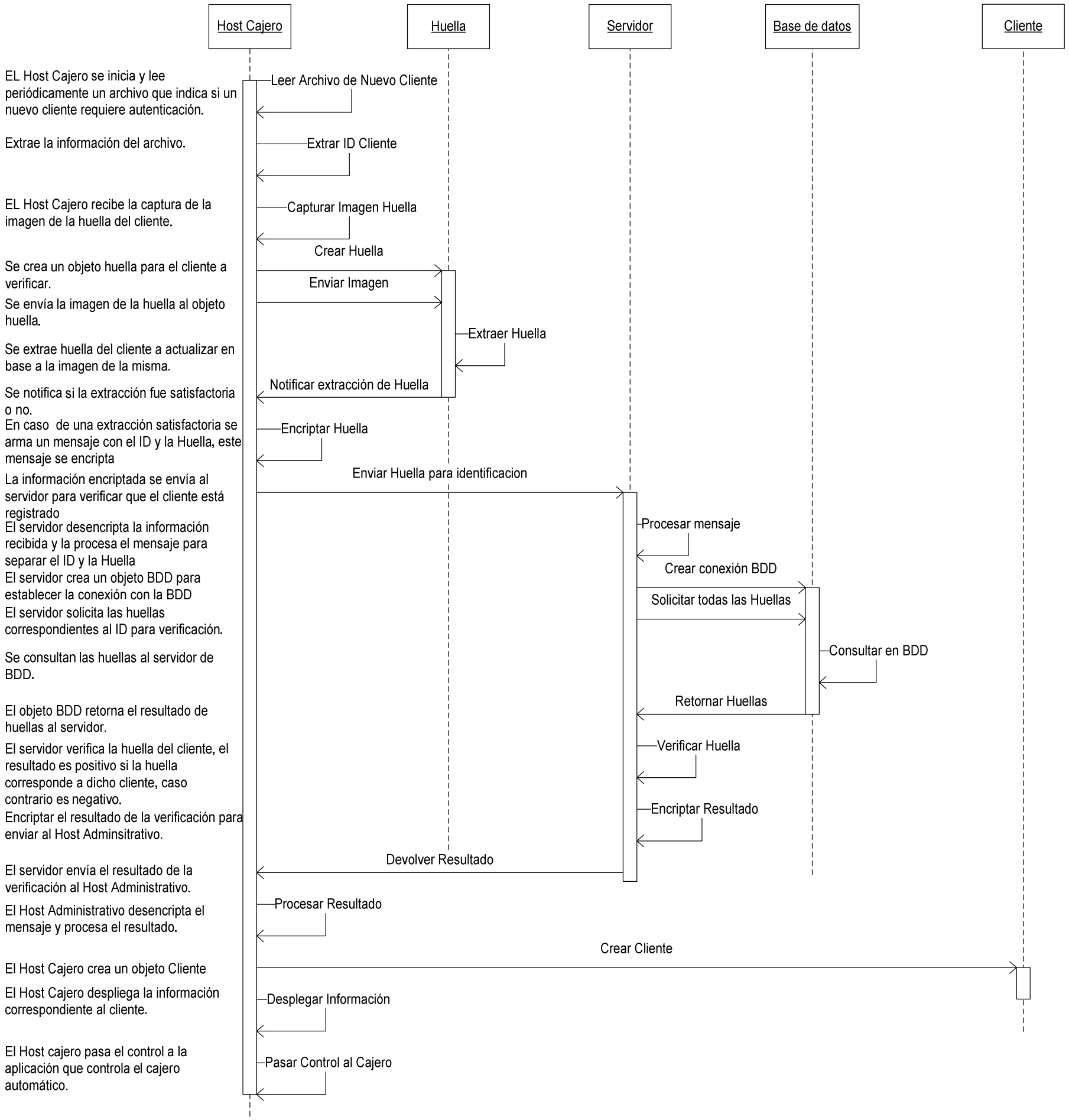


Figura 2. 19: Diagrama de Secuencia 4 - Verificar Clientes

2.1.6.5 Diagrama de secuencia 5: Administrar Sistema

- Escenario: 5.
- Caso de Uso: 6.
- Descripción: Si un Usuario Administrativo desea administrar el sistema, el objeto *Host Administrativo* debe verificar el correspondiente rol, como se encuentra ilustrado en el diagrama de la figura 2.20. Una vez verificado el rol se genera el evento de cargar la interfaz apropiada para administrar el sistema y el Host Administrativo podrá realizar la operación deseada.

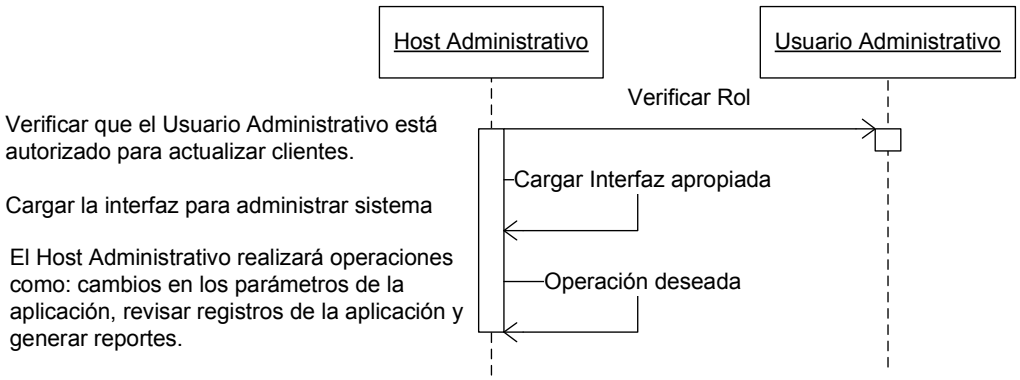


Figura 2. 20: Diagrama de Secuencia 5 - Administrar el sistema

2.1.7 DIAGRAMAS DE COLABORACIÓN

Los siguientes diagramas ilustran el orden en que se presentan los mensajes, eventos y operaciones entre los distintos objetos presentes en la aplicación, para cada uno de los escenarios definidos anteriormente.

2.1.7.1 Diagrama de colaboración 1: Autenticar Persona (Usuario Administrativo)

- *Escenario:* 1.
- *Caso de Uso:* 2.
- *Descripción:* El primer evento a generarse cuando una persona desea autenticarse es la captura de la huella del dedo de la misma en el Host

Administrativo, la cual se ejecuta cuando la persona coloca su dedo en el lector de huellas y éste extrae la imagen de la misma como se observa en la figura 2.21. A continuación se realizan las operaciones relacionadas con la extracción de la huella a partir de su imagen y se consulta al servidor de Base de Datos la información de las huellas para identificar al usuario.

2.1.7.2 Diagrama de colaboración 2: Enrolar Cliente

- *Escenario:* 2.
- *Caso de Uso:* 3.
- *Descripción:* La persona autorizada a enrolar un cliente debe recolectar toda la información necesaria del mismo y posteriormente interactuar con la aplicación, para guardar dicha información a través de la aplicación en el servidor de Base de Datos mediante el objeto Base de Datos; este último retorna los resultados del almacenamiento al Host Administrativo como se indica en la figura 2.22.

2.1.7.3 Diagrama de colaboración 3: Actualizar Cliente

- *Escenario:* 3.
- *Caso de Uso:* 4.
- *Descripción:* La persona autorizada a actualizar un cliente debe solicitar la captura de la huella del dedo registrado del mismo, para verificar que efectivamente se encuentra ingresado en la base de datos; en el proceso de verificación el Host Administrativo crea un objeto Base de Datos para establecer la conexión con el servidor de Base de Datos, para consultar la información necesaria para la verificación del cliente a actualizar. Si la verificación es correcta el Usuario Administrativo recolecta la información del cliente que se va a actualizar y posteriormente interactuará con la aplicación para guardar dicha información en la base de datos como indica el diagrama de la figura 2.23.

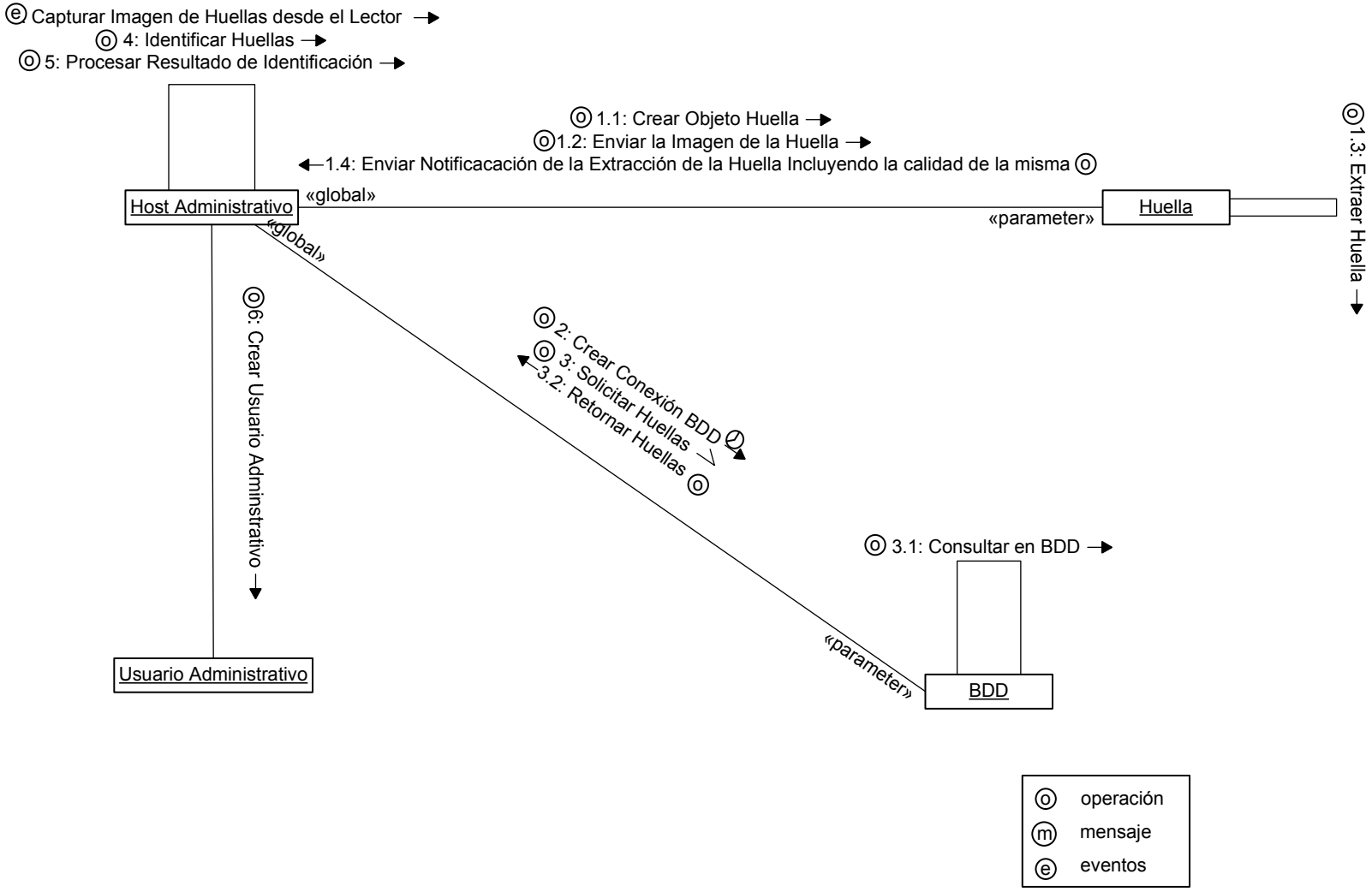


Figura 2. 21: Diagrama de Colaboración 1 - Autenticar Persona (Usuario Administrativo)

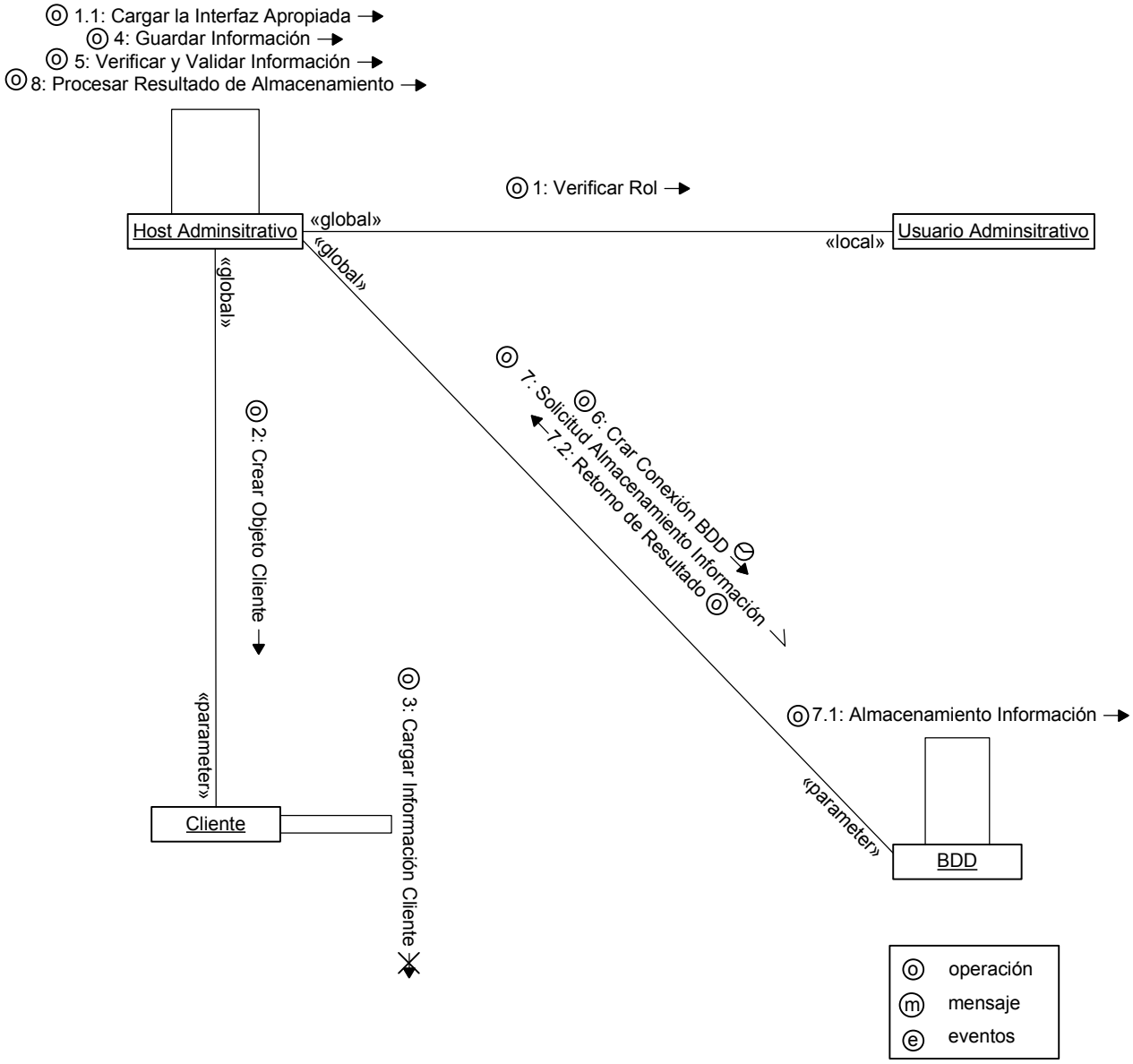


Figura 2. 22: Diagrama de Colaboración 2 - Enrolar Cliente

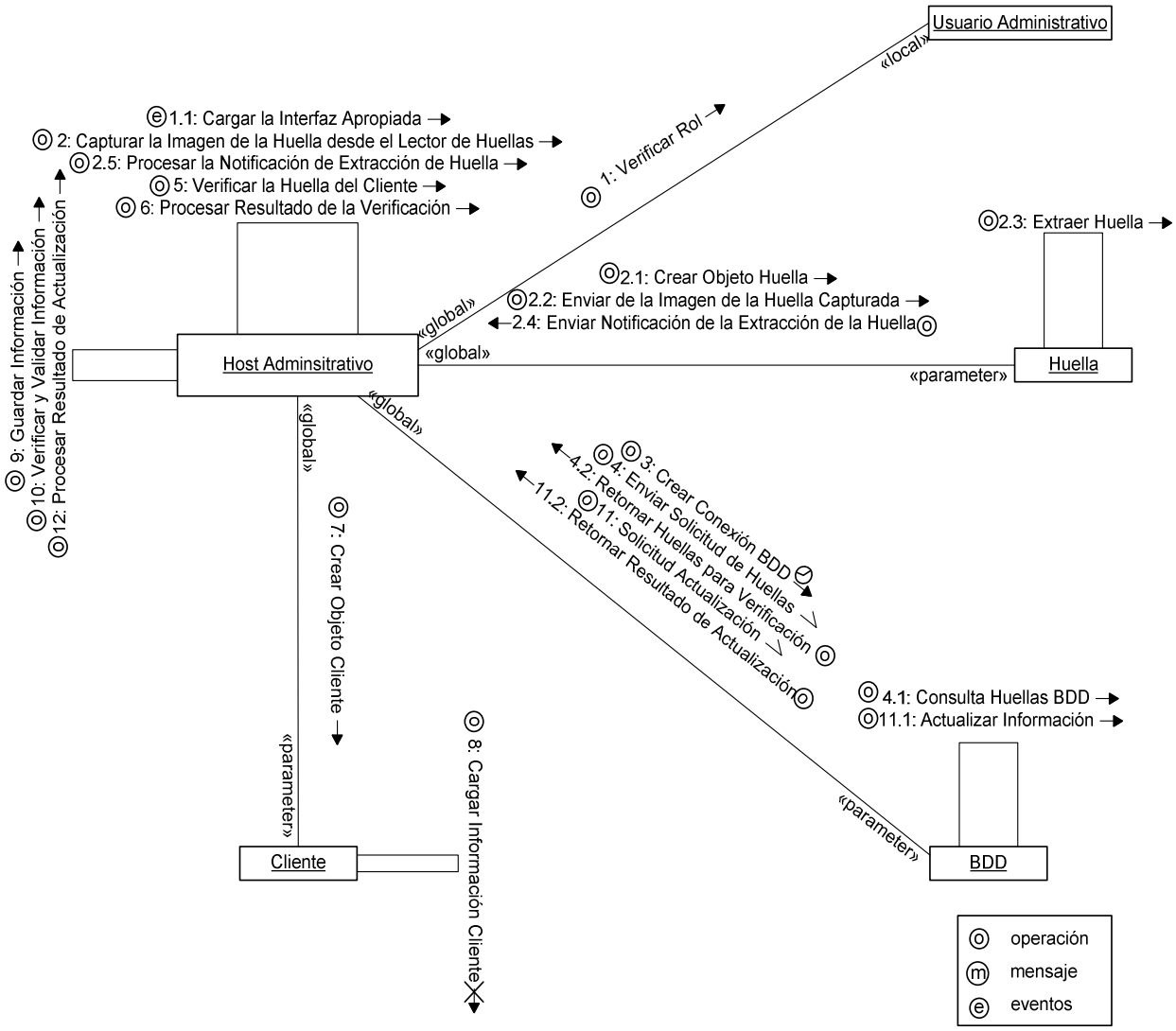


Figura 2. 23: Diagrama de Colaboración 3 - Actualizar Cliente

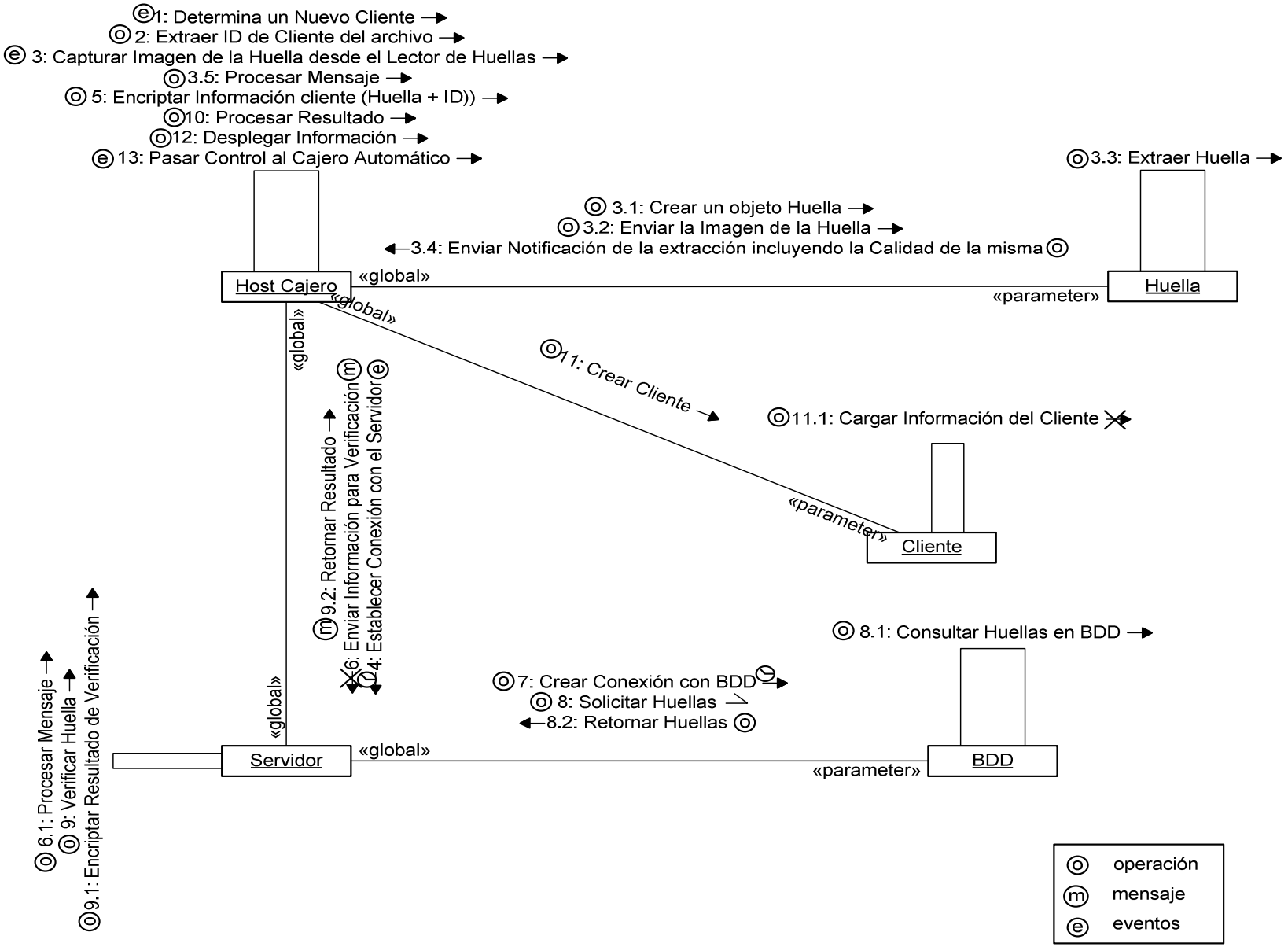


Figura 2. 24: Diagrama de Colaboración 4 - Verificar Cliente

2.1.7.4 Diagrama de colaboración 4: Verificar Cliente

- *Escenario:* 4.
- *Caso de Uso:* 5.
- *Descripción:* Se detecta un nuevo cliente en busca de autenticación mediante la lectura de un archivo y se captura la imagen de la huella dactilar del cliente, como se indican en los dos primeros eventos (secuencia 1 y 3) del diagrama de la figura 2.24; de esta imagen se extrae la plantilla de la huella dactilar y junto con el número de identificación del cliente (obtenido de un archivo), se encriptan en un mensaje que será enviado al servidor para la verificación. Para el proceso de verificación el Host Cajero establece una conexión con el servidor encargado de verificar al cliente y se comunican mediante el protocolo TCP/IP, para intercambiar mensajes con información relacionada a la verificación (huella y número de identificación del cliente a verificar y resultados de la verificación); si la verificación fue correcta se despliega la información respectiva y se pasa el control a la aplicación existente en el cajero.

2.1.7.5 Diagrama de colaboración 5: Administrar Sistema

- *Escenario:* 5.
- *Caso de Uso:* 6.
- *Descripción:* Una vez que el usuario administrativo verifica que está autorizado para administrar el sistema se carga la interfaz apropiada para que el cliente pueda realizar las operaciones de administración deseadas, como se indica en el diagrama de colaboración de la figura 2.25.

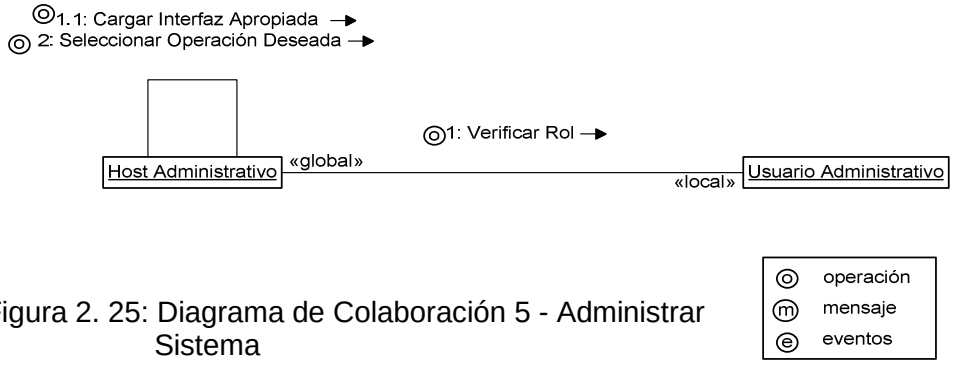


Figura 2. 25: Diagrama de Colaboración 5 - Administrar Sistema

2.1.8 DIAGRAMAS DE ESTADOS

En los siguientes diagramas se representan los estados por los que atraviesan los objetos descritos anteriormente a causa de los eventos recibidos y las respuestas que presentan ante dichos eventos.

2.1.8.1 Diagrama de Estados 1

Descripción: En el diagrama de la figura 2.26 se puede observar que en el estado inicial de *Autenticar Personas,* el Lector de Huellas espera que un usuario en busca de autenticación coloque su dedo para capturar su huella e identificar si el usuario se encuentra registrado o no; si la autenticación falla se pasa a un estado de *error* que retorna al estado anterior, si la autenticación es correcta pasa a un estado *Verificación de Rol,* en el que se verifica el tipo de actividad que puede realizar el usuario administrativo (enrolar clientes, actualizar clientes y/o administrar el sistema). El usuario selecciona la actividad que desea realizar y pasa al estado correspondiente (*Enrolar Clientes*, *Actualizar Clientes* y *Administrar el Sistema*) donde cada uno tiene sus propias acciones; si el usuario desea realizar nuevamente la misma actividad se mantiene en el estado que se encuentra caso contrario vuelve al estado de *autenticar personas*.

2.1.8.2 Diagrama de Estados 2

Descripción: En el estado inicial el sistema obtiene el número de identificación del cliente y espera que el cliente coloque su dedo sobre el lector de huellas dactilares para capturar su huella y extraerla. Una vez recibidos estos eventos y realizadas las acciones correspondientes en el estado *Esperar Cliente,* como se indica en el diagrama de la figura 2.27, pasa a un estado de *Verificar Cliente* donde se comprueba si el cliente se encuentra registrado o no; si la verificación da un resultado negativo la actualización falla y se pasa a un estado de error que retorna al estado anterior. Si la verificación da un resultado positivo la autenticación es correcta y pasa a un estado de *Cliente Aceptado*, en el que se pasa el control a la aplicación externa del cajero.

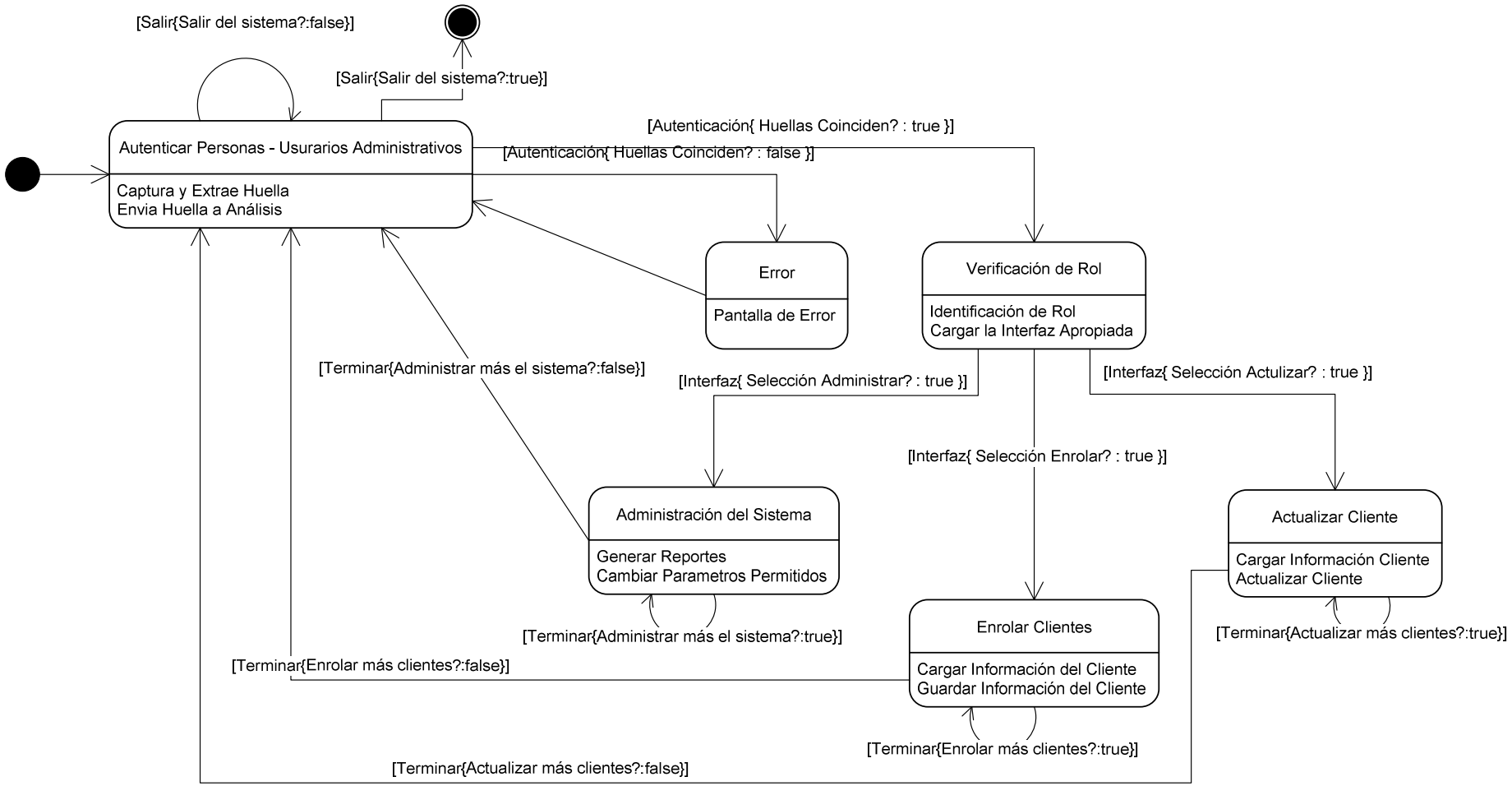


Figura 2. 26: Diagrama de Estados 1

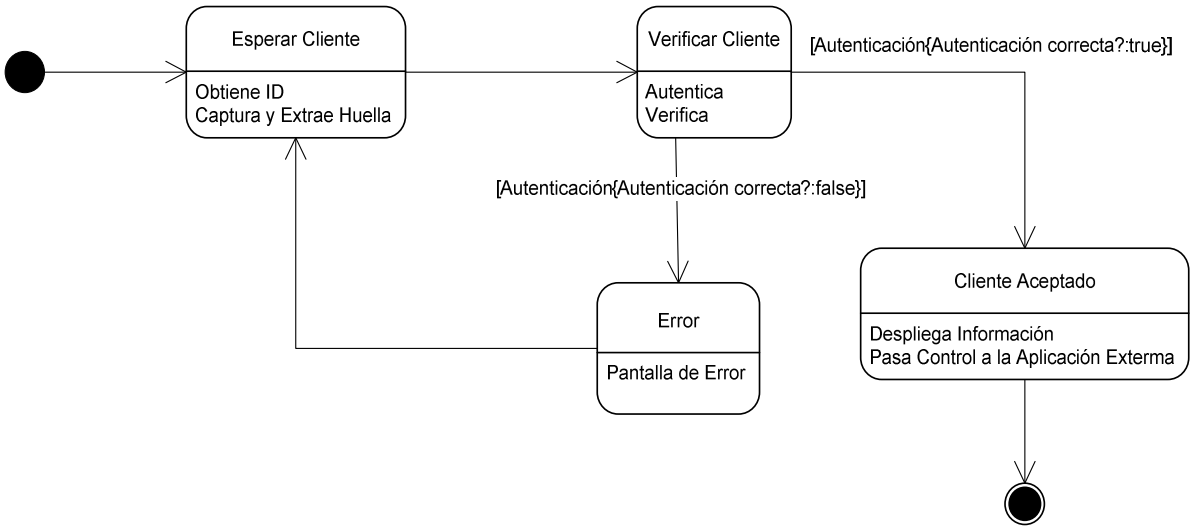


Figura 2. 27: Diagrama de Estados 2

2.1.9 DIAGRAMAS DE ACTIVIDAD

Descripción: El diagrama de la figura 2.28 representa las actividades que realiza la aplicación cuando se ha generado un evento interno como es la necesidad de comunicarse con el servidor mediante el protocolo TCP/IP, para lo cual es necesario determinar la dirección IP y el puerto del host local (Host Cajero o Host Administrativo) y luego la dirección IP y el puerto del equipo remoto (en este caso el servidor). Identificados estos parámetros se procede a establecer la conexión, si el establecimiento de la conexión falla lo vuelve a intentar, caso contrario los mensajes a transmitirse deberán encriptarse y se procederá a abrir la conexión para transmitir los mensajes. Una vez terminada la transmisión se procede a cerrar la conexión.

2.1.10 DIAGRAMAS ESTÁTICOS

En la figura 2.29 se representa el diagrama estático de la aplicación. Este diagrama se realizó en base a los diagramas anteriores y se muestran todas las clases u objetos a ser desarrollados. Este diagrama es el sustento para la creación de las aplicaciones del servidor, del host administrativo y del host del cajero.

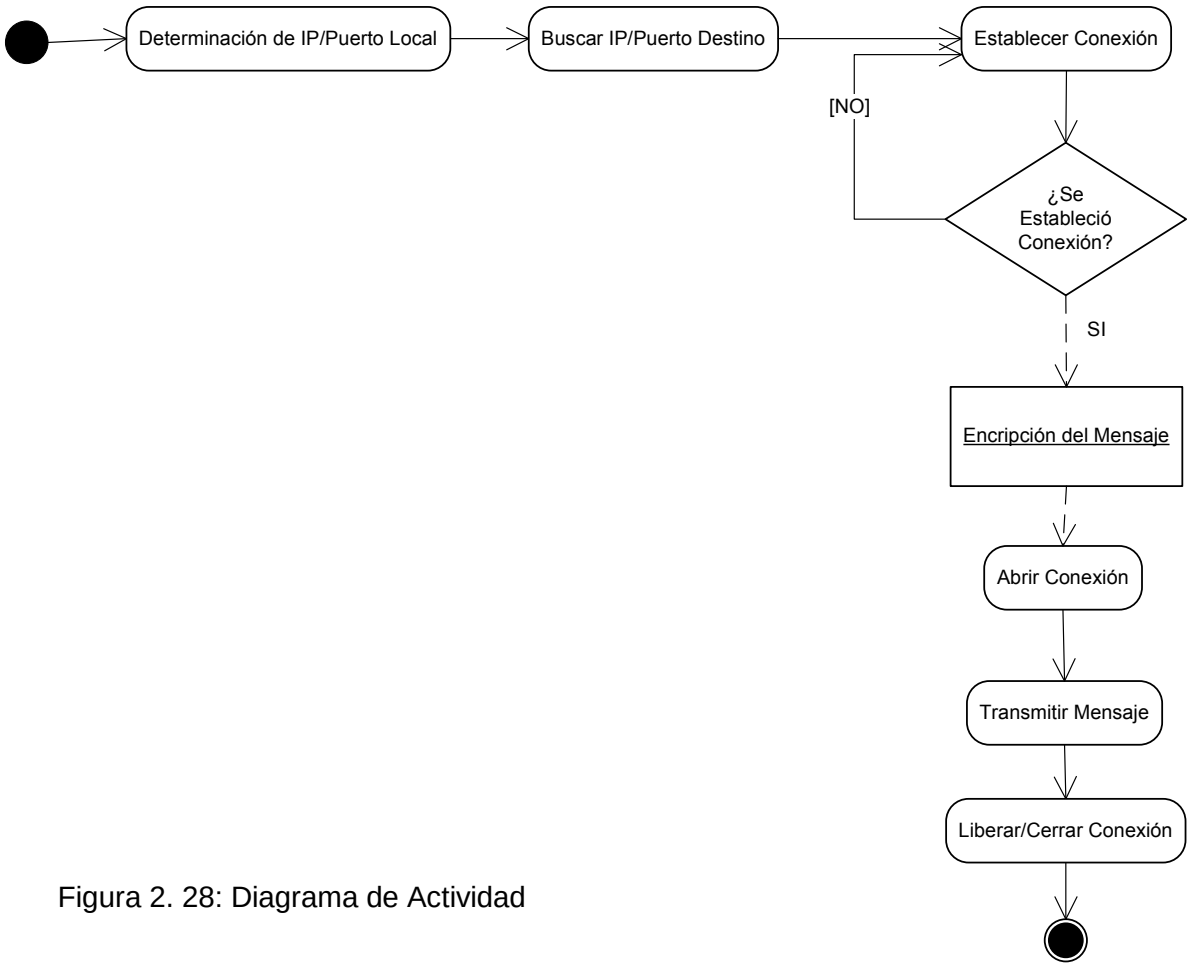


Figura 2. 28: Diagrama de Actividad

2.1.11 DIAGRAMAS DE IMPLEMENTACIÓN

Estos están relacionados a detalles de la implementación en cuanto a código y características de ejecución.

2.1.11.1 Diagramas de Componentes

- *Categoría de Clases:* En el diagrama de la figura 2.30 se indica a *ATMDactilar* como la categoría de clase principal, de donde se tiene el *Host* como parte de esta categoría el cual representa una clase abstracta que puede representar un *Host Cajero, Host Administrativo* o *Servidor;* además un Host tendrá *Usuarios* y *Utilitarios*, donde los Usuarios pueden ser *Cliente* o *Usuario Administrativo*. A su vez, la clase *Servidor* está relacionada a una clase *Base de Datos*, la misma que interactúa con una base de Datos mediante la clase de *procedimientos almacenados*.

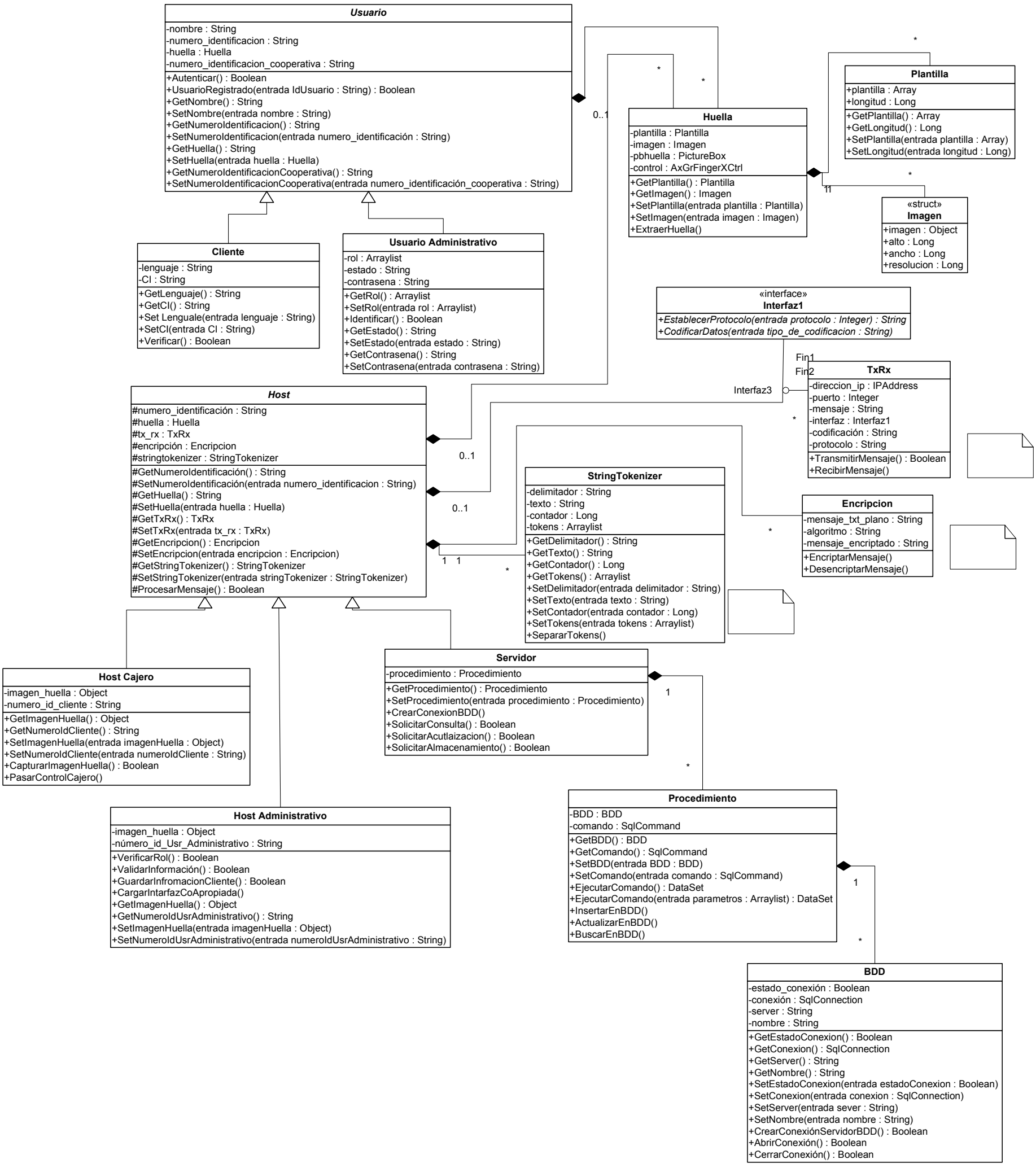


Figura 2. 29: Diagrama Estático de la Aplicación

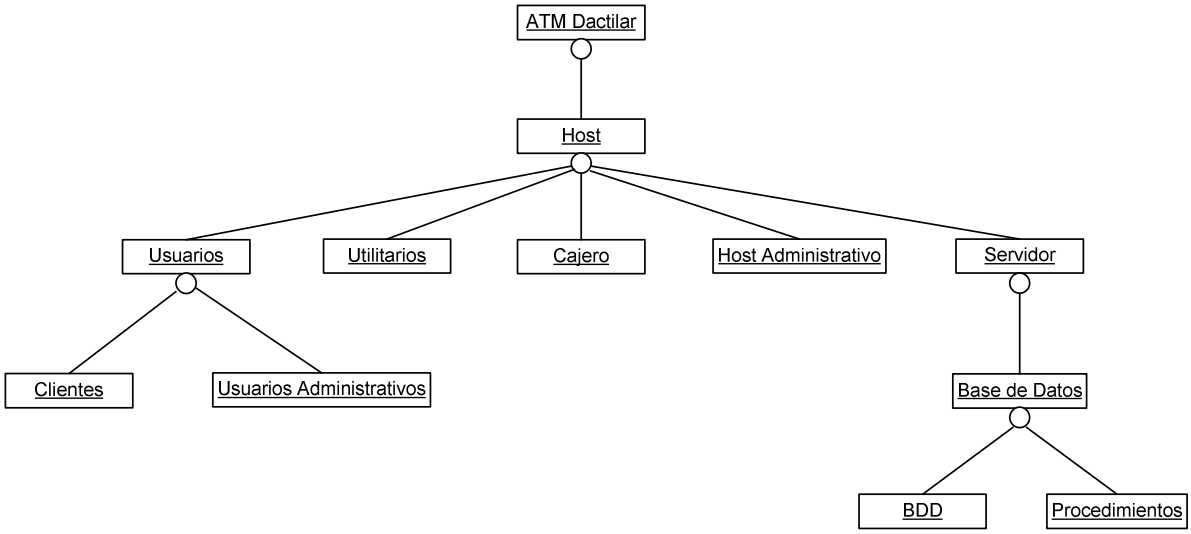


Figura 2. 30: Categoría de Clases

- *Diagrama de Paquetes*: Los paquetes de implementación se ven representados en la figura 2.31, de donde se tiene el paquete ATMdactilar que corresponde a la aplicación en sí, el paquete del Kit de Desarrollo para extracción y procesamiento de Huellas, el controlador del Lector de Huellas Dactilares.

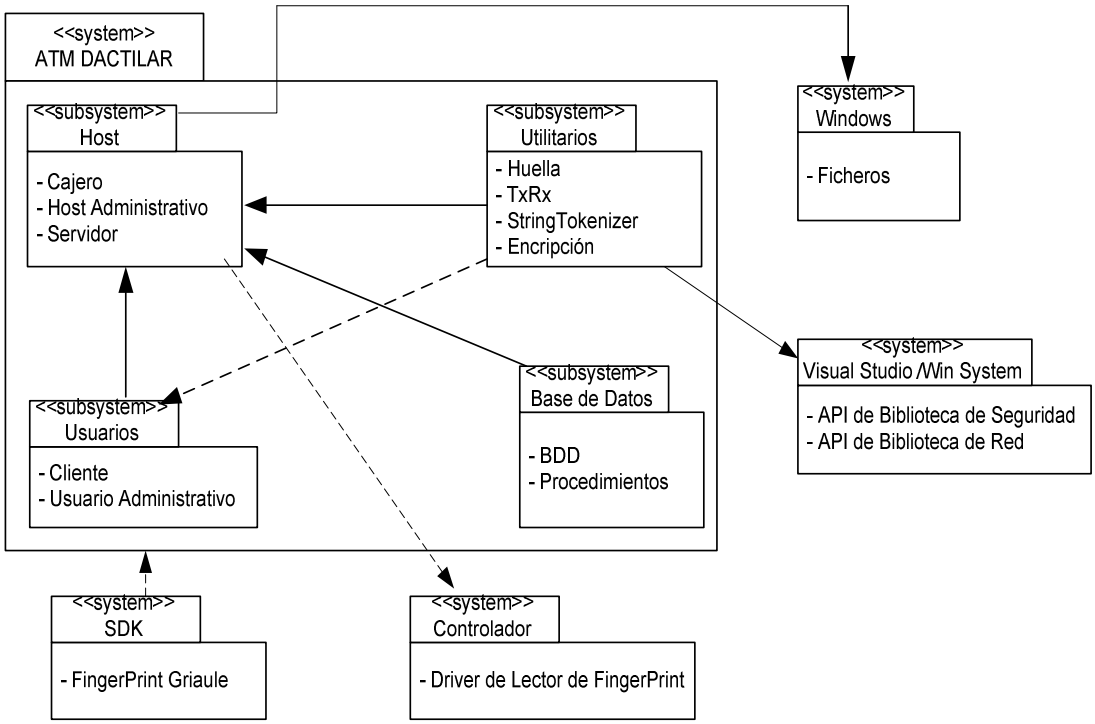


Figura 2. 31: Diagrama de Paquetes

También se tienen el paquete de Windows que proporciona sus ficheros y las API del sistema dentro del paquete Visual Studio/Win System.

- *Diagrama de Módulos:* En el diagrama de la figura 2.32 se muestran los módulos del sistema entre los cuales se tienen el módulo ATMDactilar que corresponde al programa principal y se relaciona con los módulos Servidor, Cajero y Host Administrativo, los mismos que representan una especificación y a su vez un subsistema; estos tres módulos se relacionan con el módulo utilitarios y el módulo Servidor, tiene una relación adicional con el módulo Base de Datos.

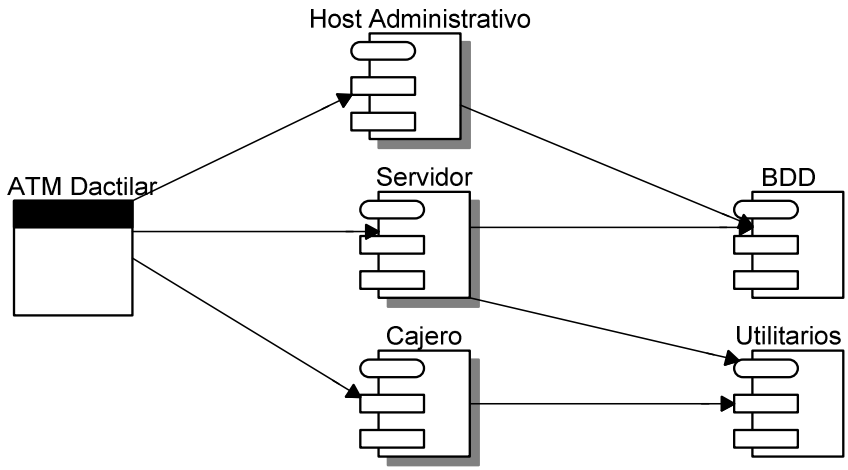


Figura 2. 32: Diagrama de Módulos

2.1.11.2 Diagramas Desplegables

El diagrama desplegable ilustrado en la figura 2.33 representa los procesadores y dispositivos presentes en el sistema y sus conexiones. Se tienen tres dispositivos básicos que son el Cajero, el Host Administrativo y el Servidor donde cada uno posee su propio procesador y periféricos. Tanto el Cajero como el Host Administrativo se conectan al Servidor y se comunican a través del protocolo TCP/IP.

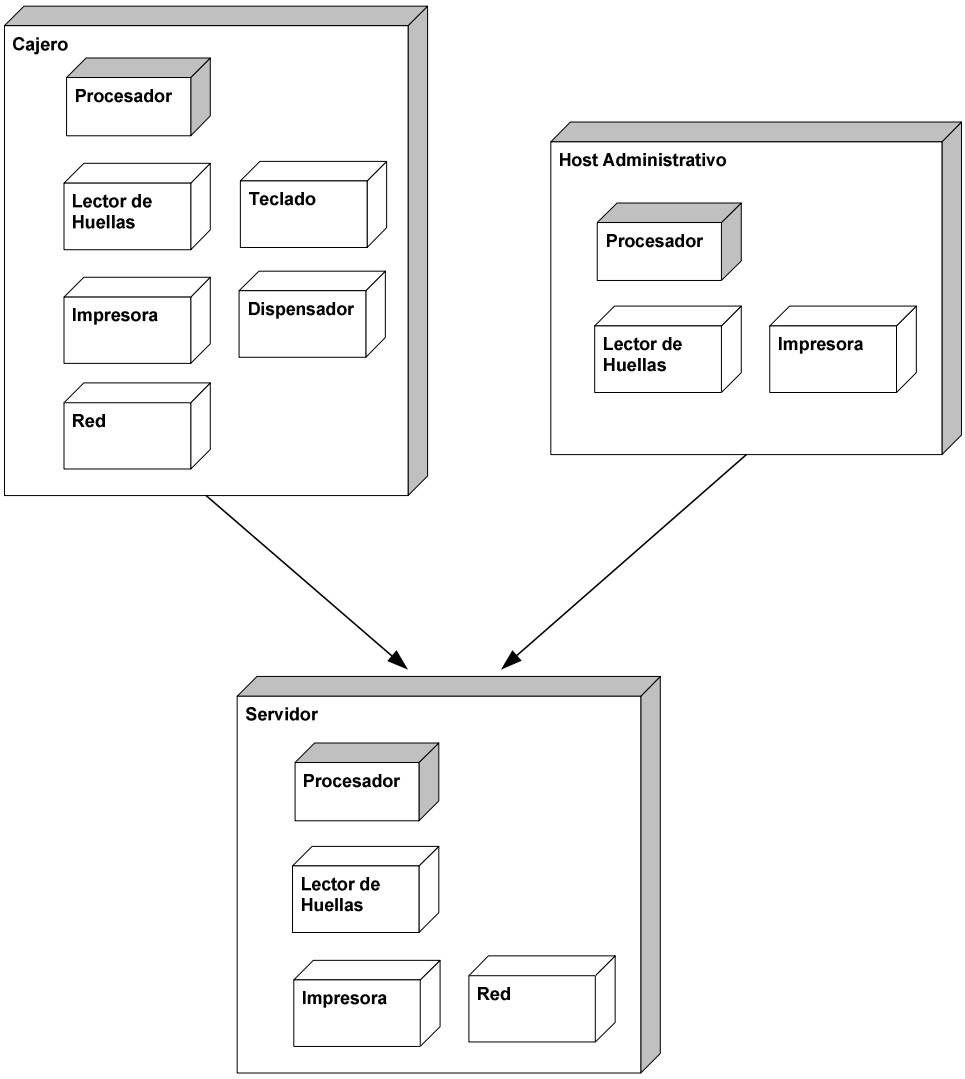


Figura 2. 33: Diagrama Desplegable

2.2 DISEÑO DE BASE DE DATOS

Para el diseño de la base de datos, se empleará un modelo relacional por las ventajas y características descritas en la sección 1.3; el lenguaje en el cual se realizará la base de datos es SQL. Por la amplia difusión, por la potencia del motor y por otras características ya descritas en el capítulo 1, la base de datos se la implementará sobre *Microsoft SQL Server* 2005, en una arquitectura de 3 capas.

2.2.1 REQUERIMIENTOS DE DATOS

A partir de los diagramas de la sección anterior, y basados principalmente en los diagramas estáticos se pueden definir los siguientes requerimientos de datos (adicionalmente se especifica el tipo de datos):

- *Usuario Administrativo:*
    - Número de Identificación: Alfabético.
    - Nombre: Alfabético.
    - Identificación de la Cooperativa: Alfabético.
    - Huella: Huella.
    - Contraseña: Numérico (Entero).
    - Identificación de Rol: Numérico (Entero).
- *Cliente:*
    - Número de Identificación: Alfabético.
    - Nombre: Alfabético.
    - Identificación de la Cooperativa: Alfabético.
    - Huella: Huella.
    - Idioma: Alfabético.
    - Usuario Administrativo que Registro: Alfabético.
- *Rol:*
    - *Nombre: Alfabético.*
    - *Descripción: Alfabético.*
- *Huella:*
    - Plantilla: Datos binarios.
    - Tamaño: Numérico (Entero).
    - Dedo: Numérico (Entero).
    - Usuario: Alfabético.

- *Cooperativa:*
    - Número de Identificación: Alfabético.
    - Nombre: Alfabético.
- *Cajero:*
    - Número de Identificación: Alfabético.
    - Ubicación: Alfabético.
    - Sistema: Alfabético.
    - Cooperativa: Alfabético.
- *Log:*
    - Usuario: Alfabético.
    - Cajero: Alfabético.
    - Estado: Alfabético.
    - Fecha: Tiempo.

2.2.2 MODELO DE BASE DE DATOS

En base a los requerimientos descritos en el subcapítulo 2.2.1, se ha diseñado la base de datos modelada en la figura 2.34.

2.3 DICCIONARIO DE DATOS

El diccionario de datos tanto de la aplicación como de la base de datos se adjunta por su extensión en el anexo F.

2.4. IMPLEMENTACIÓN

La implementación ha sido desarrollada en función del diseño realizado en las secciones 2.2 y 2.1, tanto para la aplicación como para la base de datos respectivamente; a continuación se detalla la implementación de cada una de ellas.

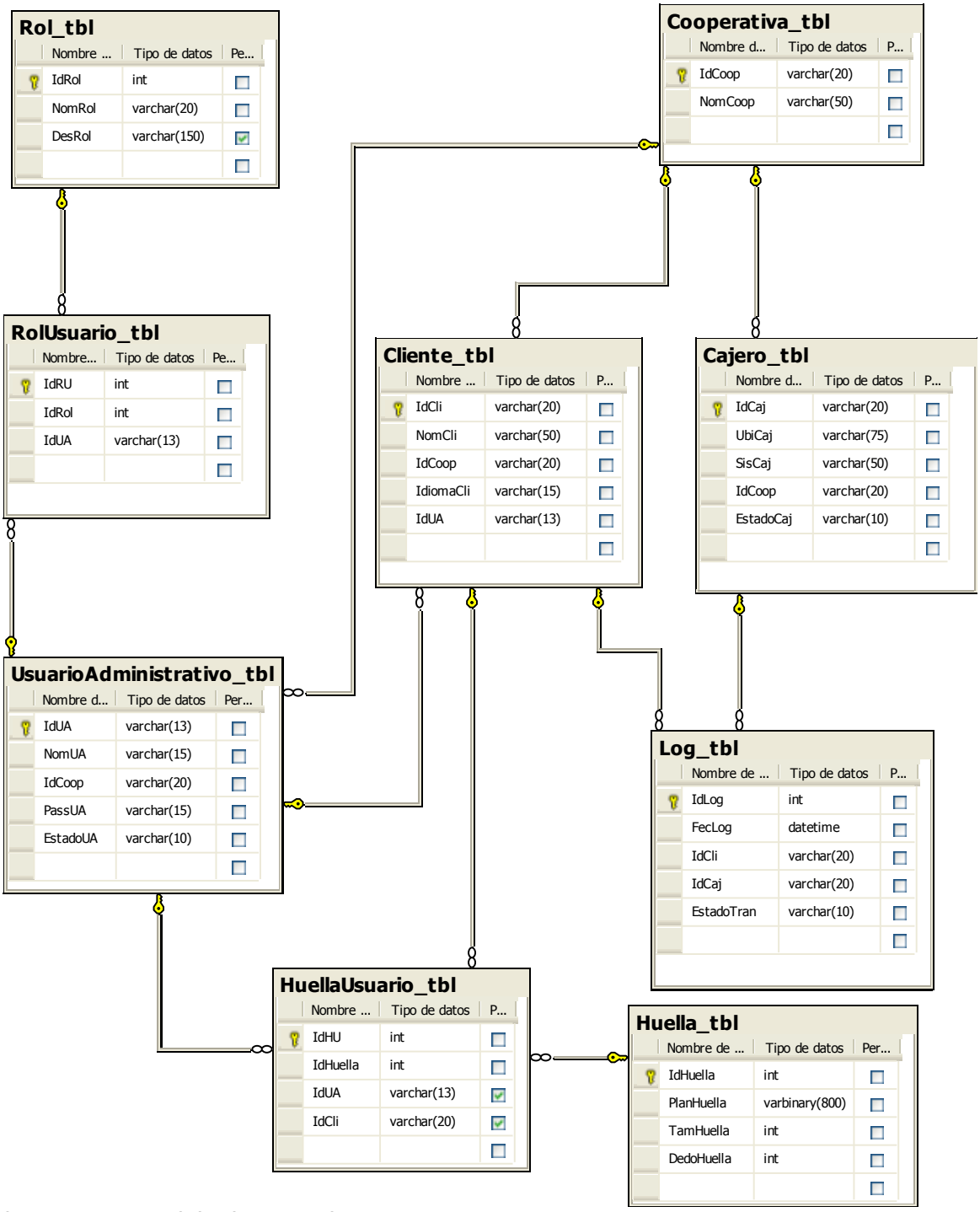


Figura 2. 34: Modelo de Base de Datos

2.4.1 APLICACIÓN

El lenguaje utilizado para el desarrollo de la aplicación es Visual Basic .NET en su versión Visual Basic 2005 por las características y facilidades que ofrece.

Ha sido necesario diseñar una aplicación para cada sistema, es decir, una para el sistema *Host Cajero* llamada *HATMDactilar*, otra para el sistema *Host Administrativo* que corresponde a la aplicación *AATMDactilar* y finalmente otra para el sistema *Servidor* denominada *SATMDactilar*. Todas ellas están compuestas de uno o más formularios que permiten al usuario interactuar con cada una de las aplicaciones y opciones requeridas. Dentro de cada aplicación existe uno o más espacios de nombres que permiten una agrupación lógica de las clases, así se tiene:

- Un espacio de nombres para agrupar todas las clases relacionadas a los usuarios denominado *Usuario*.
- Un espacio de nombres para agrupar las clases correspondientes al manejo de datos llamado *BDD*.
- El espacio de nombres denominado *Huellas* que contiene todas las clases relacionadas al manejo de una huella dactilar.
- El espacio de nombres *TxRx*, el mismo que agrupa todas las clases correspondientes a la comunicación cliente-servidor mediante los protocolos IP y TCP.
- Finalmente el espacio de nombres *Seguridad* contiene las clases que permiten la encripción y desencripción de mensajes.

Estos espacios de nombres no son requeridos por todas las aplicaciones por las funcionalidades de cada una de ellas, así se tiene que la aplicación *HATMDactilar* solo requiere de Huellas, TxRx y *Seguridad*; la aplicación *AATMDactilar* solo tiene los espacios *Huellas*, *BDD* y *Usuario*, y la *SATMDactilar* si requiere de todos estos espacios.

Además de los espacios de nombres, las aplicaciones *HATMDactilar* y *SATMDactilar* utilizan los siguientes módulos:

- El módulo *FuncionesUtilitarias*, que así como su nombre lo indica, contiene funciones y métodos adicionales requeridos por la aplicación para el procesamiento de texto y conversión de funciones.
- *MensajesRespuesta* es un módulo correspondiente solo a la aplicación SATMDactilar y está compuesto por los valores que conformarán el mensaje de respuesta al cliente *HATMDactilar*.

Tanto los espacios de nombres como los módulos, así como las clases, métodos y funciones de estas aplicaciones se encuentran detallados en el Diccionario de datos en el anexo F.

2.4.1.1 Manejo de Huellas Dactilares

Para el manejo de las huellas dactilares se ha utilizado el SDK de Griaule Biometrics conocido como *Fingerprint SDK* en su versión trial de 90 días. Entre los lectores soportados por este SDK se ha seleccionado el lector *Nitgen Hamster I* para la implementación del prototipo, por sus características de lectura rápida de la imagen de la huella y por su gran utilización para la autenticación de personas.

Para la utilización de *Fingerprint SDK* en Visual Basic 2005 fue necesario añadir los DLLs y el componente *ActiveX* del SDK en cada una de las aplicaciones, para ser utilizados de la siguiente manera:

- La aplicación *HATMDactilar* requiere de este componente y sus librerías para el manejo del lector de huellas dactilares y la extracción de plantillas de las mismas, para posteriormente ser enviadas al *Servidor* donde la aplicación *SATMDactilar* se encarga de la respectiva verificación de los *Clientes.*
- La aplicación *SATMDactilar* realiza la verificación de los clientes en base al identificador de la tarjeta y a la huella dactilar del portador de la misma, con el fin de garantizar que la persona que desea acceder a los servicios del cajero sea efectivamente el dueño de la tarjeta. El SDK proporciona la función que permite realizar la verificación de la huella dactilar de un *Cliente* en base a un puntaje mínimo de aceptación, y consiste en comparar la plantilla de la huella dactilar proveniente del *Host Cajero* con la plantilla consultada de la base de datos perteneciente al portador de la

tarjeta cuyo número identificador es el recibido en el mensaje. Si la verificación da como resultado un puntaje mayor al establecido como mínimo, este puntaje es retornado por la función, caso contrario retorna cero, o en su defecto si se produce un error retornará el valor correspondiente al error generado (valor menor a cero). El puntaje representa el grado de coincidencia hallado entre la plantilla a verificar y la plantilla original. El puntaje de referencia se selecciona en base a la tasa de falsa coincidencia (FMR o falso positivo[21]) y a la tasa de falsa no coincidencia (FNMR o falso negativo[22]). Estos dos valores forman una curva y el punto en donde son iguales se le conoce como Tasa de Igual Error (EER por sus siglas en inglés), siendo éste el valor referencial en el cual la probabilidad de tener un falso positivo es igual a la de tener un falso negativo. Estas curvas se pueden apreciar en la figura 2.35. A partir de esta curva se puede deducir que mientras mayor sea el puntaje referencial se minimiza el FMR, dándole mayor robustez en cuanto a la seguridad a la aplicación, pero al mismo tiempo se maximiza el FNMR con lo que se aumenta la probabilidad de que usuarios válidos sean rechazados por el sistema. El valor recomendado por el proveedor de este SDK es de 25 para procesos de verificación.

- La aplicación *AATMDactilar* demanda las librerías y componentes del SDK para el manejo del lector de huellas dactilares, extracción de plantillas, identificación y verificación de usuarios. La identificación de *Usuarios Administrativos* se realiza al inicio de la aplicación para cargar el formulario que le permite realizar las respectivas tareas administrativas, también para confirmar las transacciones que requieren mayor seguridad como son la actualización de *Clientes* o de otros *Usuarios Administrativos*. Al igual que en la verificación, el SDK proporciona una función que permite identificar la plantilla de la huella dactilar de un usuario en base a un puntaje, éste será más alto que el de verificación ya que no existen más parámetros que garanticen la identidad del usuario; los valores retornados por esta función serán *GR_MATCH* si el puntaje de identificación es mayor al establecido

[21] *Falso Positivo:* Reconoce como usuario genuino a un impostor.
[22] *Falso Negativo:* Reconoce como impostor a un usuario genuino.

como referencia, *GR_NOT_MATCH* si este puntaje es menor al referencial, o en su defecto retornará un valor menor a cero si se produce un error durante el proceso de identificación. El puntaje seleccionado para identificación es de 60, el mismo que garantiza una baja tasa de falsos positivos (de 300000 resultados 1 podría ser falso positivo). Para autenticar a un cliente se ha desarrollado una función que permita comparar la huella capturada por el lector con cada una de las huellas registradas en la base de datos utilizando la función de identificación de Fingerprint SDK.

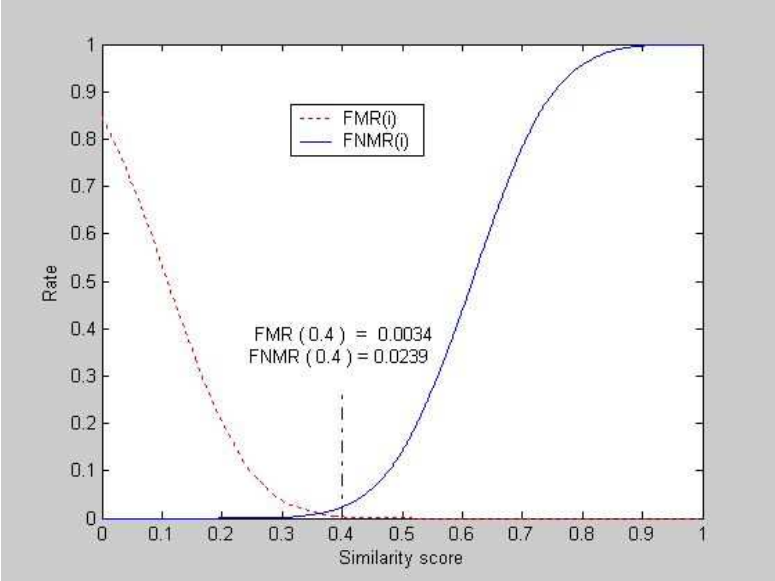


Figura 2. 35: Curvas de FMR y FNMR [16]

2.4.1.2 Manejo de la comunicación TCP/IP

Como se indicó anteriormente el sistema *Host Cajero* y el *Servidor* deben establecer una comunicación mediante los protocolos IP y TCP, para lo cual fue creado el espacio de nombres *TxRx*, el mismo que agrupa las clases correspondientes al cliente en la aplicación *HATMDactilar* del *Host Cajero*, y agrupa las clases correspondientes al *Servidor* en la aplicación *SATMDactilar* en el *Servidor.*

Para el manejo de la comunicación entre estos dos entes fue necesaria la utilización de hilos (*threads* o subprocesos) para que se encarguen del control de la comunicación entre el cliente TCP y el servidor TCP, de esta manera en el *Servidor* se crea un hilo para manejar el *socket* que está escuchando (el número

de puerto es establecido en una clave del registro de Windows) por peticiones de conexión de los clientes, mientras que en el *Host Cajer*o se crea un hilo para solicitar al *Servidor* establecer una conexión. El establecimiento de comunicación se maneja mediante un *handshake* de tres vías. Una vez que la petición del cliente ha llegado al *Servidor,* éste crea otro hilo para manejar la comunicación a través de otro puerto (dejando así libre el puerto inicial para que otros clientes puedan solicitar servicio) y envía un mensaje de confirmación al *Host Cajero,* quien a su vez confirma el establecimiento de la conexión. A continuación el *Host Cajero* envía al *Servidor* el mensaje con los correspondientes parámetros para la verificación de un *Cliente* y después de ser procesado el mensaje por el *Servidor,* éste envía al *Host Cajero* un mensaje con el resultado de la verificación. Cuando se recibe el resultado de la verificación el *Host Cajero* procede a terminar la conexión con el *Servidor* mediante un *handshake* de cuatro pasos. Una vez terminada la conexión tanto el *Host Cajero* como el Servidor cierran los subprocesos que les permitió manejar la comunicación. Este proceso se puede apreciar en la figura 2.36.

2.4.1.3 Manejo de la Seguridad

Los mensajes que son intercambiados entre el *Host Cajero* y el *Servidor* después de ser establecida la conexión deben ser encriptados por políticas de seguridad, para lo cual se ha empleado 3DES como el mecanismo de encripción por los beneficios que ofrece (Ver Capítulo 1). Visual Basic 2005 ofrece algunas clases que permiten la encripción y desencripción mediante este algoritmo, las mismas que han sido empleadas dentro de las clases desarrolladas para este proceso en el espacio de nombres *Seguridad*. Adicionalmente antes de enviar el mensaje del *Host Cajero* al *Servidor*, se aplica a la plantilla una compresión de los datos mediante una conversión de base decimal a base 36, que dependiendo de los valores de la plantilla, ésta puede reducirse un cuarenta por ciento en promedio. Estas funciones de conversión se encuentran en el módulo correspondiente a *FuncionesUtilitarias*.

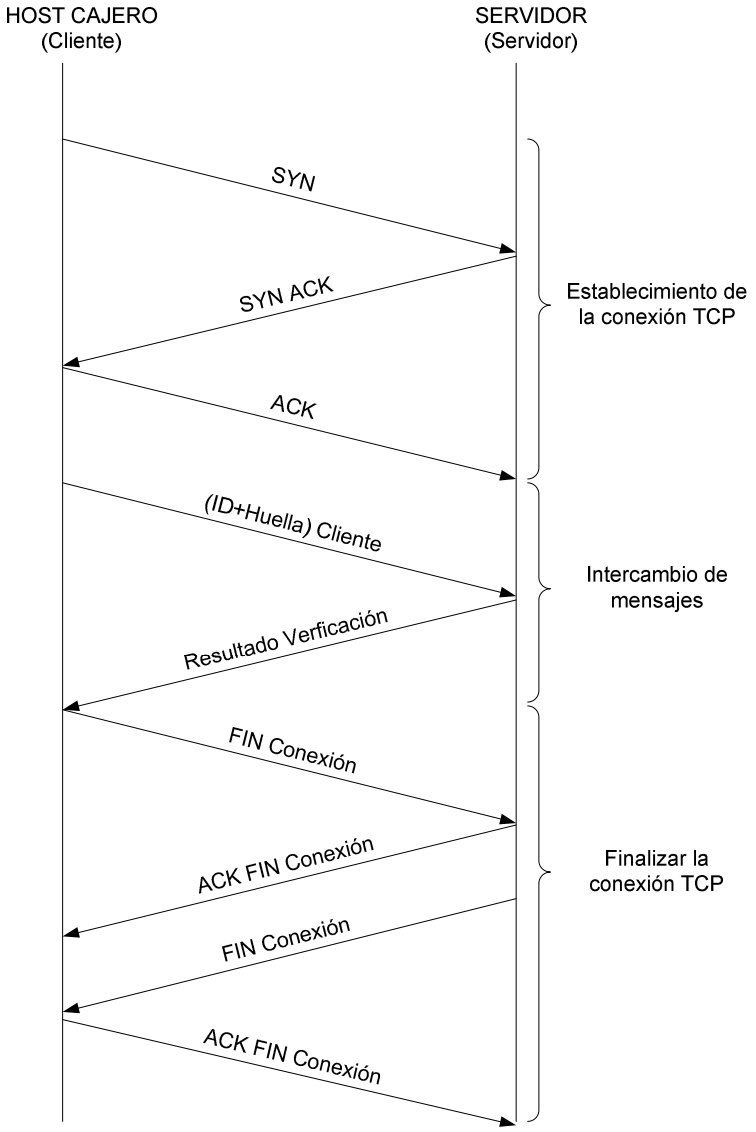


Figura 2. 36: Comunicación Host Cajero- Servidor

2.4.1.4 Manejo de Datos

Para el manejo de los datos .NET Framework ofrece los *datasets* que son espacios de memoria donde se almacenan temporalmente los datos consultados de la base de datos a través de llamadas a los procedimientos almacenados. De esta manera la aplicación solo se conecta a la base de datos el momento que desea realizar la respectiva consulta y los *datasets* son llenados (a través de un objeto del tipo *SqlDataAdapter*); posterior a ese proceso la conexión con la base de datos es cerrada. De la misma manera para almacenar o actualizar

información, la aplicación se conecta a la base de datos para ejecutar el respectivo procedimiento almacenado a través de la función *ExecuteNonQuery.*

Cabe destacar que la aplicación nunca ejecuta comandos directos desde su código en la base, en vez de esto y por seguridad siempre ejecuta procedimientos almacenados con lo que se garantiza que no sean posibles ataques del tipo *SQL Insertion*.

2.4.2 APLICACIONES DESARROLLADAS

En las siguientes secciones se describe la funcionalidad de cada una de las 3 aplicaciones que conforman el sistema en general.

2.4.2.1 Aplicación "HATMDactilar"

Esta aplicación debe ser instalada en todos los *Cajeros* que estarán conectados al *Servidor.* En el momento de ser iniciada se despliega una ventana que contiene un listado de todos los eventos generados mientras la aplicación está corriendo, además se inicia un subproceso que ha sido creado para leer un archivo denominado *FingerprintRequest.* La aplicación *HATMDactilar* debe determinar cada 200 milisegundos si la información en este archivo ha sido modificada; esta información consiste en un *timestamp* (fecha y hora) y el identificador de la tarjeta de un cliente, como se detalla en la figura 2.37.

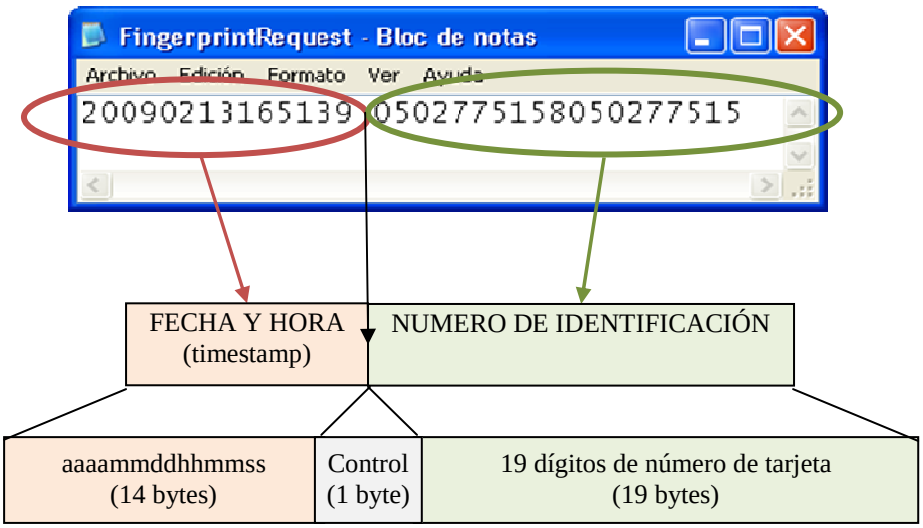


Figura 2. 37: Información contenida en el archivo *FingerprintRequest*

Si el archivo ha sido modificado significa que un nuevo *Cliente* desea ser autenticado para acceder a los servicios del *Cajero.* La aplicación existente en el cajero ha modificado el archivo con la información del día y la hora en que el cliente ha ingresado su tarjeta, del mismo modo que se escribe el número identificador de la misma; posteriormente la aplicación HATMDactilar pasa a un estado en el que espera (mediante un temporizador) que el *Cliente* coloque su dedo sobre el lector de huellas. Si la huella ha sido capturada durante este intervalo de tiempo se emite al Servidor el mensaje correspondiente para la verificación, caso contrario la aplicación escribe en otro archivo denominado *FingerprintResponse* el resultado *timeout huella* y vuelve a su estado inicial. Si se da el caso en el que se colocó la huella, una vez enviado el mensaje al servidor, la aplicación espera otro intervalo de tiempo (mediante otro temporizador) para recibir el mensaje de respuesta a la autenticación proveniente del *Servidor;* si el mensaje no ha llegado después de transcurrido ese intervalo de tiempo la aplicación escribe en el archivo *FingerprintResponse* el resultado correspondiente a *timeout servidor*; caso contrario escribe en este archivo el mensaje recibido del *Servidor* y vuelve a su estado inicial. El archivo *FingerprintResponse* también contiene un *timestamp* igual al leído del archivo *FingerprintRequest,* seguido del resultado respectivo como se indica en la figura 2.38.

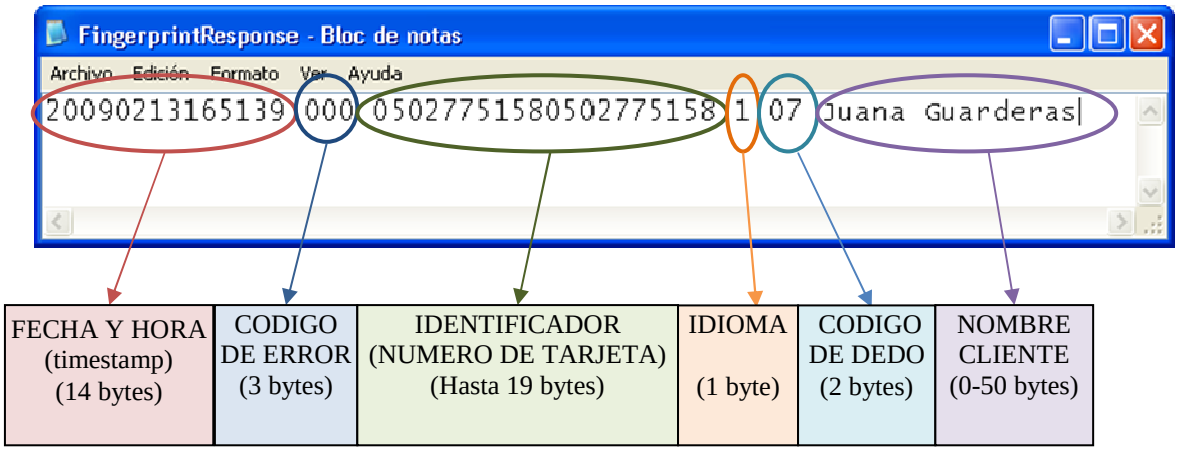


Figura 2. 38: Campos del archivo *FingerprintResponse*

Entre cada campo existe un byte de control representado por el espacio.

En la tabla 2.1 se indican los valores posibles para los códigos de error, mientras que en las tablas 2.2 y 2.3 se indican los valores para el idioma y dedo respectivamente. Estos valores de las tablas son los que se encuentran especificados en el módulo *MensajesRespuesta* de la aplicación.

Por motivos de control todos los eventos que se van generando en la aplicación son grabados en un archivo log denominado *LogHATMDactiar_aaaa_mm_dd* (aaaa corresponde al año, mm al mes y dd al día en que los eventos fueron generados).

CODIGO	ERROR
000	Cliente Verificado (OK)
501	No existe Cliente con ese ID.
502	No coincide la huella para ese Cliente.
503	Error de comunicación con la base de datos.
510	Error del Lector de Huellas.
511	Error de Extracción de la imagen de la Huella.
513	*Timeout* de la huella (Cliente tardó en colocar dedo sobre el lector).
514	*Timeout* del servidor (Servidor tarda en responder).
515	No se encuentra al servidor o la comunicación fue rechazada por el mismo.

Tabla 2. 1: Códigos de Error

CÓDIGO	IDIOMA
1	Español
2	Quechua
3	Inglés
0	Error

Tabla 2. 2: Códigos de Idioma

CÓDIGO	DEDO	MANO
01	Meñique	Izquierda
02	Anular	
03	Medio	
04	Índice	
05	Pulgar	
06	Pulgar	Derecha
07	Índice	
08	Medio	
09	Anular	
10	Meñique	

Tabla 2. 3: Códigos de Dedo

En la figura 2.39 se indica un ejemplo de un log de la aplicación.

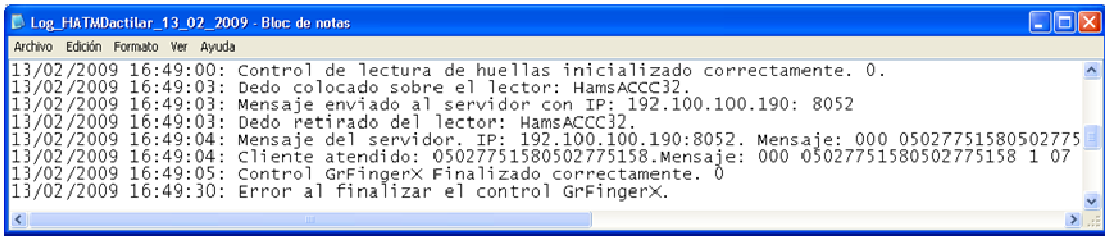


Figura 2. 39: Log de la aplicación HATMDactilar

En la figura 2.40 se puede observar la ventana de la aplicación, la misma que estará corriendo detrás de la aplicación propia del cajero y que permite la captura de la huella dactilar en el momento en que se necesite.

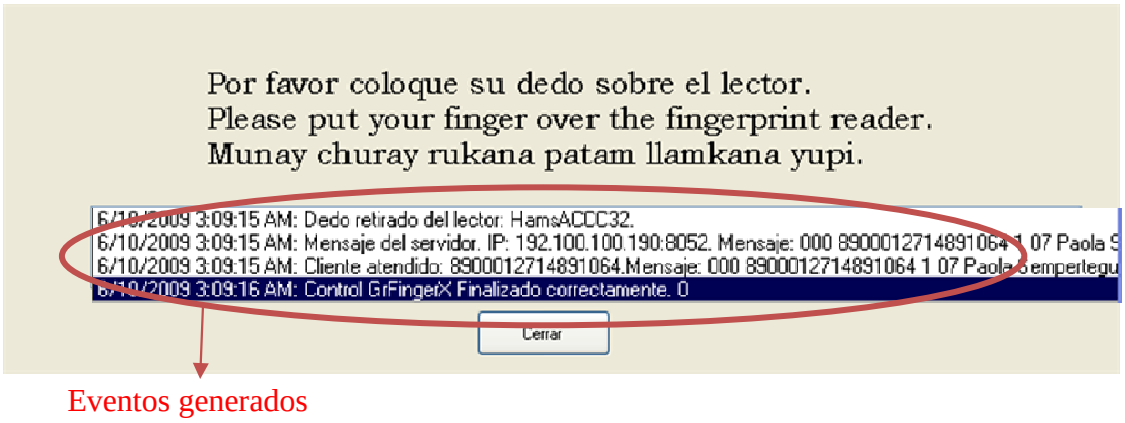


Figura 2. 40: Ventana de la aplicación HATMDactilar

2.4.2.2 Aplicación “SATMDactilar”

Esta aplicación es el servidor de autenticación, motivo por lo que el servidor en el cual correrá la aplicación debe ser accedido solo por usuarios autorizados para el manejo y administración del sistema. Al iniciar esta aplicación, se abre la ventana correspondiente al formulario principal de la aplicación, y el hilo que administra el *socket* que espera las conexiones de clientes arranca y solo se cierra una vez que la aplicación es cerrada. El formulario principal contiene un subformulario en el que se pueden observar todos los eventos que se han generado mientras corre la aplicación, como por ejemplo la conexión de un *Host Cajero*, el arribo de un mensaje, los errores generados entre otros; cabe indicar que este formulario siempre estará visible en el formulario principal y no se lo puede cerrar. En la figura 2.41 se tiene una imagen del servidor funcionando.

Los eventos generados además de ser mostrados en la ventana de la aplicación también se van guardando en un archivo log, llamado *LogSATMDactilar_aaaa_mm_dd* (aaaa corresponde al año, mm al mes y dd al día en que los eventos fueron generados).

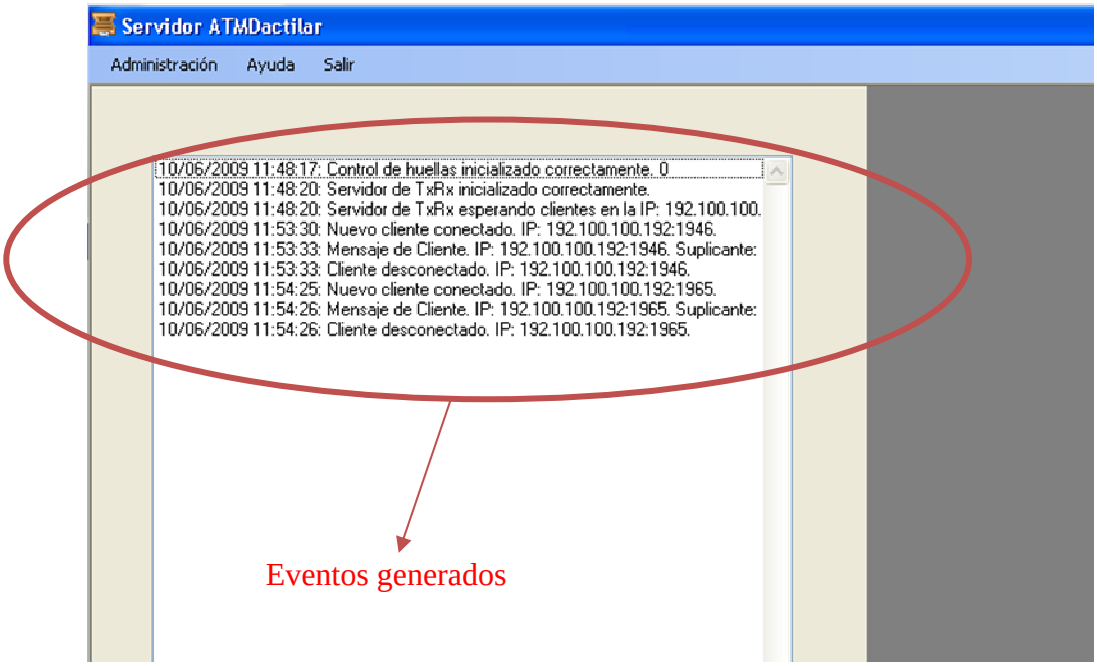


Figura 2. 41: Ventana de la aplicación SATMDactilar

En la parte superior de la ventana se encuentra una barra de menú con diferentes opciones, las cuales permiten al administrador del sistema cambiar parámetros y ver reportes; las opciones a las que se pueden acceder son:

- Generar reportes.
- Probar conexión con la base de datos.
- Cambiar los parámetros relacionados a la base de datos, como son la dirección o nombre del servidor de base de datos, el nombre de la base de datos, el nombre de Usuario de la base de datos y su contraseña.
- Verificar el estado del servidor (Clientes conectados).
- Cambiar los parámetros de comunicación como son la dirección IP del servidor y el puerto de escucha del *socket*.
- Visualizar los errores generados.

2.4.2.3 Aplicación "AATMDactilar"

Esta aplicación debe ser instalada en todos los *Host* de los *Usuarios Administrativos* que estén autorizados para enrolar clientes, actualizar clientes y/o administrar el sistema. El momento de iniciar la aplicación una pantalla de bienvenida solicita que un *Usuario Administrativo* coloque su dedo sobre el lector, para lo cual se activa e inicializa el lector de huellas dactilares. Una vez colocado el dedo sobre el lector, la huella es capturada y la aplicación procede a identificar al usuario. Si el resultado de la identificación es positivo se carga la interfaz correspondiente en base a los roles asignados a dicho usuario, caso contrario reinicializa el lector de huellas para capturar una nueva huella dactilar.

La aplicación consta de un formulario de inicio, un formulario principal que representa el formulario padre de otros 13 formularios que conforman la aplicación, dando un total de 15 formularios. Además en la parte superior del formulario principal se encuentra una barra de menú que permite al Usuario Administrativo seleccionar la tarea a realizar en base a los roles otorgados. Para cada tarea se abre un nuevo formulario (de los 13 formularios hijos), los mismos que pueden estar abiertos más de uno al mismo tiempo pero no pueden ser maximizados, permitiendo al Usuario Administrativo realizar distintas tareas a la

vez. Existe un formulario encargado de mostrar al Usuario Administrativo la imagen de la huella dactilar capturada en el lector y el puntaje con el que fue autenticado o no el dueño de dicha huella.

Dentro de las tareas que puede realizar un usuario administrativo están las siguientes:

- *Enrolar Clientes*: Para lo cual el usuario administrativo debe ingresar la información del cliente (Identificador de la tarjeta, nombre, lenguaje de preferencia y dedo seleccionado); posteriormente, la aplicación verifica que el cliente no se encuentre registrado y procede a capturar la huella del dedo seleccionado. Si la imagen capturada es de buena o media calidad el sistema solicita que el cliente coloque nuevamente su dedo para verificarlo; si la verificación es positiva, el cliente ha sido ingresado en la base de datos, caso contrario el usuario administrativo debe decidir si continuar o no con el registro de dicho cliente. Si la imagen capturada es de baja calidad el usuario administrativo es quien decide si continúa o no con el registro del cliente.
- *Verificar Clientes*: Mediante el ingreso del identificador de la tarjeta del cliente y su huella dactilar, se pueden verificar los datos de un cliente si éste es reconocido por el sistema.
- *Actualizar Clientes*: Esta tarea sirve para que el usuario administrativo pueda actualizar la información del cliente; como primera etapa se debe verificar la identidad del cliente que se desea actualizar mediante el identificador de su tarjeta y su huella dactilar, este proceso lo realiza automáticamente el sistema. Si la verificación es correcta se abre el formulario que permite actualizar los datos del cliente (dedo seleccionado y lenguaje de preferencia); si la verificación es incorrecta el usuario administrativo decide si va a continuar o no con la actualización. Para que el usuario administrativo pueda actualizar la información del cliente, éste debe ser autenticado nuevamente a través de su huella dactilar; una vez autenticado se actualiza la información del cliente y el usuario administrativo tiene la opción de actualizar o no la huella dactilar del cliente.

- *Ingresar Usuarios Administrativos*: Esta opción sirve para ingresar nuevos usuarios administrativos al sistema para lo cual el usuario administrativo debe ingresar la información correspondiente al nuevo usuario. Una vez que la aplicación comprueba que el nuevo usuario no se encuentra registrado en la base de datos, se captura la imagen de la huella dactilar del dedo seleccionado; si la imagen es de buena o media calidad, el nuevo usuario administrativo es ingresado y en caso de ser de mala calidad la imagen, el nuevo usuario no es registrado.
- *Actualizar Usuarios Administrativos*: Esta tarea permite actualizar la información almacenada tanto del usuario administrativo que está usando la aplicación como la información de algún otro usuario administrativo. El proceso es similar a la actualización del cliente ya que se tiene en primera instancia que buscar la información del usuario mediante su número de identificación, se procede a realizar una verificación de identidad a través de su huella dactilar y posterior a eso se puede modificar la información del usuario. Antes de guardar la información se debe decidir si se desea actualizar la huella dactilar almacenada en la base de datos. Para confirmar que la actualización es legal el usuario administrativo que está actualizando la información debe confirmar el proceso mediante su huella dactilar.
- Ver Parámetros de Configuración: Con esta opción se puede ver y cambiar los parámetros relacionados a la base de datos.
- Añadir/Buscar Cajeros: Mediante esta opción se puede agregar o buscar en la base de datos los identificadores e información de los cajeros que están integrados en la red.

En la aplicación se tienen definidos 3 roles para los usuarios administrativos. Según el rol o roles asignados a un usuario, éste tiene un nivel de privilegios para acceder a las diferentes opciones de la aplicación. Los roles y las acciones que se encuentran creadas en la aplicación son:

- *Enrolar*: El usuario administrativo puede solamente enrolar clientes.
- *Actualizar*: El usuario administrativo puede solamente actualizar clientes.

- *Administrar*: El usuario administrativo puede ingresar usuarios administrativos y actualizarlos, además puede cambiar parámetros de configuración, buscar o añadir información sobre los cajeros automáticos relacionados a la cooperativa a la que pertenece.

2.4.3 BASE DE DATOS

La base de datos fue implementada en *Microsoft SQL Server* 2005 bajo licenciamiento de desarrollo. Para la creación de la base de datos se ha desarrollado un *script* (que se encuentra adjunto en el Anexo G) basado en el lenguaje *Transact-SQL*, cuyo contenido permite la creación de todas las tablas, relaciones y procedimientos almacenados requeridos por la aplicación en base a los modelos previamente desarrollados.

Para los permisos de acceso a la base de datos se ha considerado brindar la seguridad a nivel de SQL, es decir mediante la autenticación de *SQL Server*, creando un inicio de sesión para que el usuario se conecte al motor de base de datos indicando el nombre del inicio de sesión y la contraseña. A nivel de base de datos, el usuario de este inicio de sesión tiene la función de administrador (*sysadmin*), la misma que concede todos los permisos para realizar cualquier actividad de administración de usuarios y bases de datos.

En cuanto a la estructura de la base de datos, está compuesta por las relaciones ilustradas en la sección 2.2.2 en el modelo de datos. Para realizar consultas a la base de datos se emplearán procedimientos almacenados que se encuentran detallados en Diccionario de Datos en el anexo F.

2.5 DIMENSIONAMIENTO DE SERVIDORES

2.5.1 SERVIDOR DE APLICACIÓN

La principal función de este servidor es la de “correr” la aplicación SATMDACTILAR.

2.5.1.1 Diseño del Sistema y Requerimientos

En el subcapítulo 2.1 se encuentra el diseño de la aplicación y en la sección 2.1.2 se enumeran los requerimientos de la misma. A partir del diseño anterior y del

desempeño que se desea obtener del sistema, es importante considerar los siguientes requerimientos del sistema:

- *Protocolos IP y TCP:* La aplicación en el servidor esperará por los mensajes provenientes del cajero automático para la autenticación de clientes; esta comunicación empleará los protocolos IP y TCP por lo tanto además será necesario que el servidor tenga un adaptador de red.
- *.NET Framework:* La aplicación desarrollada Visual Basic .NET requerirá del paquete .Net Framework.
- *Crystal Reports:* Para los reportes del estado de la aplicación y de históricos de uso, se utilizará esta herramienta para la generación y presentación de los mismos.
- *Túneles VPN:* Por políticas de seguridad se manejará un túnel VPN para cada comunicación entre un Host Cajero y el Servidor.

2.5.1.2 Modelo de Dimensionamiento

- *Tipo de aplicación:* Como se indicó anteriormente, es una aplicación que se comunica a través de los protocolos TCP/IP con los Host Cajeros a través de túneles VPN. La aplicación atiende las peticiones de autenticación de clientes que recibe a través de un puerto definido y consulta a la base de datos la información correspondiente para realizar la autenticación. Por lo tanto es importante definir el puerto de escucha de las peticiones, así como el nombre y ubicación de la base de datos a la que requiere conectarse la aplicación; toda esta información se maneja a través del registro del sistema.
- *Tipos de Usuarios:* Se tienen dos tipos de usuarios, el uno corresponde al cliente que enviará las peticiones de autenticación al servidor y el otro corresponde al usuario administrativo que tiene acceso al mismo para observar los reportes y cambiar parámetros de configuración.
- *Tipo de Interfaz:* La aplicación fue desarrollada en Visual Basic .NET y se presenta en varios formularios que permiten al administrador observar las

transacciones realizadas y el estado del servidor, así como también cambiar algunos de los parámetros de configuración.

- *Cantidad de Información procesada:* Para analizar este punto es importante conocer el formato del mensaje que recibe el Servidor por parte de un Host Cajero el mismo que se encuentra ilustrado en la figura 2.42.

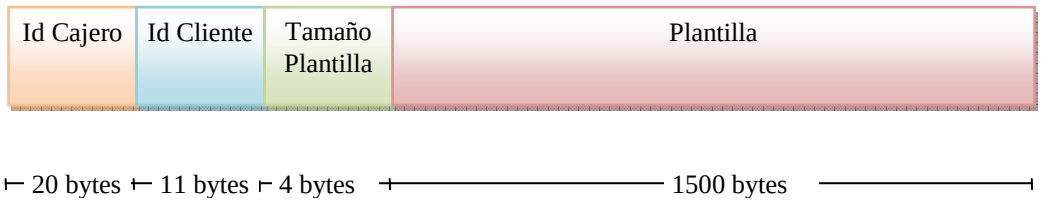


Figura 2. 42: Formato mensaje cliente

Los campos correspondientes a los identificadores del cajero y del cliente, y el campo correspondiente al tamaño de la plantilla añaden un byte de control para ser procesados en el lado del servidor. El tamaño de la plantilla es variable ya que depende de la fisiología del dedo que es única para cada persona, además depende del ángulo de ubicación del dedo sobre el lector de huella dactilar, del tipo de lector de huella dactilar entre otros factores.

En base a las pruebas realizadas, el tamaño máximo de la plantilla de la huella que se podría llegar a obtener es de 1200 bytes aproximadamente, el mismo que se duplica al ser agregados los caracteres de control. Por este motivo se aplica un algoritmo de compresión basado en un cambio de base, como se detalla en la sección de implementación, y como resultado se logra obtener una reducción del 40% de la plantilla en promedio; en consecuencia el tamaño máximo del campo plantilla que puede ser enviado dentro del mensaje al servidor será de:

$$1200 bytes \times 2 \times 0.6 = 1440\ bytes$$

La tabla 2.4 muestra el tamaño de cada campo y de todo el mensaje en el peor de los casos que se da cuando se tiene una plantilla grande.

Campo	Bytes
Id Cajero	21
Id Cliente	21
Tamaño Plantilla	5
Plantilla	1440
Total	***1487***

Tabla 2. 4: Tamaño mensaje cliente

Así se tiene que cada vez que llega un mensaje al servidor es necesario procesar en el peor de los casos al menos 1487 bytes por cada cliente. Como el servidor debe manejar simultaneidad en el procesamiento de requerimientos, esta simultaneidad dependerá del número de host conectados en un momento dado en el servidor. El peor de los casos se dará cuando todos los cajeros tengan una solicitud que debe ser atendida, por lo tanto la máxima cantidad de información procesada cuando se reciben mensajes simultáneos por parte de todos los host Cajeros es la siguiente:

$$I = 1487 \times N$$

Donde N es el número de cajeros existentes en el sistema e I es la cantidad de información en bytes.

- *Número de usuarios promedio:* El número de usuarios promedio que se estima se tendrá en el sistema es de 2000 por día; aunque realmente el valor más importante que se debe tener en cuenta para la mayoría de los cálculos es la simultaneidad de uso de la aplicación, es decir el número de clientes realizando una transacción al mismo tiempo. En el peor de los casos el máximo número de usuarios que se tendrán simultáneamente se da cuando todos los cajeros conectados al servidor estén realizando una transacción. Existen factores que podrían llegar a modificar el número promedio de usuarios como el cobro del bono solidario, cobro de

quincenas, festividades, feriados, fines de semana, conciertos, eventos públicos, etc.

- *Frecuencia y tiempo de las transacciones:* La frecuencia de las transacciones dependerá de factores como los indicados en el punto anterior.

  Para determinar el tiempo (la duración) de una transacción es necesario realizar algunas pruebas de peticiones secuenciales y determinar el tiempo máximo que podría demorar una transacción. Con esta información (frecuencia y tiempo) se puede estimar la frecuencia que requiere el procesador del servidor.

2.5.1.3 Requerimientos de Hardware

Basados en el modelo de dimensionamiento del punto anterior se pueden determinar los requerimientos de hardware necesarios para obtener el mejor desempeño del servidor. Por el tipo de servicio, el principal recurso a tomar en cuenta son el procesador y el adaptador de red, ya que las solicitudes provenientes del cajero deben ser atendidas lo más pronto posible.

- *Procesador:* En el punto anterior se indicó que la frecuencia y tamaño de una transacción permiten estimar la frecuencia del procesador. Por tanto, para determinar el tiempo de una transacción, se realizaron varias pruebas de peticiones secuenciales de autenticación de clientes desde un host cajero a un servidor. En la tabla 2.5 se tienen las características de los equipos usados en estas pruebas.

  Los resultados obtenidos del tiempo que toma cada uno de los eventos que se dan en el proceso de autenticación se encuentran en la tabla 2.6

  Así se tiene que el tiempo promedio que demora el servidor en procesar el mensaje recibido (desencriptar y leer contenido) es de 8.65 milisegundos, el tiempo promedio que le toma verificar la huella es de 5.9 milisegundos y el tiempo que demora en procesar el mensaje de respuesta (armar y encriptar) es de 0.4 milisegundos en promedio; dando como resultado que el tiempo promedio que demora el servidor en una solicitud es de 14.95 milisegundos. En el peor de los escenarios tomando en cuenta los tiempos

máximos obtenidos en cada uno de los procesos (desencripción, verificación y encripción) se tendría que el tiempo total que demoraría una solicitud sería de $16 + 16 + 16 = 48\ [ms]$ (Se consideran despreciables los tiempos de propagación y transmisión ya que son mucho menores al tiempo que toma procesar cada transacción en el ámbito de una red LAN).

Para los cálculos posteriores se tomará en cuenta el tiempo máximo aproximado que podría demorar una solicitud (48 milisegundos); de aquí se tiene que aproximadamente el número de ciclos que requiere cada una es de:

$$NumCiclos_{solicitud} = FrecuenciaProcesador_{Prueba} * Tiempo_{solicitud}$$

$$NumCiclos_{solicitud} = \frac{3 * 10^9 ciclos}{1\ segundo} * \frac{0.048\ segundos}{1\ solicitud} = 0.144 * 10^9$$

DETALLE		SERVIDOR	HOST CAJERO
Sistema Operativo		Windows XP Professional	Windows XP Professional
Versión		2002	2002
Procesador	**Fabricante**	Intel Core 2	Intel Pentium 4
	Frecuencia	1.86 GHz	1.8 GHz
	Número de Núcleos:	2	1
RAM	**Tamaño**	1 GB	512 MB
Red	**Adaptador**	Intel PRO/100 VE	VIA PCI 10/100 Mbps
	Tipo de Conexión	100BaseT	100BaseT

Tabla 2. 5: Características Host Cajero y Servidor de prueba para dimensionamiento

Tiempos / Mensaje No.	Procesar[23] Mensaje RX (ms)	Verificar Huella (ms)	Procesar Mensaje TX (ms)	Total (ms)
1	0	15	0	15
2	0	16	0	16
3	0	0	0	0
4	16	0	0	16
5	0	15	0	15
6	0	16	0	16
7	15	0	0	15
8	15	0	0	15
9	16	0	0	16
10	16	0	0	16
11	16	0	0	16
12	0	16	0	16
13	16	0	0	16
14	0	16	0	16
15	16	0	0	16
16	15	0	0	15
17	0	16	0	16
18	16	0	16	32
19	0	16	0	16
20	16	0	0	16
21	15	0	0	15
22	0	16	0	16
23	0	0	0	0

[23] Los tiempos de cero milisegundos se producen porque ese es el límite inferior del método de medición empleado.

24	0	0	0	0
25	16	16	0	32
26	16	0	0	16
27	15	0	0	15
28	0	0	0	0
29	0	16	0	16
30	16	0	0	16
31	0	16	0	16
32	16	0	0	16
33	16	0	0	16
34	0	15	0	15
35	0	16	0	16
36	15	0	0	15
37	0	15	0	15
38	16	0	0	16
39	16	0	0	16
40	16	0	0	16
Promedio	*8.65*	*5.9*	*0.4*	*14.95*

Tabla 2. 6: Tiempo promedio de una transacción

Tomando en cuenta un escenario en el que la aplicación sea utilizada en cooperativas pequeñas, se espera que el número máximo de cajeros que se comuniquen con el servidor sea de 100. Suponiendo que todos estos cajeros envian simultáneamente mensajes al servidor, y que cada cajero debe recibir una respuesta en menos de 30 segundos, la frecuencia requerida del procesador está dada por:

$$FrecRequerida_{procesador} \geq NumCiclos_{solicitud} * NumSolicitudes_{simultaneas}$$

$$FrecuenciaRequerida_{procesador} \geq \frac{0.144 * 10^9 ciclos}{solicitud} * \frac{100\ solicitudes}{30\ segundos}$$

$$FrecuenciaRequerida_{procesador} \geq 0.48 * 10^9 \frac{ciclos}{segundo}$$

$$FrecuenciaRequerida_{procesador} \geq 0.48\ GHz$$

Así se tiene que la frecuencia del procesador debe ser mayor a 480 MHz, considerando un factor de un 10% adicional se consigue:

$$480 * 1.1 = 528\ MHz$$

El procesador debe tener una frecuencia mayor a 528 MHz para conseguir el rendimiento esperado.

- *Adaptador de Red:* Este elemento es importante a considerar dado que la comunicación entre el servidor y los cajeros se lo da a través de la red LAN. Generalmente las tarjetas de red vienen incorporadas en la placa madre del ordenador conectadas a través de un bus PCI. El puerto necesario para la comunicación es RJ-45 donde la velocidad que debe soportar dependerá de la red, pero habitualmente será de 100 Mbps.
- *Memoria:* Se recomienda que la memoria sea de al menos 512 MB.

2.5.1.4 Opciones de Configuración

Un solo servidor con las características detalladas anteriormente será suficiente para el buen desempeño de la aplicación, no es necesario pensar en un cluster de computadoras.

2.5.1.5 Servidor recomendado

En base al análisis que se ha detallado en las secciones anteriores el hardware del servidor recomendado se especifica en la tabla 2.7.

Tipo de Procesador	*Pentium IV* en adelante
Velocidad de Procesador	600 MHz en adelante
Memoria RAM	512 MB
Disco Duro	Arquitectura: DAS Protocolo: SCSI o SATA
Sistema Operativo	*Windows Server 2003*
Tarjeta de Red	Interfaz RJ45 para conexión 100BaseT

Tabla 2. 7: Servidor Recomendado

2.5.2 SERVIDOR DE BASE DE DATOS

En este servidor se encontrará la base de datos a la que se conectan tanto el Host Administrativo como el Servidor de la aplicación para realizar las respectivas consultas, registro o actualización de datos a través de procedimientos almacenados.

2.5.2.1 Diseño del Sistema y Requerimientos

Como se indicó en la sección 2.2 se ha utilizado el modelo relacional para el diseño de la base de datos y ésta ha sido realizada en lenguaje SQL usando *Microsoft SQL Server* para la implementación; por este motivo los requerimientos de este servidor son los siguientes:

- *Microsoft SQL Server :* Es necesaria esta herramienta en la edición 2005 bajo cualquier licenciamiento soportado en base al tipo de negocio en donde se vaya a implementar el sistema. *Requerimientos de Microsoft SQL Server 2005:* Para todas las de versiones de *SQL Server* edición 2005 se tiene como requerimientos mínimos los indicados en la tabla 2.8.

  Se recomienda que la velocidad del procesador sea de 1 GHz en adelante y la memoria RAM mayor a 1 GB para un mejor desempeño del servidor.

- *Adaptador de Red:* Debido a que las consultas al servidor se las harán a través de la red de área local existente en la institución donde será implementado el sistema, el servidor requiere de un adaptador de red que soporte las características de la red local.

Tipo de Procesador	*Pentium III*
***Velocidad* de Procesador**	500 MHz
Memoria RAM	512 MB
Disco Duro	350 MB (Instalación) + 390 MB (BDD de muestra)
Sistema Operativo	*Windows XP Home Edition* *Windows XP Professional Edition* *Windows Server 2003 Web Edition*
Otros	Para *Windows XP: Service Pack 2* en adelante Para *Windows Server: Service Pack 1* en adelante
Tarjeta de Red	Interfaz RJ45 para conexión 100BaseT

Tabla 2. 8: Requerimientos Mínimos de *SQL Server 2005 Standard Edition*

2.5.2.2 Modelo de Dimensionamiento

- *Tipo de aplicación:* En este servidor se encontrará la base de datos de la aplicación. Esta base de datos estará implementada en *Microsoft SQL Server* , y el acceso a la misma se realiza mediante autenticación de *SQL Server*.

  La base de datos dispone de un conjunto de procedimientos almacenados los cuales permiten a la aplicación SATMDactilar y HATMDactilar interactuar con la misma, proveyendo seguridad y control en la información que se consulta o ingresa a y desde la base de datos.

- *Tipos de Usuarios:* El acceso de un usuario a la base de datos será mediante la autenticación de *SQL Server*, y el acceso a la información será mediante procedimientos almacenados. Se tendrá un usuario que permite la interconexión de las aplicaciones SATMDactilar y HATMDactilar con la base de datos y se tendrán usuarios para la administración, operación y mantenimiento de la base de datos.

- *Cantidad de Información procesada:* En este punto es necesario considerar la cantidad de información que será almacenada en la base de datos determinando el tamaño de cada una de las tablas de la misma.

El tamaño de cada tabla depende del número de filas y columnas, de la característica de los tipos de datos contenidos (su naturaleza y si el dato es fijo o variable), de ciertos bytes adicionales reservados, entre otros.

Para determinar el tamaño de cada tabla se emplearán las recomendaciones de la ayuda de *Microsoft SQL Server* [30] para tablas que tienen un único índice agrupado.

Un índice es una estructura asociada a una tabla que permite encontrar rápidamente una información específica. El índice está conformado por claves, las cuales se construyen a partir de una o más columnas de la tabla, y por punteros que señalan a la ubicación de la información. Existen dos tipos de índices: los agrupados y los no agrupados.

Los índices agrupados permiten ordenar las filas de una tabla en base a los valores de las claves y se encuentran organizados en árboles-B. Un árbol-B se encuentra conformado por varias páginas conocidas como nodos de índice. Las páginas de un árbol-B se encuentran distribuidas en uno o más niveles, el nodo del nivel superior se conoce como *nodo raíz*, los nodos del nivel inferior se denominan nodos hoja y corresponden a las páginas de los datos, y los nodos de los niveles intermedios se los conoce como nodos intermedios. La estructura de un árbol-B se puede ver en la figura 2.43.

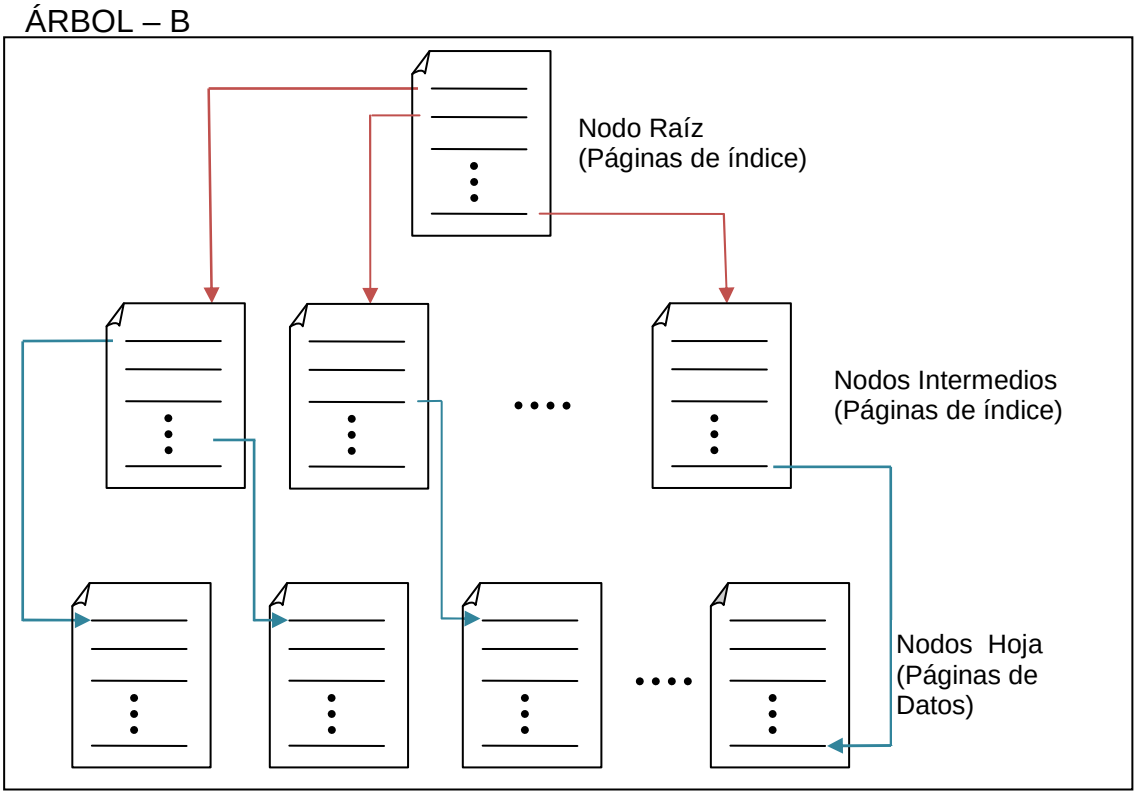


Figura 2. 43: Estructura árbol-B de un índice agrupado

Los índices no agrupados tienen una organización similar al agrupado pero no contiene a los datos dentro del índice, de tal manera que sus nodos hoja están conformados por páginas de índice y no de datos, y no permite clasificar la información en base a las claves del índice.

Las tablas implementadas en la base de datos están organizadas en índices agrupados; a continuación se indican los cálculos para estimar el espacio necesario para almacenar los datos en el nivel de hoja del índice agrupado y el espacio para almacenar la información de índice dentro del índice agrupado.

Para calcular el espacio para el almacenamiento de los datos a nivel de hoja se calculan los siguientes datos para cada tabla:

- *Número de filas (Num_Filas):* Corresponde al número de filas que se espera habrá en la tabla.
- *Número de columnas (Num_Cols):* Se determina el número total de columnas presentes en la tabla (de longitud fija y variable).

- *Tamaño total de datos de longitud fija (Tam_Fijo_Datos)*: Este valor es representado en bytes y corresponde a la sumatoria del tamaño de las columnas de longitud fija en la tabla; estos datos pueden ser del tipo *int, char, datetime, binary, bit* entre otros.
- *Número de columnas de longitud variable (Num_Cols_Var):* Corresponde al número de columnas cuyos datos son de longitud variable, como por ejemplo del tipo varchar, varbinary entre otros.
- *Tamaño máximo de todas las columnas de longitud variable (Tam_Max_Var):* Representa la sumatoria del tamaño máximo de todas las columnas de longitud variable de la tabla. Este valor está dado en bytes.
- *Mapa de bits NULL (Null_Bit_Map):* Corresponden a los bits utilizados para la administración de la nulabilidad[24] de las columnas de la tabla. Este valor se obtiene de la siguiente expresión.

$$Nul\_Bit\_Map = 2 + ((Num\_Cols + 7)/8)$$

El valor anterior se redondea al entero inferior más próximo.

- *Tamaño total de datos de longitud variable (Tam_Datos_Var):* Representa el tamaño máximo de las columnas de longitud variable con dos bytes adicionales por cada una de estas columnas. Su valor se calcula a partir de la siguiente expresión, asumiendo que estas columnas van a ser ocupadas al 100%.

$$Tam\_Datos\_Var = 2 + (Num\_Cols\_Var \times 2) + Tam\_Max\_Var$$

- *Tamaño de fila (Tam_Fila):* Representa el tamaño total que ocuparía una fila en la base de datos, su valor está dado en bytes y se obtiene a partir de la siguiente ecuación:

$$Tam\_Fila = Tam\_Fijo\_Datos + Tam\_Datos\_Var + Null\_Bit\_Map + 4$$

[24] Nulabilidad: Término que hace referencia cuando un campo puede o no tener valores nulos.

- *Filas por Página (Filas_Pagina):* Corresponde al número de filas que se puede tener por página. Se debe considerar el tamaño de cada fila y el tamaño de una página (1 página = 8096 bytes). Además se deben agregar dos bytes al tamaño de la fila que corresponden a la entrada de la fila en la página. Esta expresión se indica a continuación:

$$Filas\_Pagina = \frac{8096}{(Tam\_Fila + 2)}$$

El valor anterior se redondea al entero inferior más próximo.

- *Filas libres reservadas por página (Filas_Libres_Reserv):* Representa el número de filas reservadas por página para un futuro crecimiento según el factor de relleno[25].

$$Filas\_Libres\_Reserv = 8096 \times \frac{(100 - Factor\_Relleno)}{100} \over (Tam\_Fila + 2)$$

Este valor se redondea al entero inferior más próximo.

- *Número de páginas (Num_Paginas):* Representa el número de páginas requeridas para el almacenamiento de todas las filas.

$$Num\_Paginas = \frac{Num\_Filas}{(Filas\_Pagina - Filas\_Libres\_Reserv)}$$

Este valor es redondeado al entero superior más próximo.

- *Espacio requerido en el nivel hoja (Espacio_Hoja):* Este valor representa el espacio requerido para almacenar los datos en el nivel de hoja y se representa en bytes.

$$Espacio\_Hoja = 8192 \times Num\_Paginas$$

Para estimar el espacio requerido para almacenar información del índice se calculan los siguientes valores por cada tabla:

[25] Factor de relleno: Representa el porcentaje de espacio libre que se asigna cuando se crea una nueva página del índice

- *Número de columnas clave (Num_Cols_Clave):* Representa el total de número de columnas clave de la tabla.
- *Tamaño total de columnas clave de longitud fija (Tam_Fijo_Datos_Clave):* Se representa en bytes e indica la sumatoria de los tamaños de las columnas clave de longitud fija.
- *Número de columnas clave de longitud variable (Num_Cols_Clav_Var):* Representa el número de columnas clave cuyos datos son del tipo variable como varchar, varbinary, entre otros.
- *Tamaño máximo de columnas clave de longitud variable (Tam_Max_Var_Clave):* Este valor es representado en bytes e indica la sumatoria del tamaño de las columnas clave cuyos datos son del tipo variable.
- *Mapa de bits NULL (Null_Bit_Map_Clave)*: Corresponde a los bits que permiten administrar la nulabilidad de las columnas claves, solo se toman en cuenta las columnas clave que admiten valores nulos. Este valor está dado por la siguiente expresión:

$$Null\_Bit\_Map\_Clave = 2 + \frac{(Num\_Cols\_Clave\_Admiten\_Null + 7)}{8}$$

  De este valor solo se toma en cuenta la parte entera.

- *Tamaño total de datos de longitud variable (Tam_Datos_Var_Clave):* Este valor está dado en bytes y representa el tamaño máximo de las columnas clave de longitud variable. Su valor se determina mediante la siguiente expresión:

$$Tam\_Datos\_Var\_Clave = 2 + (Num\_Cols\_Clav\_Var \times 2) + Tam\_Max\_Var\_Clave$$

- *Tamaño de una fila de índice (Tam_Fila_Indice):* Este valor se representa en bytes e indica el tamaño de una fila de índice, el cual se calcula a partir de la siguiente expresión

$$Tam\_Fila\_Indice = Tam\_Fijo\_Datos\_Clave + Tam\_Datos\_Var\_Clave + Null\_Bit\_Map\_Clave + 1 + 6$$

Se agrega un byte para el encabezado de una fila índice y 6 bytes para el puntero de identificador de una página secundaria.

- *Número de filas de índice por página (Filas_Indice_Pagina):* Este valor es calculado en base al tamaño de una fila de índice y el tamaño de una página (1 página = 8096 bytes) como se indica a continuación; se toma en cuenta dos bytes adicionales al tamaño de la fila de índice para la entrada de la fila en la página:

$$Filas\_Indice\_Pagina = \frac{8096}{(Tam\_Fila\_Indice + 2)}$$

Este valor es redondeado hacia el entero inferior más próximo.

- *Número de niveles del índice (Niveles_Indice):* Este valor representa el número de niveles que puede tener un índice en base al número total de filas en nivel de hoja y el número de filas índice por página; la expresión utilizada es:

$$Niveles\_Indice = 1 + \log_{Filas\_Indice\_Pagina}\left(\frac{Num\_Ftlas}{Filas\_Indice\_Pagina}\right)$$

Este valor se redondea al número entero más próximo.

- *Número de páginas del índice (Num_Paginas_Indice):* Este valor está representado por la sumatoria del número de páginas de todos los niveles y se lo calcula de la siguiente manera:

$$Num\_Paginas\_Indice = \sum_{i=1}^{Niveles}\left(\frac{Num\_Filas}{Filas\_Indice\_Pagina^{i}}\right)$$

- *Espacio requerido para almacenar el índice (Espacio_Indice):* Se representa en bytes y su valor indica el tamaño requerido para el almacenamiento del índice tomando en cuenta que el tamaño para

almacenar 1 página es de 8192 bytes; su cálculo se muestra en la siguiente expresión:

$$Espacio\_Indice = 8192 \times Num\_Paginas\_Indice$$

En base a las consideraciones anteriores los valores obtenidos para cada tabla fueron los siguientes:

- *Tabla:* Cajero_tbl.

  Consideraciones: Se espera que el número máximo de cajeros que se registrará para todas las cooperativas sea de 100 y que el factor de relleno (Factor_Relleno) sea igual a 100.

  La tabla 2.9 indica las características de cada columna de la tabla Cajero_tbl; y a continuación en las tablas 2.10 y 2.11 se encuentran el resultado del espacio requerido para el nivel de hoja y para el índice respectivamente.

Columnas	**Tipo de Dato**	**Tamaño (bytes)**	**Admiten NULL**
IdCaj	Varchar	20	No
UbiCaj	Varchar	75	No
IdCoop	varchar	20	No
EstadoCaj	varchar	10	No

Tabla 2. 9: Tamaño de columnas en tabla Cajero_tbl

Num_Filas[Filas]	100
Num_Cols [columnas]	4
Tam_Fijo_Datos [bytes]	0
Num_Cols_Var [columnas]	4
Tam_Max_Var [bytes]	125
Null_Bit_Map [bytes]	3
Tam_Datos_Var [bytes]	135
Tam_Fila [bytes]	142

Filas_Pagina[Filas]	56
Filas_Libres_Reserva[Filas]	0
Num_Paginas [páginas]	2

Espacio_Hoja	***16384***

Tabla 2. 10: Espacio requerido para tabla Cajero_tbl en nivel de hoja

Num_Cols_Clave [columnas]	1
Tam_Fijo_Datos _Clave[bytes]	0
Num_Cols_Var _Clave[columnas]	1
Tam_Max_Var_Clave [bytes]	20
Null_Bit_Map_Clave [bytes]	0
Tam_Datos_Var_Clave [bytes]	24
Tam_Fila_Indice [bytes]	31
Filas_Indice_Pagina[Filas]	245
Niveles_Indice [niveles]	1
Num_Paginas_Indice [Páginas]	1

Espacio_Indice	***8192***

Tabla 2. 11: Espacio requerido para índice en tabla Cajero_tbl

De las tablas anteriores se tiene que el tamaño de la tabla Cajero_tbl sería:

$$Tam\_Tabla = Espacio\_Hoja + Espacio\_Indice = 24576\ bytes$$

- *Tabla:* Cliente_tbl.

  Consideraciones: Se espera que el número máximo de clientes que se registrará sea de 20000 y que el factor de relleno (Factor_Relleno) sea igual a 100.

  La tabla 2.12 indica las características de cada columna de la tabla Cliente_tbl; y a continuación en las tablas 2.13 y 2.14 se encuentran

el resultado del espacio requerido para el nivel de hoja y para el índice respectivamente.

Columna	Tipo de Dato	Tamaño (bytes)	Admite NULL
IdCli	varchar	20	No
NomCli	varchar	50	No
IdCoop	varchar	20	No
IdiomaCli	varchar	15	No
IdUA	varchar	13	No

Tabla 2. 12: Tamaño de columnas en tabla Cliente_tbl

Num_Filas [Filas]	20000
Num_Cols [columnas]	5
Tam_Fijo_Datos [bytes]	0
Num_Cols_Var [columnas]	5
Tam_Max_Var [bytes]	118
Null_Bit_Map [bytes]	3
Tam_Datos_Var [bytes]	130
Tam_Fila [bytes]	137
Filas_Pagina[Filas]	58
Filas_Libres_Reserva[Filas]	0
Num_Paginas [páginas]	345

Espacio_Hoja	***2826240***

Tabla 2. 13: Espacio requerido para tabla Cliente_tbl en nivel de hoja

Num_Cols_Clave [columnas]	1
Tam_Fijo_Datos _Clave[bytes]	0
Num_Cols_Var _Clave[columnas]	1
Tam_Max_Var_Clave [bytes]	20
Null_Bit_Map_Clave [bytes]	0
Tam_Datos_Var_Clave [bytes]	24
Tam_Fila_Indice [bytes]	31

Filas_Indice_Pagina[Filas]	245
Niveles_Indice [niveles]	2
Num_Paginas_Indice [Páginas]	82

Espacio_Indice	***671744***

Tabla 2. 14: Espacio requerido para índice en tabla Cliente_tbl

De los datos anteriores se tiene que el tamaño de la tabla Cliente_tbl sería de:

$$Tam\_Tabla = Espacio\_Hoja + Espacio\_Indice = 3497984\ bytes$$

- *Tabla:* Cooperativa_tbl.

  Consideraciones: Se espera máximo unas 10 cooperativas y que el factor de relleno (Factor_Relleno) sea igual a 100.

  La tabla 2.15 indica las características de cada columna de la tabla Cooperativa_tbl; y a continuación en las tablas 2.16 y 2.17 se encuentran el resultado del espacio requerido para el nivel de hoja y para el índice respectivamente.

Columna	**Tipo de Dato**	**Tamaño (bytes)**	**Admite NULL**
IdCoop	varchar	20	No
NomCoop	varchar	50	No

Tabla 2. 15: Tamaño de columnas en tabla Cooperativa_tbl

Num_Filas	10
Num_Cols [columnas]	2
Tam_Fijo_Datos [bytes]	0
Num_Cols_Var [columnas]	2
Tam_Max_Var [bytes]	70
Null_Bit_Map [bytes]	3
Tam_Datos_Var [bytes]	76

Tam_Fila [bytes]	83
Filas_Pagina[Filas]	95
Filas_Libres_Reserva[Filas]	0
Num_Paginas [páginas]	1

Espacio_Hoja	***8192***

Tabla 2. 16: Espacio requerido para tabla Cooperativa_tbl en nivel hoja

Num_Cols_Clave [columnas]	1
Tam_Fijo_Datos _Clave[bytes]	0
Num_Cols_Var _Clave[columnas]	1
Tam_Max_Var_Clave [bytes]	20
Null_Bit_Map_Clave [bytes]	0
Tam_Datos_Var_Clave [bytes]	24
Tam_Fila_Indice [bytes]	31
Filas_Indice_Pagina[Filas]	245
Niveles_Indice [niveles]	0
Num_Paginas_Indice [Páginas]	1

Espacio_Indice	***8192***

Tabla 2. 17: Espacio requerido para índice en tabla Cooperativa_tbl

En base a esta información se tiene que el tamaño de la tabla Cooperativa_tbl puede llegar a ser de:

$$Tam\_Tabla = Espacio\_Hoja + Espacio\_Indice = 16384\ bytes$$

- *Tabla:* Huella_tbl.

  Consideraciones: Se espera 20000 huellas correspondientes a los Clientes y 100 huellas correspondientes a los Usuarios Administrativos. Además se maneja un factor de relleno (Factor_Relleno) igual a 100.

La tabla 2.18 indica las características de cada columna de la tabla Huella_tbl; y a continuación en las tablas 2.19 y 2.20 se encuentran el resultado del espacio requerido para el nivel de hoja y para el índice respectivamente.

Columna	**Tipo de Dato**	**Tamaño (bytes)**	**Admite NULL**
IdHuella	int	4	No
PlanHuella	varbinary	1500	No
TamHuella	int	4	No
DedoHuella	int	4	No

Tabla 2. 18: Tamaño de columnas en tabla Huella_tbl

Num_Filas	20100
Num_Cols [columnas]	4
Tam_Fijo_Datos [bytes]	12
Num_Cols_Var [columnas]	1
Tam_Max_Var [bytes]	1500
Null_Bit_Map [bytes]	3
Tam_Datos_Var [bytes]	1504
Tam_Fila [bytes]	1523
Filas_Pagina[Filas]	5
Filas_Libres_Reserva[Filas]	0
Num_Paginas [páginas]	4020

Espacio_Hoja	***32931840***

Tabla 2. 19: Espacio requerido para tabla Huella_tbl en nivel índice

Num_Cols_Clave [columnas]	1
Tam_Fijo_Datos _Clave[bytes]	4
Num_Cols_Var _Clave[columnas]	0
Tam_Max_Var_Clave [bytes]	0
Null_Bit_Map_Clave [bytes]	0
Tam_Datos_Var_Clave [bytes]	0

Tam_Fila_Indice [bytes]	11
Filas_Indice_Pagina[Filas]	622
Niveles_Indice [niveles]	2
Num_Paginas_Indice [Páginas]	33

Espacio_Indice	***270336***

Tabla 2. 20: Espacio requerido para índice en tabla Huella_tbl

De aquí se tiene que la tabla Huella_tbl podría alcanzar un tamaño de:

$$Tam\_Tabla = Espacio\_Hoja + Espacio\_Indice = 33202176\ bytes$$

- *Tabla:* HuellaUsuario_tbl.

  Consideraciones: Se espera 20000 huellas correspondientes a los Clientes y 100 huellas correspondientes a los Usuarios Administrativos, entonces el número de relaciones Huella-Usuario es el mismo al número de huellas. Además se maneja un factor de relleno (Factor_Relleno) igual a 100.

  La tabla 2.21 indica las características de cada columna de la tabla HuellaUsuario_tbl; y a continuación en las tablas 2.22 y 2.23 se encuentran el resultado del espacio requerido para el nivel de hoja y para el índice respectivamente.

Columna	**Tipo de Dato**	**Tamaño (bytes)**	**Admite NULL**
IdHU	int	4	No
IdHuella	int	4	No
IdUA	varchar	13	Si
IdCli	varchar	19	Si

Tabla 2. 21: Tamaño de columnas en tabla HuellaUsuario_tbl

Num_Filas	20100
Num_Cols [columnas]	4
Tam_Fijo_Datos [bytes]	8
Num_Cols_Var [columnas]	2

Tam_Max_Var [bytes]	32
Null_Bit_Map [bytes]	3
Tam_Datos_Var [bytes]	38
Tam_Fila [bytes]	53
Filas_Pagina[Filas]	147
Filas_Libres_Reserva[Filas]	0
Num_Paginas [páginas]	137

Espacio_Hoja	***1122304***

Tabla 2. 22: Espacio requerido para tabla HuellaUsuario_tbl en nivel índice

Num_Cols_Clave [columnas]	1
Tam_Fijo_Datos _Clave[bytes]	4
Num_Cols_Var _Clave[columnas]	0
Tam_Max_Var_Clave [bytes]	0
Null_Bit_Map_Clave [bytes]	0
Tam_Datos_Var_Clave [bytes]	0
Tam_Fila_Indice [bytes]	11
Filas_Indice_Pagina[Filas]	622
Niveles_Indice [niveles]	2
Num_Paginas_Indice [Páginas]	33

Espacio_Indice	***270336***

Tabla 2. 23: Espacio requerido para índice en tabla HuellaUsuario_tbl

En base a estos datos el tamaño que podría alcanzar la tabla HuellaUsuario_tbl sería de:

$$Tam\_Tabla = Espacio\_Hoja + Espacio\_Indice = 1392640\ bytes$$

- *Tabla:* Log_tbl.

  Consideraciones: El tamaño de esta tabla está dado por el número de intentos de autenticación por parte de los clientes; suponiendo que en promedio cada cliente intenta acceder cuatro veces al mes a

los servicios del cajero automático, se tendría que cada mes el tamaño de la tabla se incrementaría en 80000 filas. En un año se tendría que la tabla alcanzaría un tamaño de 960000 filas.

La tabla 2.24 indica las características de cada columna de la tabla Log_tbl. Calculando el dimensionamiento del servidor de base de datos para cinco años y tomando en cuenta un factor de relleno de 100, se tiene en las tablas 2.25 y 2.26 el resultado del espacio requerido para el nivel de hoja y para el índice respectivamente.

Columna	Tipo de Dato	Tamaño (bytes)	Admite NULL
IdLog	Int	4	No
FecLog	datetime	8	No
IdCli	varchar	19	No
IdCaj	varchar	20	No
EstadoTran	varchar	10	No

Tabla 2. 24: Tamaño de columnas en tabla Log_tbl

Num_Filas [filas]	4800000
Num_Cols [columnas]	5
Tam_Fijo_Datos [bytes]	12
Num_Cols_Var [columnas]	3
Tam_Max_Var [bytes]	49
Null_Bit_Map [bytes]	3
Tam_Datos_Var [bytes]	57
Tam_Fila [bytes]	76
Filas_Pagina[Filas]	103
Filas_Libres_Reserva[Filas]	0
Num_Paginas [páginas]	46602

Espacio_Hoja	***381763584***

Tabla 2. 25: Espacio requerido para tabla Log_tbl en nivel índice

Num_Cols_Clave [columnas]	1
Tam_Fijo_Datos _Clave[bytes]	4
Num_Cols_Var _Clave[columnas]	0

Tam_Max_Var_Clave [bytes]	0
Null_Bit_Map_Clave [bytes]	0
Tam_Datos_Var_Clave [bytes]	0
Tam_Fila_Indice [bytes]	11
Filas_Indice_Pagina[Filas]	622
Niveles_Indice [niveles]	2
Num_Paginas_Indice [Páginas]	7730

Espacio_Indice	***63324160***

Tabla 2. 26: Espacio requerido para índice en tabla Log_tbl

A continuación se indica el tamaño estimado para la tabla Log_tbl:

$$Tam\_Tabl = Espacio\_Hoja + Espacio\_Indice = 445087744\ bytes$$

- *Tabla:* Rol_tbl.

  Consideraciones: Sólo se toman en cuenta tres roles que puede adoptar un Usuario Administrativo: Enrolar, Actualizar o Administrar.

  La tabla 2.27 indica las características de cada columna de la tabla Rol_tbl; y a continuación en las tablas 2.28 y 2.29 se encuentran el resultado del espacio requerido para el nivel de hoja y para el índice respectivamente.

Columna	**Tipo de Dato**	**Tamaño (bytes)**	**Admite NULL**
IdRol	int	4	No
NomRol	varchar	20	No
DesRol	varchar	150	Si

Tabla 2. 27: Tamaño de columnas en tabla Rol_tbl

Num_Filas [filas]	3
Num_Cols [columnas]	3

Tam_Fijo_Datos [bytes]	4
Num_Cols_Var [columnas]	2
Tam_Max_Var [bytes]	170
Null_Bit_Map [bytes]	3
Tam_Datos_Var [bytes]	176
Tam_Fila [bytes]	187
Filas_Pagina[Filas]	42
Filas_Libres_Reserva[Filas]	0
Num_Paginas [páginas]	1

Espacio_Hoja	***8192***

Tabla 2. 28: Espacio requerido para tabla Rol_tbl en nivel índice

Num_Cols_Clave [columnas]	1
Tam_Fijo_Datos _Clave[bytes]	4
Num_Cols_Var _Clave[columnas]	0
Tam_Max_Var_Clave [bytes]	0
Null_Bit_Map_Clave [bytes]	0
Tam_Datos_Var_Clave [bytes]	0
Tam_Fila_Indice [bytes]	11
Filas_Indice_Pagina[Filas]	622
Niveles_Indice [niveles]	0
Num_Paginas_Indice [Páginas]	1

Espacio_Indice	***8192***

Tabla 2. 29: Espacio requerido para índice en tabla Rol_tbl

El tamaño esperado para la tabla Rol_tbl es de:

$$Tam\_Tabla = Espacio\_Hoja + Espacio\_Indice = 16384\ bytes$$

- *Tabla:* RolUsuario_tbl.

Consideraciones: Se tendrá un máximo de 100 Usuarios Administrativos, suponiendo que a cada uno le toca un promedio de dos roles se tendrán 200 relaciones Rol-Usuario Administrativo.

La tabla 2.30 indica las características de cada columna de la tabla RolUsuario_tbl; y a continuación en las tablas 2.31 y 2.32 se encuentran el resultado del espacio requerido para el nivel de hoja y para el índice respectivamente.

En base a los datos de las tablas se tiene que la tabla RolUsuario_tbl puede alcanzar un tamaño de:

$$Tam\_Tabla = Espacio\_Hoja + Espacio\_Indice = 16384\ bytes$$

Columna	Tipo de Dato	Tamaño (bytes)	Admite NULL
IdRU	int	4	No
IdRol	int	4	No
IdUA	varchar	13	No

Tabla 2. 30: Tamaño de columnas en tabla RolUsuario_tbl

Num_Filas [filas]	200
Num_Cols [columnas]	3
Tam_Fijo_Datos [bytes]	8
Num_Cols_Var [columnas]	1
Tam_Max_Var [bytes]	13
Null_Bit_Map [bytes]	3
Tam_Datos_Var [bytes]	17
Tam_Fila [bytes]	32
Filas_Pagina[Filas]	238
Filas_Libres_Reserva[Filas]	0
Num_Paginas [páginas]	1

Espacio_Hoja	***8192***

Tabla 2. 31: Espacio requerido para tabla RolUsuario_tbl en nivel índice

Num_Cols_Clave [columnas]	1
Tam_Fijo_Datos _Clave[bytes]	4
Num_Cols_Var _Clave[columnas]	0
Tam_Max_Var_Clave [bytes]	0
Null_Bit_Map_Clave [bytes]	0
Tam_Datos_Var_Clave [bytes]	0
Tam_Fila_Indice [bytes]	11
Filas_Indice_Pagina[Filas]	622
Niveles_Indice [niveles]	1
Num_Paginas_Indice [Páginas]	1

Espacio_Indice	***8192***

Tabla 2. 32: Espacio requerido para índice en tabla RolUsuario_tbl

- *Tabla:* UsuarioAdministrativo_tbl.

  Consideraciones: Se espera tener un máximo de 100 Usuarios Administrativos.

  La tabla 2.33 indica las características de cada columna de la tabla UsuarioAdministrativo_tbl; y a continuación en las tablas 2.34 y 2.35 se encuentran el resultado del espacio requerido para el nivel de hoja y para el índice respectivamente.

Columna	Tipo de Dato	Tamaño (bytes)	Admite NULL
IdUA	varchar	13	No
NomUA	varchar	15	No
IdCoop	varchar	20	No
PassUA	varchar	15	No
EstadoUA	varchar	10	No

Tabla 2. 33: Tamaño de columnas en tabla UsuarioAdministrativo_tbl

Num_Filas	100

Num_Cols [columnas]	5
Tam_Fijo_Datos [bytes]	0
Num_Cols_Var [columnas]	5
Tam_Max_Var [bytes]	73
Null_Bit_Map [bytes]	3
Tam_Datos_Var [bytes]	85
Tam_Fila [bytes]	92
Filas_Pagina[Filas]	86
Filas_Libres_Reserva[Filas]	0
Num_Paginas [páginas]	2

Espacio_Hoja	***16384***

Tabla 2. 34: Espacio requerido para tabla UsuarioAdministrativo_tbl en nivel índice

Num_Cols_Clave [columnas]	1
Tam_Fijo_Datos _Clave[bytes]	0
Num_Cols_Var _Clave[columnas]	1
Tam_Max_Var_Clave [bytes]	13
Null_Bit_Map_Clave [bytes]	0
Tam_Datos_Var_Clave [bytes]	17
Tam_Fila_Indice [bytes]	24
Filas_Indice_Pagina[Filas]	311
Niveles_Indice [niveles]	1
Num_Paginas_Indice [Páginas]	1

Espacio_Indice	***8192***

Tabla 2. 35: Espacio requerido para índice en tabla UsuarioAdministrativo_tbl

Finalmente se tiene que el tamaño estimado para la tabla Usuario_Administrativo puede alcanzar un valor de:

$$Tam\_Tabla = Espacio\_Hoja + Espacio\_Indice = 24576\ bytes$$

- *Número de usuarios:* El número de usuarios que accederán a la base de datos está dado por el número de Host Administrativos más uno correspondiente al Servidor, los mismos que se conectarán a la base de datos mediante las respectivas aplicaciones (AATMDACTILAR y SATMDACTILAR) utilizando la autenticación de *SQL Server*.
- *Frecuencia y tamaño de las transacciones:* La frecuencia de las transacciones depende de dos factores:
    - El número de clientes que buscan autenticación para lo cual la aplicación en el cajero HATMDactilar se comunica con la aplicación del servidor SATMDactilar y éste a su vez cada que recibe un mensaje consulta a la base de datos la información del Cliente que busca ser verificado en base al identificador de la tarjeta. De esta manera considerando sólo este factor, el número de transacciones por minuto que atendería el servidor de base de datos sería igual al que atiende el servidor de aplicación.
    - El otro factor a considerar es el número de transacciones por minuto solicitadas por el Host Administrativo, las mismas que dependerán del número de Usuarios Administrativos en busca de autenticación o actualización y el número de clientes que requieren ser enrolados o actualizados.

2.5.2.3 Requerimientos de Hardware

- *Procesador:* Los requerimientos mínimos del procesador son los indicados anteriormente para *Microsoft SQL Server* 2005, es decir que sea del tipo Pentium III en adelante y que su frecuencia sea mayor a los 600 MHz, aunque se recomienda que sea mayor o igual a 1 GHz.
- *Disco Duro:* El tamaño de disco mínimo requerido está dado por el tamaño esperado que tendrá la base de datos en base al tamaño de cada una de las tablas, así se tiene que el tamaño estimado de la base de datos sería de:

Nombre Tabla	Tamaño Tabla (bytes)
Cajero_tbl	24576
Cliente_tbl	3497984
Cooperativa_tbl	16384
Huella_tbl	33202176
HuellaUsuario_tbl	1392640
Log_tbl	445087744
Rol_tbl	16384
RolUsuario_tbl	16384
UsuarioAdministrativo_tbl	24576
Tamaño Base de Datos (bytes) [26]	**483278848**
Tamaño Base de Datos (MB)	**461**

Tabla 2. 36: Espacio requerido para la Base de Datos

Así se tiene que el tamaño del disco duro debe ser mayor a 461 Megabytes.

- *Memoria RAM:* En base a las recomendaciones de *SQL Server* 2005 se tiene que la memoria RAM debe ser de 512 Megabytes en adelante, pero se considerará la recomendación que indica un tamaño de memoria RAM mayor o igual a 1 Gigabyte.
- *Adaptador de Red:* El acceso al servidor de base de datos se lo realiza a través de la red LAN; como se indicó anteriormente generalmente las tarjetas de red vienen incorporadas en la placa madre del ordenador. Por tanto esta interfaz deberá ser RJ-45 y debe soportar una velocidad de 100 Mbps.

2.5.2.4 Opciones de Configuración

Tomando en cuenta el tipo de trabajo que va a desempeñar el servidor, es suficiente considerar un solo equipo que cumpla con los requerimientos indicados en la sección 2.5.2.3.

[26] Se considera que un 1 KB es igual a 1024 bytes, y que 1 MB es igual a 1024 KB.

2.5.2.5 Servidores Recomendados

Tipo de Procesador	*Pentium IV* en adelante
Velocidad de Procesador	1 GHz en adelante
Memoria RAM	512 MB
Disco Duro	Arquitectura: DAS Protocolo: SCSI Tamaño: 10 GB en adelante
Sistema Operativo	*Windows XP Home Edition* *Windows XP Professional Edition* *Windows Server 2003 Web Edition*

Tabla 2. 37: Servidores Recomendados

CAPÍTULO 3

PRUEBAS DE CAMPO Y ANÁLISIS DE RESULTADOS

3.1 PRUEBAS DE CAMPO

En este capítulo se presentan las pruebas a través de las cuales se demostrará la fiabilidad de la aplicación de verificación de huellas dactilares como método válido a ser aplicado en la autenticación de usuarios de cajeros automáticos.

Se pueden realizar diferentes tipos de pruebas para revisar la confiabilidad y comportamiento del sistema en todos sus niveles o componentes. Por ejemplo se pueden realizar pruebas del algoritmo empleado, se pueden realizar pruebas para ver el nivel de desempeño y fiabilidad del SDK, se pueden hacer pruebas sobre el tiempo de respuesta de la aplicación, se pueden realizar pruebas de confiabilidad de aplicación y se pueden realizar pruebas para ver la aceptación de la aplicación por parte de los usuarios.

De las pruebas antes mencionadas, las pruebas del algoritmo así como las pruebas del SDK normalmente son realizadas por organismos internacionales a través de una competencia mundial en la cual se llevan pruebas con diferentes lectores e imágenes. Esta competencia se conoce como *FVC* por sus siglas en inglés (*Fingerprint Verification Competition*) y se realiza cada dos años. Esta competencia es la más reconocida a nivel mundial y es organizada por varias universidades y organizaciones entre las que destacan la universidad de Bologna de Italia, la Universidad del Estado de San José de Estados Unidos, la Universidad Estatal de Michigan y la Universidad Autónoma de Madrid. En este concurso se prueba los algoritmos implementados en los SDKs mediante pruebas de tiempos de respuesta hacia la base de datos, tiempos de respuesta promedios frente a una solicitud, y respuesta frente a solicitudes, validando si la huella es la correcta o no, así como si se dio un falso positivo o un falso negativo. El SDK de Griaule Biometrics, que es el que se ha utilizado en el presente proyecto de titulación, es el ganador del concurso FVC del año 2006 lo que garantiza la fiabilidad de este SDK.

Las pruebas de confiabilidad de la aplicación así como las pruebas del tiempo de respuesta de la aplicación frente a una solicitud se las ha realizado a nivel de laboratorio bajo dos escenarios; en el primer escenario se probó la aplicación desarrollada en un computador en donde se emuló el cajero automático mientras que en el segundo escenario se probó la aplicación ya instalada en un cajero automático. Los resultados de estas pruebas se detallan en las secciones 3.1.1 y 3.1.2.

Bajo los dos escenarios, se realizaron 4 tipos de pruebas con las cuales se probó en general la aplicación desarrollada, estas pruebas son:

- *Tiempo de Respuesta (Prueba 1):* Para la realización de esta prueba se midieron los tiempos desde que la aplicación cliente captura la huella dactilar, procesa la huella generando la plantilla a ser verificada, se envía a través de la red hasta el servidor, el servidor procesa el mensaje y consulta en la base de datos la plantilla de referencia, se hace la verificación de las huellas dactilares entre la plantilla de referencia y la plantilla a ser verificada, se envía la respuesta a la aplicación cliente quien muestra los resultados. Para la medición de los tiempos se utilizó archivos de logs en los cuales se escribía la fecha y hora en la que se generaba un evento tanto en el servidor como en el cliente; luego se procedió a comparar los logs para obtener los tiempos de respuesta en una transacción. Los tiempos que se van a analizar en esta prueba para los dos escenarios se encuentran en la tabla 3.1 y todos los tiempos son medidos en milisegundos.

Procesar Mensaje RX (ms)	**Verificar Huella (ms)**	**Procesar Mensaje TX (ms)**	**Total (ms)**

Tabla 3. 1: Tiempos a ser medidos en pruebas

- *Tiempo de Respuesta con Carga del Servidor (Prueba 2):* Para esta prueba se utilizó el software *Load Runner* de HP en su versión 9.5. Este software permite visualizar el desempeño de una aplicación simulando tener varios clientes conectados simultáneamente y poder analizar así los posibles “cuellos de botella” en los servidores, en la red o en la aplicación. *Load Runner* consta de tres herramientas; la primera es el generador de usuarios virtuales que graba

el comportamiento típico de los procesos de una aplicación o cliente dentro de un *script* que al "correr" simula una situación real del comportamiento del usuario, éste se conoce como un "usuario virtual". En la figura 3.1 se observa la interfaz de usuario que permite crear un usuario virtual; para esta prueba se necesita simular peticiones de autenticación por parte del Host ATMDactilar, por ende el usuario virtual graba en un *script* todas las acciones necesarias para simular este comportamiento.

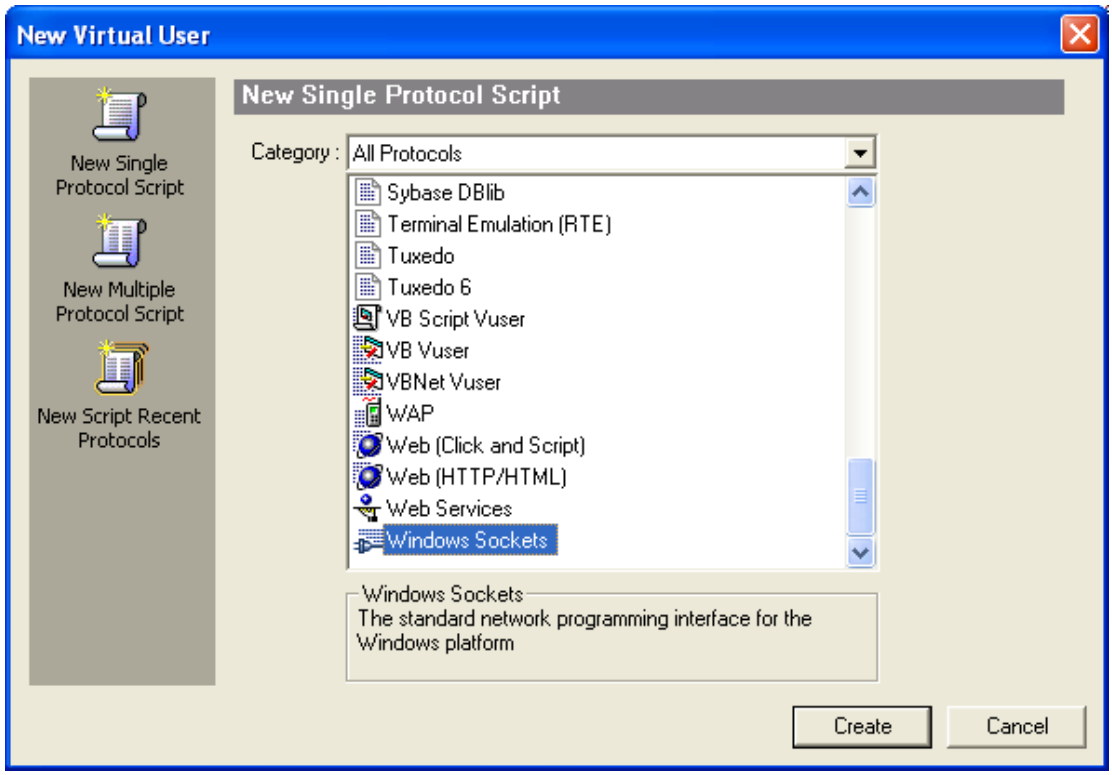


Figura 3. 1: Interfaz *Virtual User Generator Load Runner*, creación de nuevo usuario virtual

La segunda herramienta permite generar escenarios de varios clientes virtuales simultáneos y configurar los parámetros de desempeño que se desean monitorear; esta herramienta es el *Load Runner Controller*, que por limitaciones de la versión trial solo permite correr en un escenario 25 clientes simultáneos. En la figura 3.2 se puede observar la interfaz que permite la configuración del escenario (clientes simultáneos, secuenciales o una combinación de ambos escenarios).

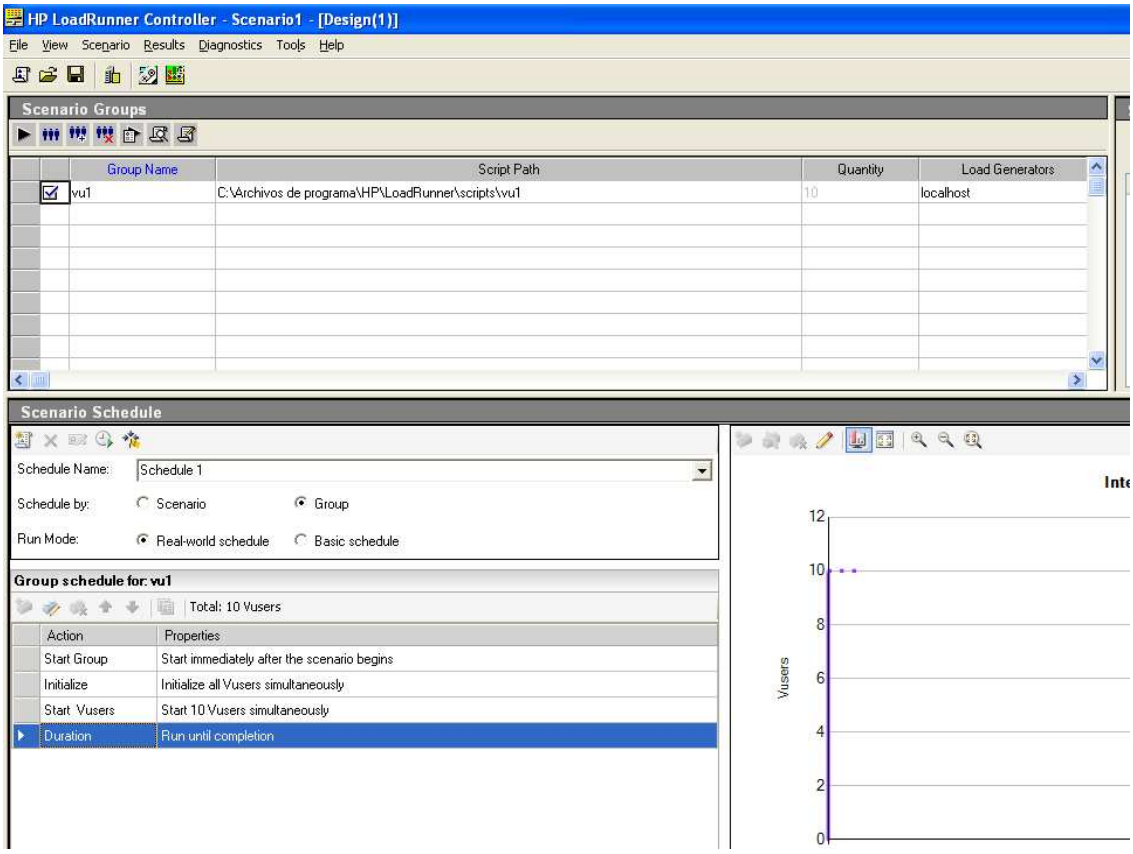


Figura 3. 2: Interfaz *Load Runner Controller*, creación de nuevos escenarios

Las características que se pueden tener en un escenario con esta versión son las presentadas en la tabla 3.2.

Característica	Valor
Número de Usuarios Virtuales Simultáneos	10
Número de peticiones secuenciales por Usuario Virtual	6
Duración del escenario	Hasta que todos las peticiones sean respondidas

Tabla 3. 2: Características Escenario de Prueba

Finalmente, la tercera herramienta de *Load Runner* es la de *Analysis Load Runner* que permite justamente procesar y analizar los resultados.

- *Confiabilidad de la Aplicación (Prueba 3):* Mediante esta prueba se intenta evaluar la capacidad de la aplicación para reconocer las huellas y dar como válida una huella cuando realmente lo es. Esta prueba se evaluó en dos fases;

en la primera de ellas se validó la capacidad de la aplicación para determinar los falsos negativos esto se lo hizo probando la aplicación sólo con huellas válidas; mientras que en la segunda parte se validó la capacidad de la aplicación determinando los falsos positivos mediante pruebas en las que se colocó huellas que no eran las correctas y determinando si la aplicación las verificaba como válidas.

- *Nivel de Aceptación del Usuario (Prueba 4):* Con esta prueba se determinó la aceptación del usuario frente al sistema desarrollado; el parámetro de medición fue tomado en base a las opiniones de cada persona que usó el sistema y su grado de confortabilidad frente al sistema en general. Vale la pena destacar que a través de esta prueba se hizo una evaluación subjetiva, ya que el grado de confortabilidad de un usuario depende del grado de exigencia del usuario y su grado de disponibilidad de uso de la aplicación. Esta prueba servirá para determinar si el usuario prefiere usar un cajero automático que base su autenticación en huellas dactilares o en el uso del PIN. La realización de la prueba se la hizo bajo la consulta de tres preguntas después de usar la aplicación, pero sólo se la realizó para el escenario 2.

3.1.1 COMPUTADORA EMULANDO CAJERO

En este escenario se probó la aplicación en un computador que mantenía conexión al servidor mediante una LAN. El hardware de este computador tiene características similares a la de los componentes que se tienen disponibles en un cajero automático para acercarse lo más posible a la realidad del cajero.

Bajo este escenario se realizaron 6000 pruebas en el transcurso de 4 meses en los que se probó reiterativamente la aplicación y sus resultados son los siguientes:

- *Prueba 1:* El tiempo promedio total de respuesta que se tuvo en las pruebas fue de 14.95 milisegundos con un máximo de 32 milisegundos y un mínimo de 0 milisegundos. En la tabla 3.3 se presentan estos resultados y sus gráficas estadísticas se indican en la figura 3.3.
- *Prueba 2:* El tiempo promedio total de respuesta que se tuvo en las pruebas fue de 18.65 milisegundos con un máximo de 57 milisegundos y

un mínimo de 0 milisegundos. En la tabla 3.4 se presentan estos resultados y sus gráficas estadísticas se indican en la figura 3.4.

- *Prueba 3*: En la tabla 3.5 se tienen los resultados tanto para la primera como para la segunda fase. Sus estadísticas se encuentran en la figura 3.5. Lo destacable de la prueba es que no se tuvo ningún falso positivo.

Máximo tiempo en Procesar Mensaje RX (ms)	16
Mínimo tiempo en Procesar Mensaje RX (ms)	0
Promedio[27] en Procesar Mensaje RX (ms)	8.65
Mediana[28] en Procesar Mensaje RX (ms)	15

Máximo tiempo en Verificar Huella (ms)	16
Mínimo tiempo en Verificar Huella (ms)	0
Promedio en Verificar Huella (ms)	5.9
Mediana en Verificar Huella (ms)	0

Máximo tiempo en Procesar Mensaje TX (ms)	16
Mínimo tiempo en Procesar Mensaje TX (ms)	0
Promedio en Procesar Mensaje TX (ms)	0.4
Mediana en Procesar Mensaje TX (ms)	0

Máximo tiempo Total (ms)	32
Mínimo tiempo Total (ms)	0
Promedio tiempo Total (ms)	14.95
Mediana tiempo Total (ms)	16

Tabla 3. 3: Resultados de la Prueba 1- Computadora Emulando Cajero

Máximo tiempo en Procesar Mensaje RX (ms)	32
Mínimo tiempo en Procesar Mensaje RX (ms)	0
Promedio en Procesar Mensaje RX (ms)	10.2
Mediana en Procesar Mensaje RX (ms)	15

Máximo tiempo en Verificar Huella (ms)	16
Mínimo tiempo en Verificar Huella (ms)	0
Promedio en Verificar Huella (ms)	5.6
Mediana en Verificar Huella (ms)	2

[27] Promedio: Devuelve el promedio o la media aritmética.
[28] Mediana: Representa el número central de un conjunto de números.

Máximo tiempo en Procesar Mensaje TX (ms)	32
Mínimo tiempo en Procesar Mensaje TX (ms)	0
Promedio en Procesar Mensaje TX (ms)	2.85
Mediana en Procesar Mensaje TX (ms)	0

Máximo tiempo Total (ms)	57
Mínimo tiempo Total (ms)	0
Promedio tiempo Total (ms)	18.65
Mediana tiempo Total (ms)	16.5

Tabla 3. 4: Resultados de la Prueba 2 – Computadora Emulando Cajero

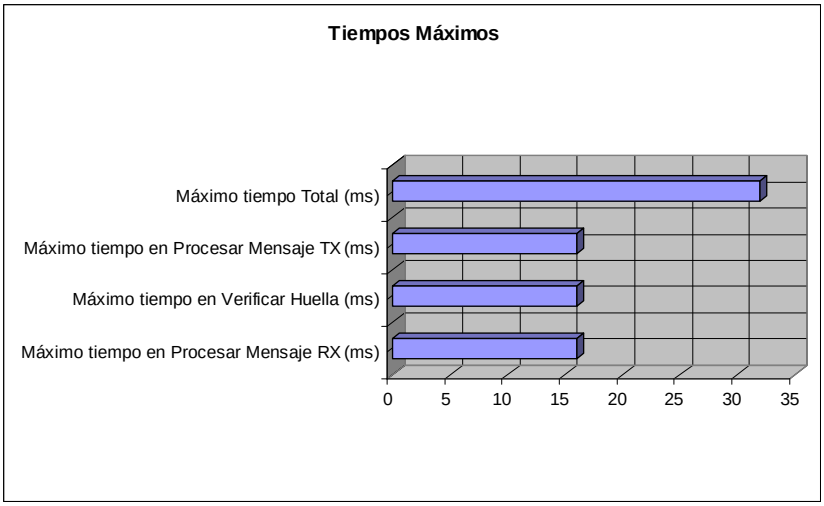
Fase 1

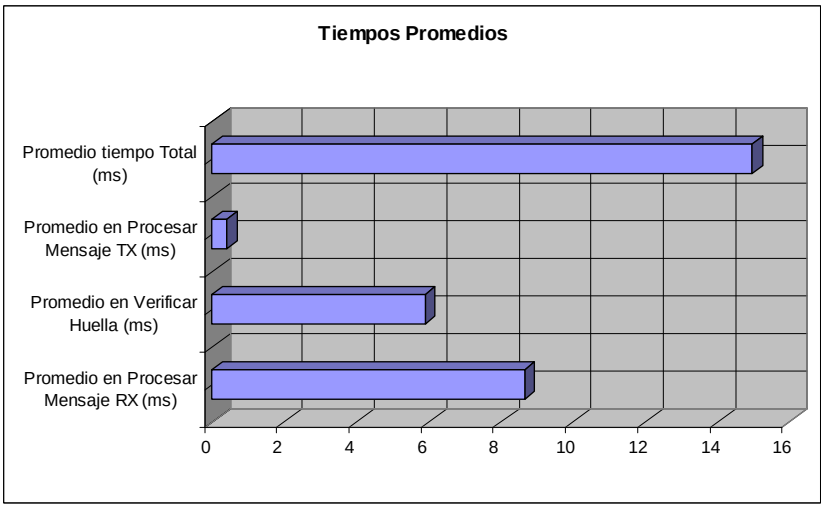
Número de Intentos	3000
Huellas Autenticadas	2892
Huellas No Autenticadas	108
Porcentaje de Huellas Autenticadas	96.40%
Porcentaje de Huellas No Autenticadas	3.60%

Fase 2

Número de Intentos	3000
Huellas Autenticadas	0
Huellas No Autenticadas	3000
Porcentaje de Huellas Autenticadas	0.00%
Porcentaje de Huellas No Autenticadas	100.00%

Tabla 3. 5: Resultados de la Prueba 3 – Computadora Emulando Cajero





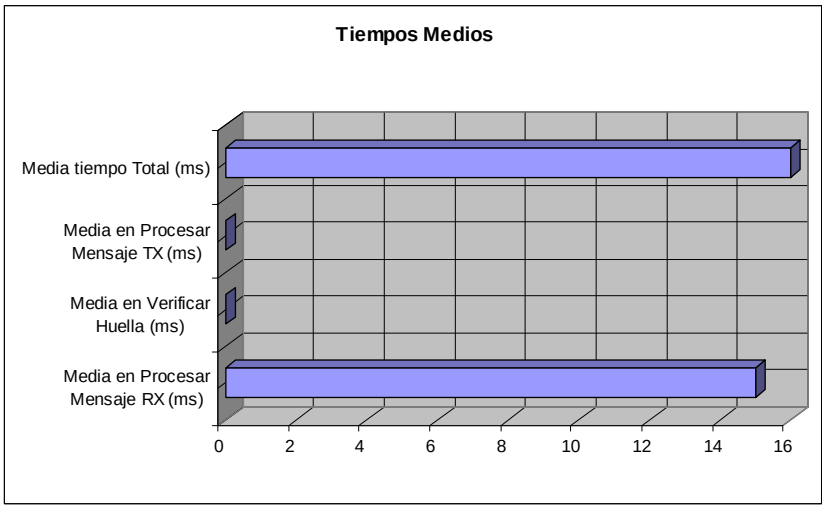
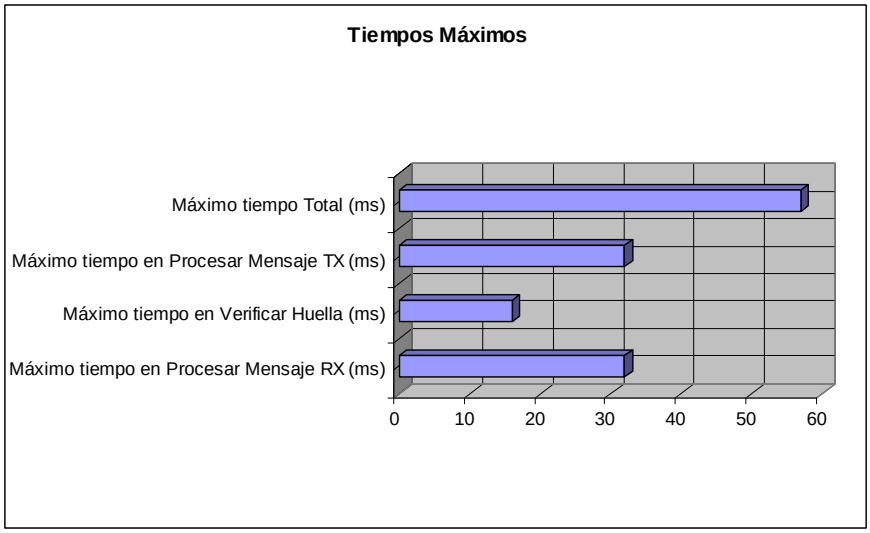
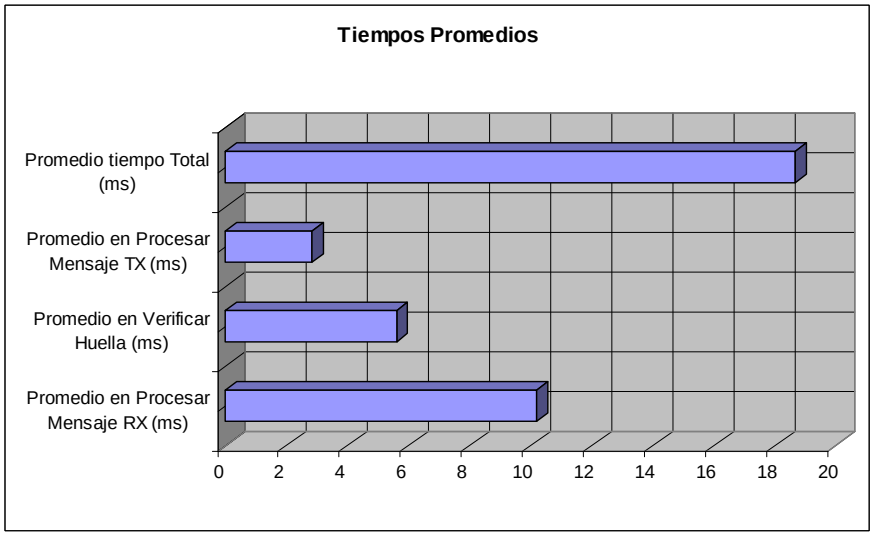


Figura 3. 3: Resultados de la Prueba 1- Computadora Emulando Cajero





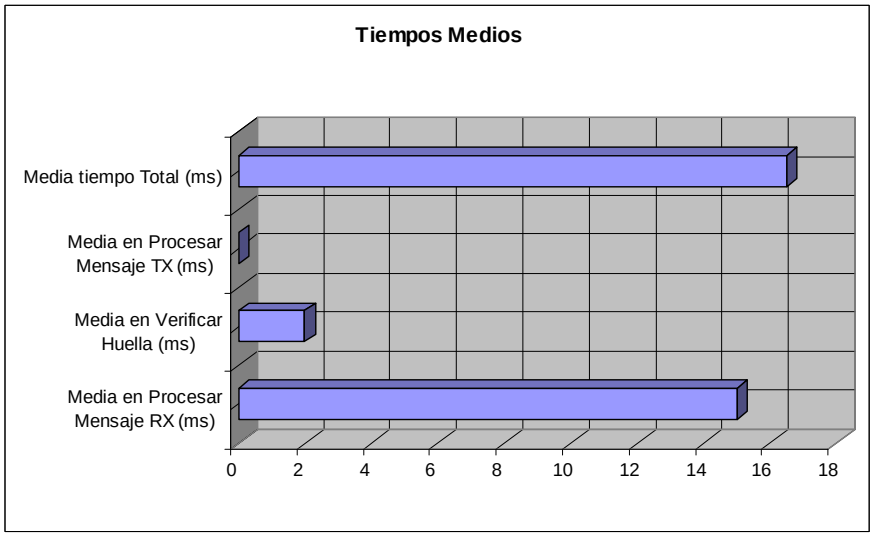


Figura 3. 4: Resultados de la Prueba 2 – Computadora Emulando Cajero

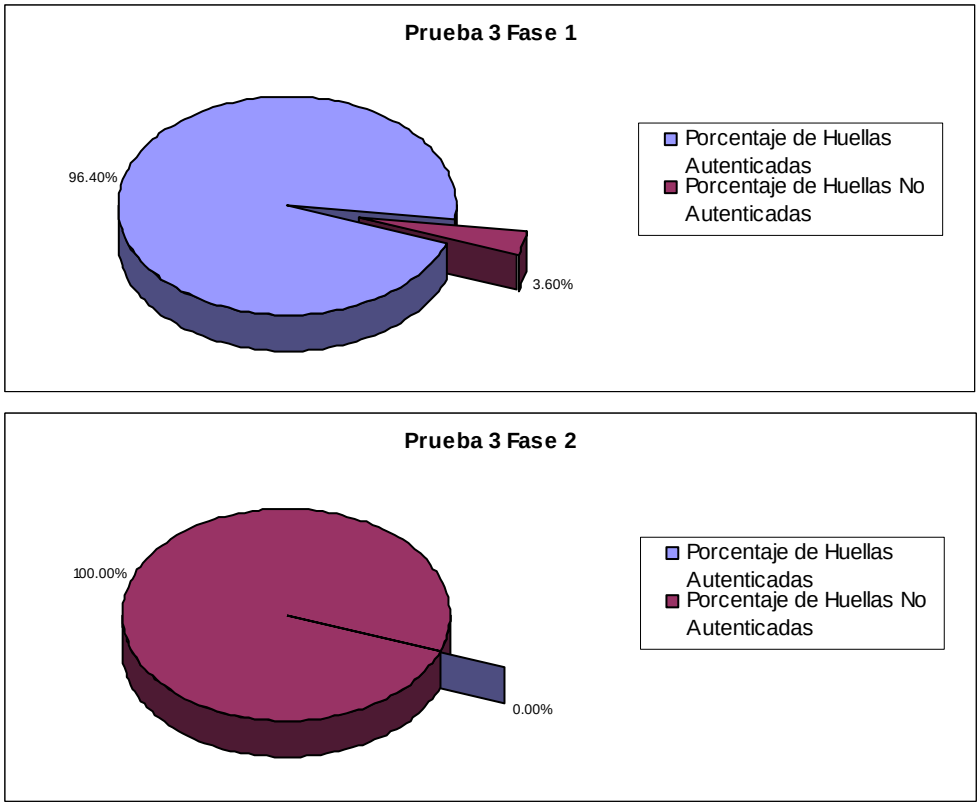


Figura 3. 5: Resultados de la Prueba 3 – Computadora Emulando Cajero

3.1.2 CAJERO AUTOMÁTICO

En este escenario se probó la aplicación en un cajero automático de pruebas, marca NCR que mantenía una conexión al servidor mediante una LAN. Este cajero automático tiene todos los dispositivos en el cajero y el dinero dispensado era dinero de prueba así como las opciones del cajero configuradas sólo permiten el retiro del bono.

Bajo este escenario se realizaron 10000 ensayos para las pruebas 1, 2 y 3 en el transcurso de 2 meses; mientras que para la prueba 4 se realizaron 200 pruebas gracias a la colaboración de voluntarios. Los resultados de estas pruebas son:

- *Prueba 1:* El tiempo promedio total de respuesta que se tuvo en las pruebas fue de 10.38 milisegundos con un máximo de 35 milisegundos y un mínimo de 0 milisegundos. En la tabla 3.6 se presentan estos resultados y sus gráficas estadísticas se indican en la figura 3.6.

- *Prueba 2:* El tiempo promedio total de respuesta que se tuvo en las pruebas fue de 14.1 milisegundos con un máximo de 40 milisegundos y un mínimo de 0 milisegundos. En la tabla 3.7 se presentan estos resultados y sus gráficas estadísticas se indican en la figura 3.7.

Máximo tiempo en Procesar Mensaje RX (ms)	17
Mínimo tiempo en Procesar Mensaje RX (ms)	0
Promedio en Procesar Mensaje RX (ms)	7.275
Media en Procesar Mensaje RX (ms)	5.5

Máximo tiempo en Verificar Huella (ms)	16
Mínimo tiempo en Verificar Huella (ms)	0
Promedio en Verificar Huella (ms)	2.45
Media en Verificar Huella (ms)	0

Máximo tiempo en Procesar Mensaje TX (ms)	14
Mínimo tiempo en Procesar Mensaje TX (ms)	0
Promedio en Procesar Mensaje TX (ms)	0.65
Media en Procesar Mensaje TX (ms)	0

Máximo tiempo Total (ms)	35
Mínimo tiempo Total (ms)	0
Promedio tiempo Total (ms)	10.375
Media tiempo Total (ms)	12.5

Tabla 3. 6: Resultados de la Prueba 1 – Cajero Automático

Máximo tiempo en Procesar Mensaje RX (ms)	16
Mínimo tiempo en Procesar Mensaje RX (ms)	0
Promedio en Procesar Mensaje RX (ms)	2.625
Media en Procesar Mensaje RX (ms)	0

Máximo tiempo en Verificar Huella (ms)	14
Mínimo tiempo en Verificar Huella (ms)	0
Promedio en Verificar Huella (ms)	1.25
Media en Verificar Huella (ms)	0

Máximo tiempo en Procesar Mensaje TX (ms)	28
Mínimo tiempo en Procesar Mensaje TX (ms)	0
Promedio en Procesar Mensaje TX (ms)	10.225
Media en Procesar Mensaje TX (ms)	12

Máximo tiempo Total (ms)	40
Mínimo tiempo Total (ms)	0
Promedio tiempo Total (ms)	14.1
Media tiempo Total (ms)	14

Tabla 3. 7: Resultados de la Prueba 2 – Cajero Automático

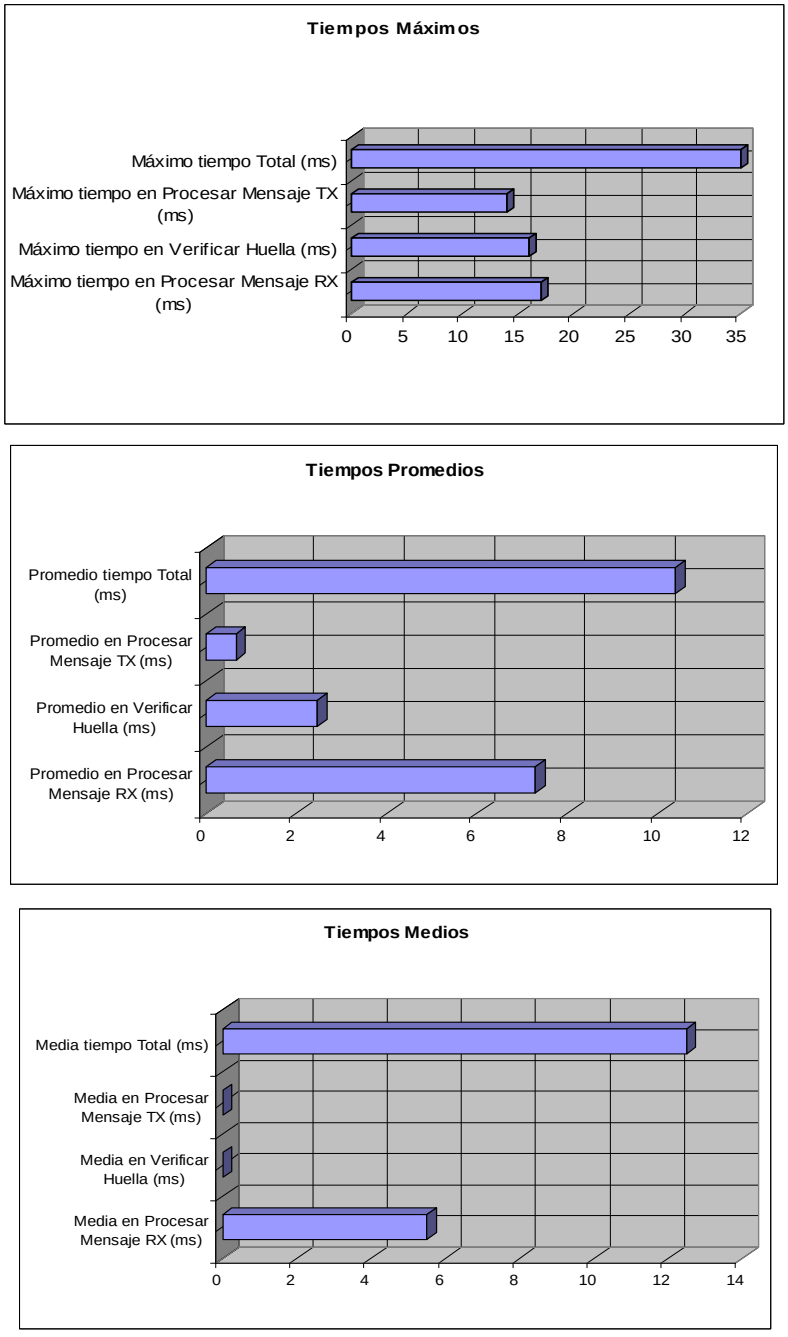


Figura 3. 6: Resultados de la Prueba 1 – Cajero Automático

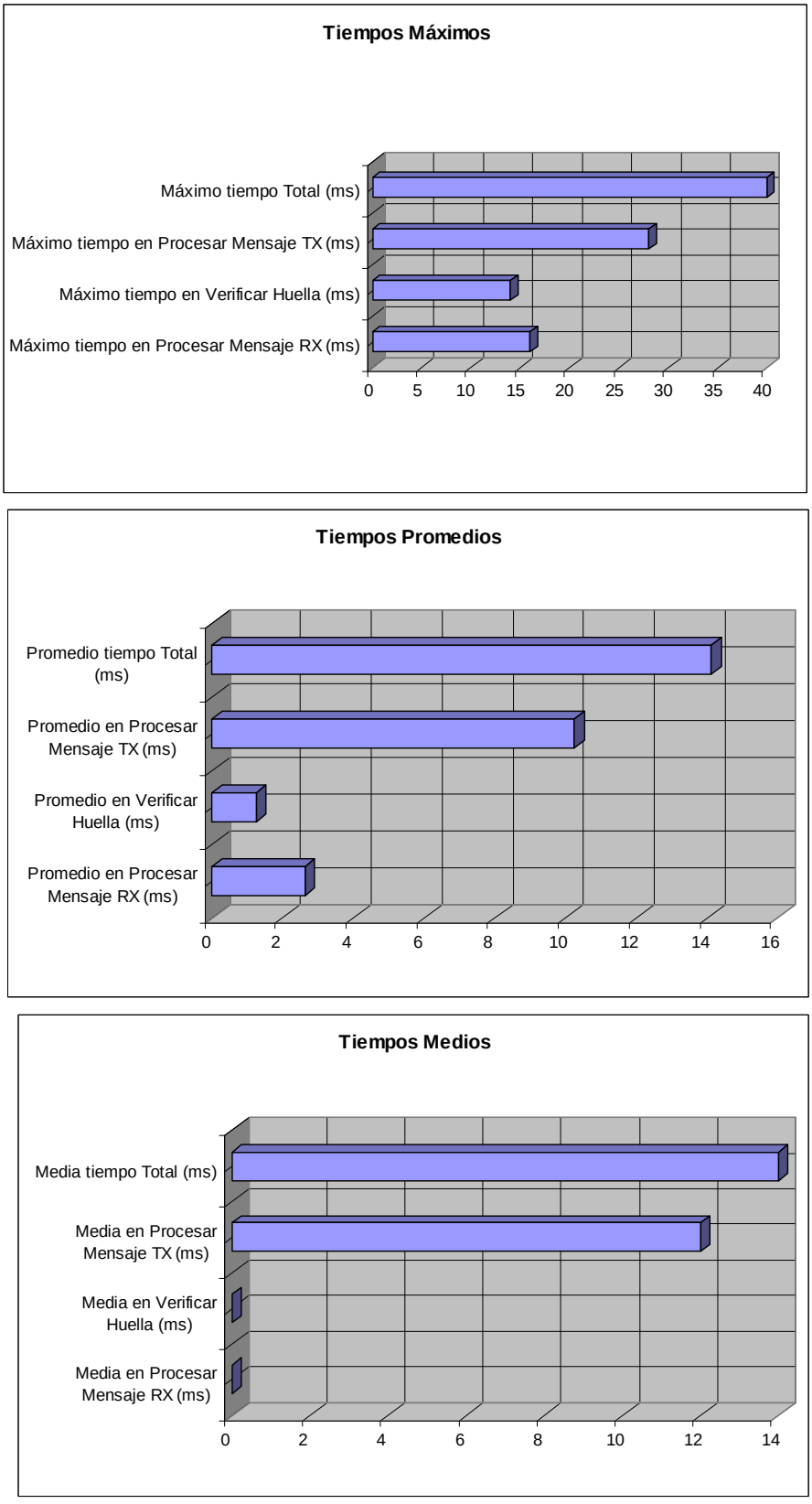


Figura 3. 7: Resultados de la Prueba 2 – Cajero Automático

En la figura 3.8 se muestran los resultados obtenidos en la prueba de carga generada desde un cajero automático mediante el *Load Runner* y en la tabla 3.8 se ven los resultados de dicha prueba.

Variables	Prueba 1
Tiempo de Respuesta del Escenario [s]	2
Tiempo de Respuesta Promedio a cada petición [ms]	14.1
Tiempo de Respuesta Promedio para las 25 peticiones [ms]	0.4036

Tabla 3. 8: Resultados Pruebas Escenario

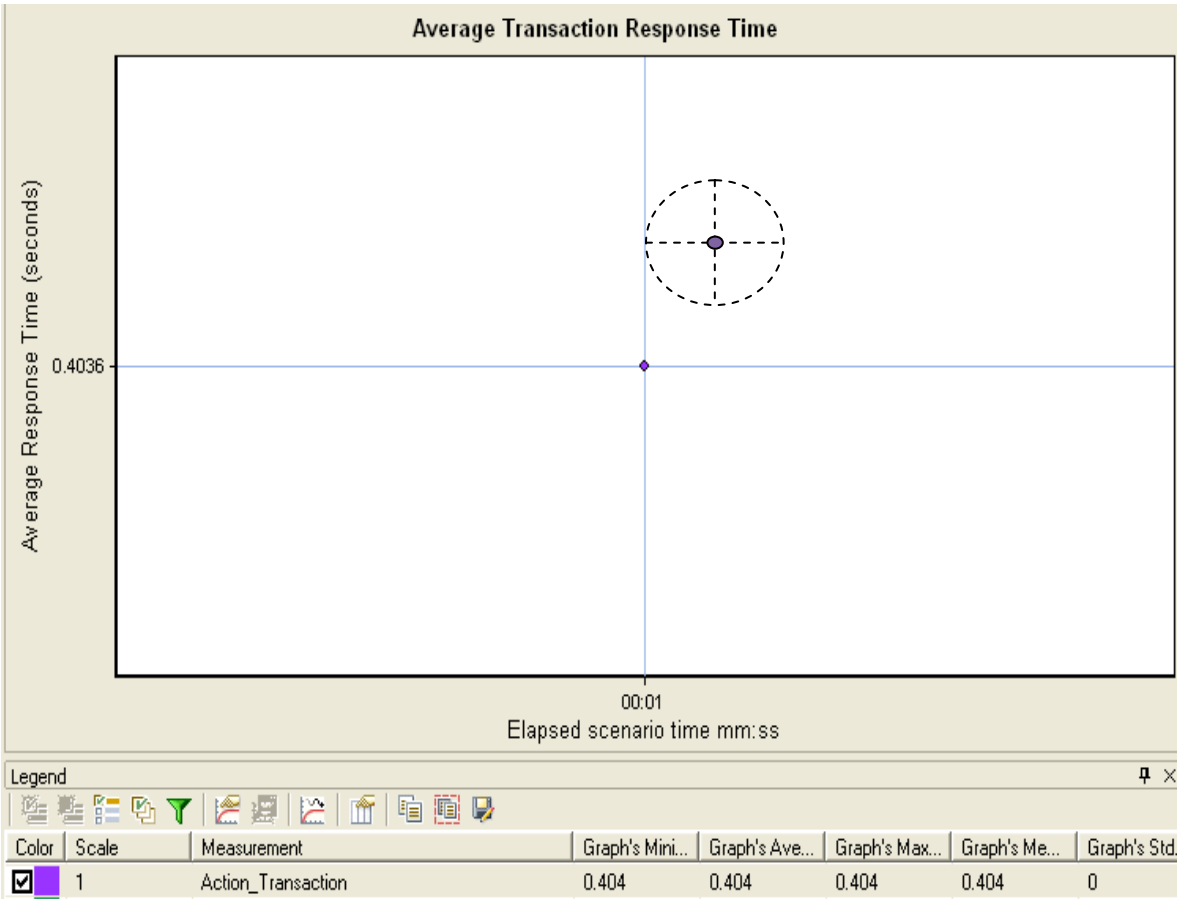


Figura 3. 8: Tiempo de Respuesta promedio del Escenario

En la figura 3.8 se puede observar que 1 segundo después de ser enviadas al servidor todas las peticiones de autenticación, éstas son procesadas en ese instante en un tiempo promedio de 0.4036 segundos. Si se divide este tiempo para las 25 transacciones se tiene que el promedio de cada una de las transacciones es de 14.1 milisegundos, con un máximo de 40 milisegundos y un mínimo de 0 milisegundos.

- *Prueba 3*: En la tabla 3.9 se tienen los resultados tanto para la primera como para la segunda fase. Sus estadísticas se encuentran en la figura 3.9. Lo destacable de la prueba es que no se tuvo ningún falso positivo.
- *Prueba 4*: Las respuestas para esta prueba se presentan en la tabla 3.10 y los gráficos estadísticos se presentan en la figura 3.10.

Fase 1

Número de Intentos	5000
Huellas Autenticadas	4885
Huellas No Autenticadas	115
Porcentaje de Huellas Autenticadas	97.70%
Porcentaje de Huellas No Autenticadas	2.30%

Fase 2

Número de Intentos	3000
Huellas Autenticadas	0
Huellas No Autenticadas	3000
Porcentaje de Huellas Autenticadas	0.00%
Porcentaje de Huellas No Autenticadas	100.00%

Tabla 3. 9: Resultados de la Prueba 3 – Cajero Automático

Pregunta 1: ¿El sistema fue intuitivo en su uso?

Total	200
SI	174
NO	26
SI %	87.00%
NO %	13.00%

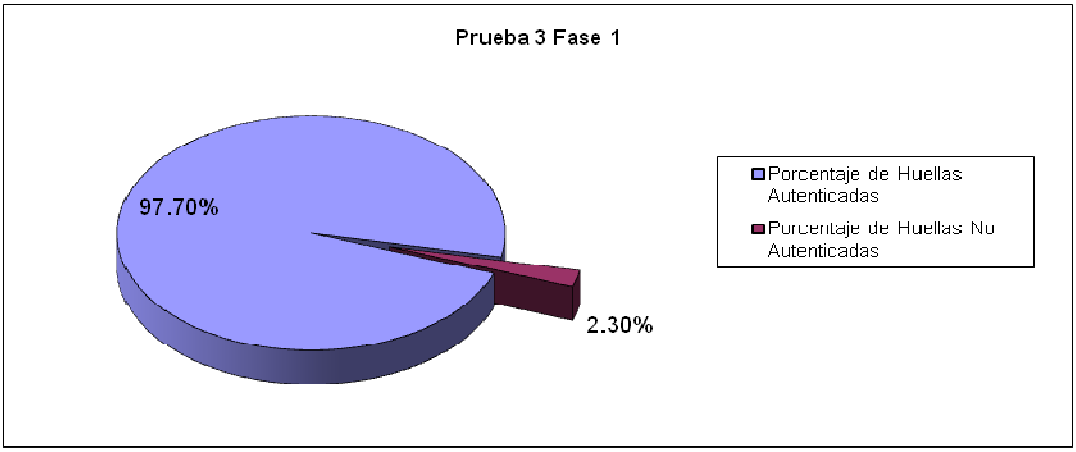
Pregunta 2: ¿Le parece un sistema más seguro que el uso de claves?

Total	200
SI	193
NO	7
SI %	96.50%
NO %	3.50%

Pregunta 3: ¿Estaría dispuesto a usar un cajero automático con huellas dactilares?

Total	200
SI	185
NO	15
SI %	92.50%
NO %	7.50%

Tabla 3. 10: Resultados de la Prueba 4 – Cajero Automático



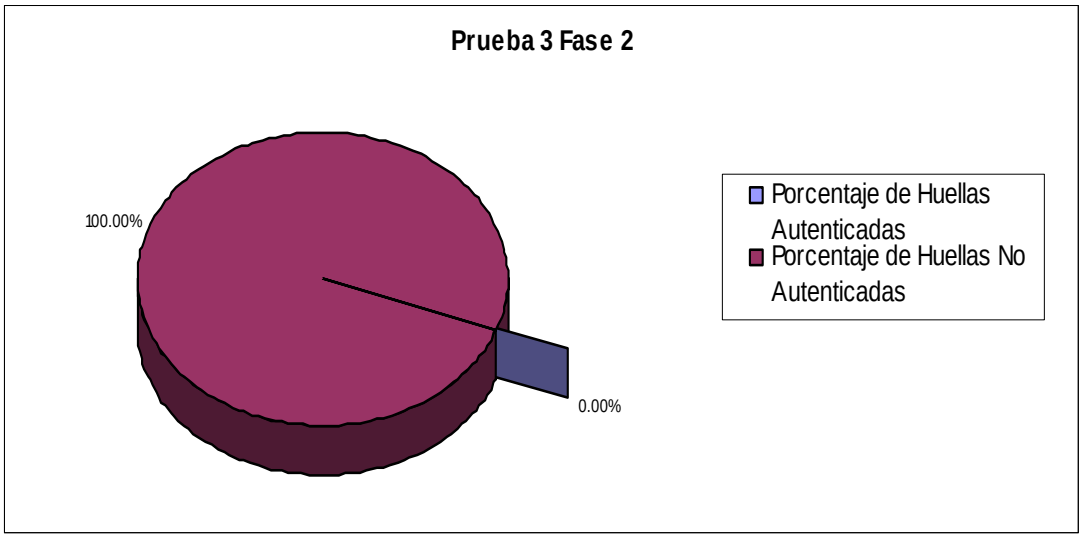
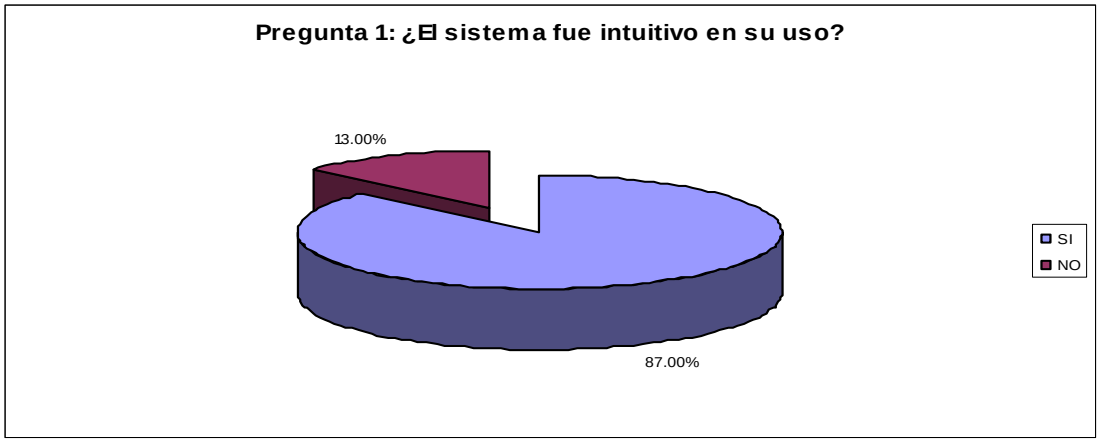
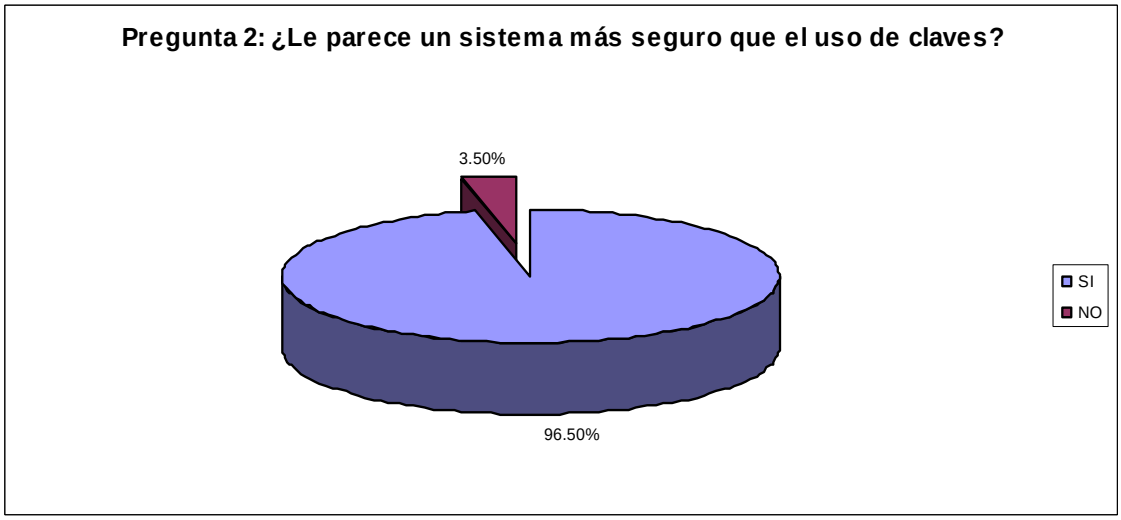


Figura 3. 9: Resultados de la Prueba 3 – Cajero Automático





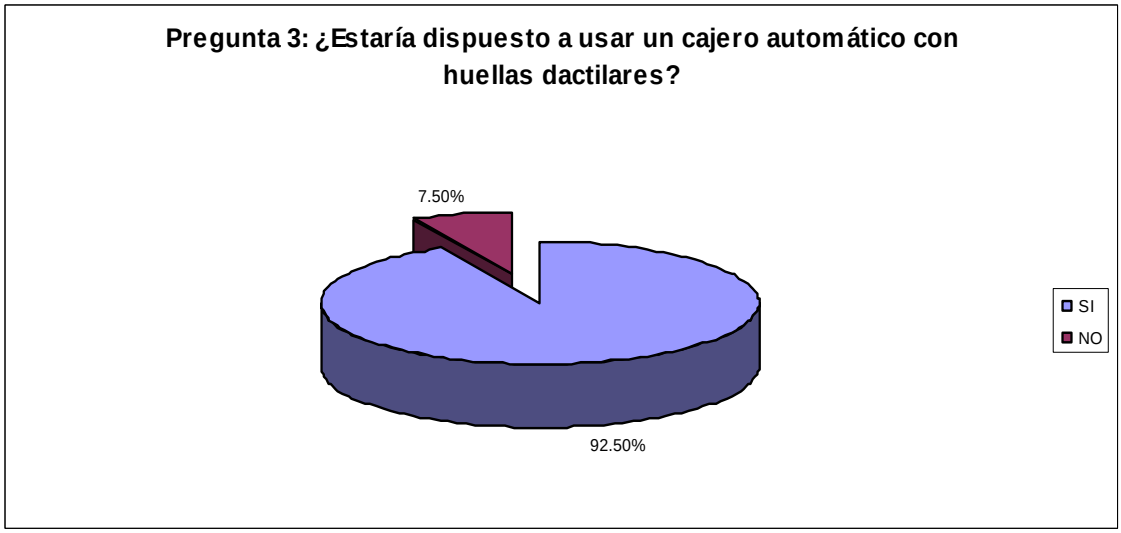


Figura 3. 10: Resultados de la Prueba 4 – Cajero Automático

3.1.3 PRUEBAS DE SEGURIDAD

Además de las pruebas indicadas anteriormente, se realizaron pruebas para comprobar el grado de seguridad de la aplicación determinando a través de un *sniffer* si la trama que viaja del cajero al servidor realmente tenía encriptada la huella dactilar; además se revisó si la trama que viaja con la respuesta del servidor es susceptible de algún ataque. El *sniffer* que se empleó para esta prueba es el *Wireshark* y los resultados se indican en figura 3.11.

En la figura 3.12 se indica el arribo del mensaje desde el cliente al servidor visto desde la aplicación SATMDactilar. La atención de este cliente es la que hace referencia a la figura 3.11.

De la trama que revela el *sniffer* en la figura 3.11 se puede apreciar que la huella se encuentra completamente encriptada; esto se evidencia ya que si la huella estuviera en texto plano en la parte derecha de la captura se vería el texto plano con información de la trama, es decir con el número de identificación y la huella. La figura 3.13 muestra la trama capturada.

Del mismo modo la trama de respuesta del servidor viaja encriptada hacia el host ya que se tiene la información del proceso de autenticación del cliente. Esta trama se la puede evidenciar en la figura 3.14.

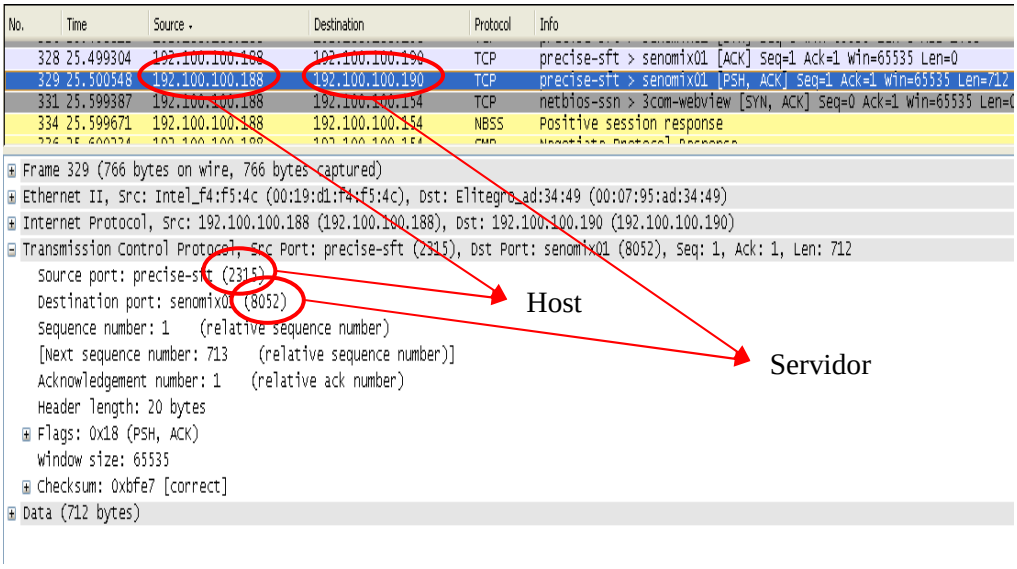


Figura 3. 11: Trama del host hacia el servidor capturada por Wireshark

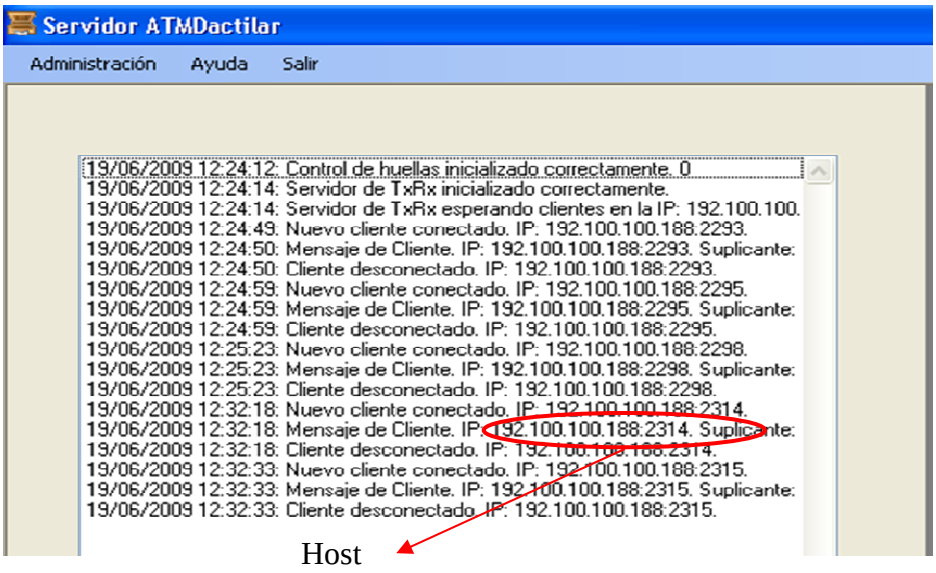


Figura 3. 12: Atención de un cliente por parte del servidor

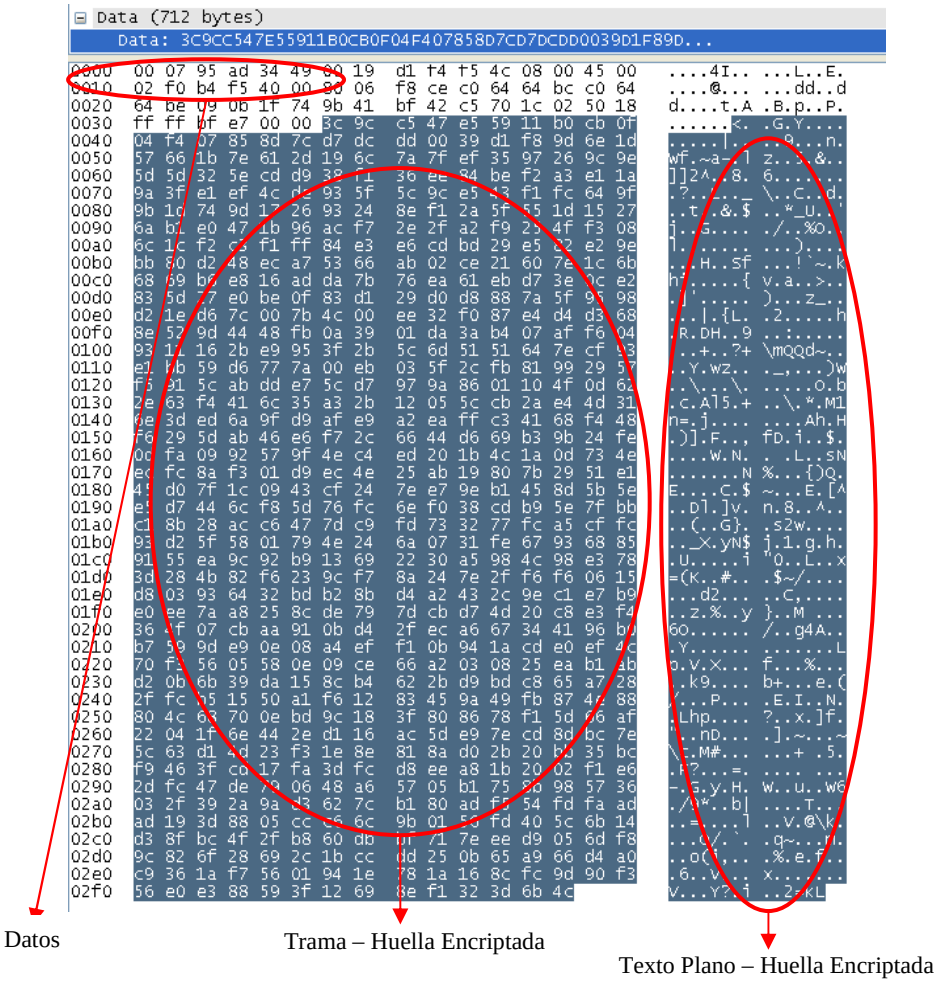


Figura 3. 13: Huella dactilar encriptada

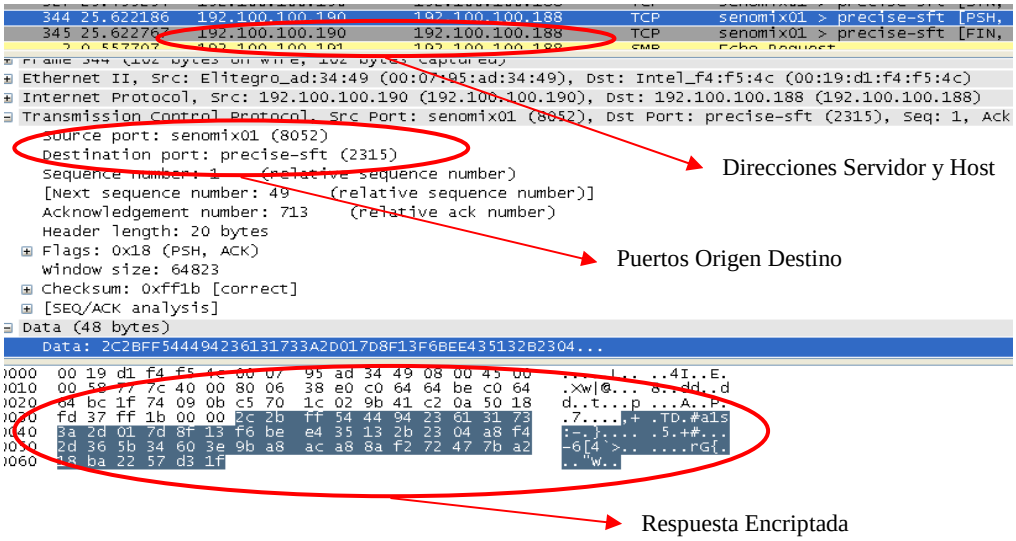


Figura 3. 14: Trama de Respuesta del Servidor hacia el Host

3.2 ANÁLISIS DE RESULTADOS

3.2.1 PROMEDIO DE TIEMPO PARA IDENTIFICACIÓN DE UN USUARIO

De los análisis realizados se puede apreciar que la diferencia de tiempo entre el uso del cajero automático y un computador es de 3.7 ms en condiciones normales y cuando se realizan múltiples conexiones hacia el servidor la diferencia de tiempo entre tener una sola transacción y tener varias es de 0.15 ms. Por ser valores de tiempo muy pequeños las diferencias son imperceptibles para el usuario y bastante difíciles de medir.

Ya en producción el sistema, se evidenciará que los tiempos de procesamiento de la huella así como la validación de la misma con el servidor son despreciables, frente al tiempo total de una transacción que puede alcanzar los 2 minutos. Del mismo modo ya en ambiente de producción los tiempos de comunicación entre el host y el servidor variarán dependiendo su interconexión; lo que incrementará el tiempo total de la autenticación. Por ejemplo si un cajero se conecta al servidor mediante un enlace satelital, el tiempo que dure la autenticación será diferente al tiempo que dure la autenticación entre un cajero que esté conectado al mismo servidor en un ámbito LAN a través de una red 100BaseT.

3.2.2 NIVEL DE ACEPTACIÓN POR PARTE DEL USUARIO

De los usuarios entrevistados, se puede apreciar que las respuestas demuestran que la aplicación tiene una gran aceptación y se evidencia también que su uso es bastante intuitivo para los nuevos usuarios. De lo conversado con las personas que usaron la aplicación, reconocen que la misma es más segura que el uso de una clave (96.5 %). Pero manifiestan su miedo frente al hecho que les puedan cortar el dedo o la mano para poder retirar dinero, por eso es que el 3.5 % de las personas prefieren usar un cajero con clave (el 3.5% es la diferencia entre las personas que lo consideran más seguro al cajero con huellas dactilares pero que no estarían dispuestos a usarlo).

3.2.3 NIVEL DE PRECISIÓN DEL SISTEMA

El 2.3% de los intentos de autenticación de una huella válida en el sistema falló, este valor representa los falsos negativos que presenta el sistema en un escenario en el cual ya se tiene instalado el cajero y conectado al servidor. Este valor aunque puede causar una molestia al cliente es comparable al escenario en donde una persona equivoca su clave. Este valor no depende de la aplicación sino de condiciones externas como intentar autenticarse con una huella mojada o en un lector sucio. En la práctica, el porcentaje de errores cometidos por el sistema es menor al porcentaje de transacciones rechazadas en los cajeros automáticos por error en el ingreso de la clave.

Lo más importante de esta prueba es el resultado de la segunda fase, en donde el 100% de las huellas que no debían ser autenticadas fueron rechazadas por el sistema con lo que se comprobó que la aplicación es robusta frente a los falsos positivos dando la garantía de ser un sistema confiable frente a ataques de violación de identidad.

CAPÍTULO 4

CONCLUSIONES Y RECOMENDACIONES

4.1 CONCLUSIONES

- De las pruebas realizadas se puede apreciar que la confiabilidad del sistema frente a ataques de identidad es alto; es decir que el 100% de las huellas no válidas son rechazadas. Del mismo modo se aprecia que el porcentaje de huellas válidas no aceptadas por el sistema es comparable al porcentaje de rechazo de un cajero automático por un error en el PIN.
- El sistema desarrollado permite tener mayor seguridad a nivel de cajeros automáticos, ya que la persona tiene que estar físicamente en el sitio para validar su identidad. En caso de robo de una tarjeta ésta es inútil (en donde el ladrón de alguna forma consigue la clave) ya que no se tiene la huella de la persona con lo que no se puede hacer ninguna transacción.
- Cuando una persona retira dinero del cajero automático o realiza una transacción en el cajero, el banco tiene la seguridad de qué persona es la propietaria de la tarjeta, ya que sólo su huella se puede registrar en el sistema. Es decir con el sistema de huellas dactilares el banco tiene la seguridad que la persona es quien dice ser.
- Por las características climatológicas de nuestro país, así como por la costumbre alimenticia de nuestro pueblo, los rasgos fisiológicos de nuestra gente y los patrones de huellas son bastante marcados, por ende sistemas de autenticación biométricas basadas en huellas dactilares funcionan bastante bien.
- Si se emplean sistemas de autenticación biométricos basados en huellas dactilares, por la fisiología de nuestro pueblo en sus diferentes patrones, el lector más recomendado es el basado en lectura óptica.

- Por factores climatológicos, los lectores de huellas dactilares basados en Reflexión interna Total Óptica son los mejores para nuestro país, ya que no son susceptibles a cambios climáticos típicos de nuestro medio.
- Los cajeros automáticos son sistemas bastante confiables y seguros para el retiro de dinero, aunque la seguridad en sí dependerá de las precauciones de la persona que hace el retiro. La institución financiera puede incrementar la seguridad en sus sistemas mediante el uso de cámaras, con la presencia de guardias y obviamente con implementación de sistemas como el expuesto en el presente proyecto de titulación.
- Frente al miedo de las personas de que se les pueda cortar el dedo o la mano para robarles el dinero, se puede realizar una campaña por parte de las instituciones implementadoras, para hacer notar el hecho que los lectores que se emplean (lectores ópticos) no detectan dedos muertos, plásticos o latex.
- Los sistemas de comunicación de instituciones financieras por su naturaleza son seguros y aunque la comunicación entre el host y el servidor (o servidores) se considera confiable ya sea por la seguridad de la red LAN o por el uso de una VPN, para aumentar la seguridad y garantizarla en el sistema desarrollado se emplea una encripción 3DES, elevando aún así la seguridad total del sistema.
- La metodología de diseño de aplicaciones mediante UML permiten modelar un sistema y tener un mapa de la aplicación útil en todas las etapas de la vida del software. En la etapa de diseño permite tener en cuenta todos los aspectos que se deben diseñar y prever (lo que ayuda en el robustecimiento del diseño), en la etapa de implementación ya que permite tener una guía para ir creando los diferentes módulos y en la etapa de operación ya que bajo cualquier eventualidad se tiene la guía y el mapa de la aplicación para determinar dónde puede estar el posible punto de falla o la guía para modificar la aplicación en caso de tener un nuevo requerimiento.

4.2 RECOMENDACIONES

- Aunque las pruebas realizadas en laboratorio dieron resultados positivos, antes de comercializar el producto o ponerlo en producción, sería recomendable realizar pruebas en un ambiente más agresivo como tener un cajero en un lugar remoto para poder analizar los retardos de tiempo que se tienen por motivos de propagación.
- La capacitación del sistema, tanto para los usuarios administrativos así como para los clientes es muy importante previa la puesta en marcha, ya que sólo así se puede garantizar un correcto uso del mismo y en posterior el éxito del sistema.
- Hasta conseguir la difusión del sistema, es recomendable no forzar a los clientes a usarlo sino más bien manejarlo como un proyecto piloto y pedir voluntarios para su uso. Esto básicamente para eliminar los miedos de las personas frente su uso y también eliminar la asociación de las huellas dactilares a hechos legales o forenses.
- En caso que se implemente el proyecto para el cobro del bono de desarrollo, se puede implementar ayudas adicionales para tener una mayor penetración. Por ejemplo, se puede implementar ayudas auditivas en el idioma de preferencia de la persona (el idioma es uno de los valores que devuelve el servidor en su trama de respuesta) o se puede modificar los teclados para que tengas colores y sean de fácil distinción para las personas analfabetas.
- La base de datos y el servidor de la aplicación se deben mantener en lugares altamente seguros, ya que poseen información susceptible y privada. Esta información es valiosa y podría comprometer la seguridad de la persona.
- Por el diseño con el cual se implementó la aplicación, su integración con otras aplicaciones es bastante sencillo, por lo que se recomienda que se continúe su investigación para el desarrollo de aplicaciones afines como autenticación de usuarios en ventanillas.

REFERENCIAS BIBLIOGRÁFICAS

LIBROS, MANUALES Y REVISTAS

1. Peterson Kirk. Automated Teller Machine as a National Bank under the Federal Law, William S. Hein & Co., Inc., August 1987.
2. EDS. EDS EFIT ISO 8583 Message Standard Reference. ISO 8583 Message Structure and Definition Pág. 11-40
3. Datos obtenidos de la página de BANRED a septiembre de 2008. Página Web de BANRED: www.banred.fin.ec.
4. Javier Alfonso Villamizar Rivero. (1994). Procesamiento y Clasificación de Huellas Dactilares. Lecturas Matemáticas # 15. Pág. 149-165
5. Pawel Rotter. Las tecnologías de identificación personal: la biometría. Mirando al Futuro Pág. 72 – 75.

A. Jain, L. Hong and R. Bolle. (Abril 1997). On-Line Fingerprint Verification. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol 19, Nº 4, pp. 302-314.

6. L. Hong, Y. Wan and A. Jain. (Agosto 1998). Fingerprint Image Enhancement: Algorithm and Performance Evaluation. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol 20, Nº 8, pp. 777-789.
7. Delgado Albert. (2001). Edición Especial *SQL Server* 2000. PEARSON EDUCACIÓN, S. A. MADRID, 2001. Pág. 116 – 117.
8. Stallings William, (1998), Cryptography and Netword Security, 2da. edición, New Jersey: Prentice Hall Inc.
9. Hallberg Bruce A., (2007), Fundamentos de Redes, 4ta. edición, México: McGraw Hill.
10. Comer, Douglas E., y David L. Stevens. (2001). Internetworking with TCP/IP Volume 3: Client-server programming and applications. Linux/POSIX Sockets Version. New Jersey: Prentice Hall.

11. G. Benton Gibas, Jerry M. Enriquez, Nigel Griffiths, Corneliu Holban, Eunyoung Ko, Yohichi Kurasawa (2004), IBM Eserver pSeries Sizing and Capacity Planning A Practical Guide, IBM
12. Anindya Roy, Saurangshu Kanunjna, Swapnil Arora (2007), How to Choose the Right Server, PCQuest
13. Libro de Red de Área Local del Ingeniero Pablo Hidalgo.
14. Libro de Red de Área Extendida del Ingeniero Pablo Hidalgo.

NORMAS

15. AMERICAN NATIONAL STANDARD FOR INFORMATION SYSTEMS - Data Format for the Interchange of Fingerprint, Facial, & Scar Mark & Tatto. (SMT) Information. NIST Special Publication 500 – 245. ANSI/NIST-ITL 1-2000. Revision of ANSI/NIST-CSL 1-1993 & ANSI/NIST-ITL 1a-1997. www.nist.gov.

PÁGINAS WEB Y REFERENCIAS ELECTRÓNICAS

16. Understanding Biometrics, publicación electrónica de Griaule Biometrics, Año 2008.

    http://www.griaulebiometrics.com/page/en-us/book/export/html/1244.

17. http://es.wikipedia.org/wiki/Operador_Sobel
18. Sistema Evolutivo Reconocedor de Huellas Digitales, Paola Neri, Celia Perez, Araceli Tlamanca. http://paoneri.topcities.com
19. Marcos Faúndez Zanuy, Sergio Osuna Silvestre, http://www.eupmt.es/veu
20. Extracción de características de Galton de Huellas Dactilares por procesamiento digital de la imagen, Baez Moyano Luciano Martin. http://www.luchonet.com.ar/huellas.htm
21. Manual de Desarrollo de *Fingerprint SDK* 2007 de Griaule Biometrics © 2007. Web Site: http://www.griaule.com.
22. http://www.mysql.com/why-mysql/

23. http://www.oracle.com/global/lad/database/index.html

24. http://www.oracle.com/database/database-options.html.

25. http://www.microsoft.com/sql/prodinfo/default.mspx

26. http://www.sdtimes.com/content/article.aspx?ArticleID=30932

27. http://www.redbooks.ibm.com

28. http://pcquest.ciol.com/content/pcshootout/2007/107081601.asp

29. http://www.griaulebiometrics.com/page/en-us/fingerprint_sdk/show_buy_it?height=370&width=450

30. Microsoft Tech Net Library. http://technet.microsoft.com/es-es/library/ms189124.aspx

ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

DESARROLLO DE UNA INTERFAZ BIOMÉTRICA BASADA EN LA LECTURA DE HUELLAS DACTILARES PARA AUTENTICACIÓN DE USUARIOS EN UN CAJERO AUTOMÁTICO

TOMO II

PROYECTO PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRÓNICA Y REDES DE INFORMACIÓN

JUAN FRANCISCO SALCEDO POLANCO

PAOLA CECILIA SEMPÉRTEGUI JÁCOME

DIRECTOR: Ing. PABLO HIDALGO LASCANO

Quito, Agosto 2009

Anexo A: Estándar de NIST

Anexo B: Estructuras de Control y Bucles de Visual Basic

Anexo C: Métodos Esenciales del SDK Griaule Biometrics

Anexo D: Nitgen Fingerprint Reader

Anexo E: Código de la Aplicación

Anexo F: Diccionario de Datos

Anexo G: *Script* de la Base de Datos

Anexo H: Manual de Usuario