UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA

Sistema Integral de Laboratorio Clínico

Sofilab

T E S I S

Que para obtener el título de:

INGENIERO EN COMPUTACION

Presentan:

Marco Polo Manzano Vega

René Montesano Brand

Director: Ing. Sebastián Poblano Ordóñez

México, D.F. **2002**

AGRADECIMIENTOS

A Dios: Por permitirnos llegar a este momento y compartirlo con los seres que más queremos.

A nuestros padres: A quienes les dedicamos especialmente esta tesis por habernos otorgado el don de la vida y más haya de ello por brindarnos su cariño, comprensión y paciencia.

A nuestros familiares: Por el apoyo incondicional que nos han brindado a lo largo de nuestras vidas.

A nuestros amigos: Por su ayuda y comprensión en todo momento.

A nuestros compañeros de trabajo: Por su colaboración en el desarrollo de esta tesis.

Al Ing. Sebastián Poblano: Por su dedicación y paciencia.

A las empresas Sofilab y Lancasart: Por permitirnos participar en un proyecto tan importante.

A nuestros profesores: Por que gracias a ellos hemos podido llegar hasta aquí aplicando sus enseñanzas en la carrera.

A la Universidad Nacional Autónoma de México: Por habernos proporcionado las oportunidades de desarrollo profesional que no encontraríamos en ninguna otra universidad.

ÍNDICE

INTRODUCCIÓN.

Los hospitales cuentan con un laboratorio clínico encargado de realizar las pruebas clínicas que el médico utiliza para la definición de sus diagnósticos. La gran mayoría de estos laboratorios están divididos en secciones para una o varias pruebas específicas, por ejemplo: Inmunología, donde se realizan estudios sanguíneos. La importancia de los laboratorios clínicos se observa en el hecho de que la gran mayoría de los especialistas en medicina requieren de resultados confiables que permitan llevar a cabo una adecuada interpretación acerca de las enfermedades de sus pacientes.

El médico utiliza el laboratorio como medio de ayuda para el diagnóstico y para el tratamiento del paciente, el requerimiento de una determinada prueba es la petición de un servicio de consulta que pone en movimiento una gran cantidad de maniobras, cuyo objetivo es la obtención de un informe analítico. La mayor o menor utilidad de los datos depende de la rápida y precisa comunicación de los resultados. Cada procedimiento dirigido a generar un resultado está constituido por una serie de pasos o fases. La comprensión adecuada de cada fase permite al profesional del laboratorio alcanzar condiciones próximas al nivel óptimo y, en consecuencia, mejorar la exactitud y precisión de cada determinación. La validez de los datos obtenidos en las muestras depende en gran medida de la calidad de la técnica de laboratorio, en la que influyen la correcta manipulación del equipo, el empleo de reactivos lo suficientemente puros y el control ambiental.

La presente tesis propone dar solución a las necesidades de obtención y procesamiento de información en las diversas áreas del laboratorio, para ello se pretende automatizar los procesos antes mencionados, haciendo más fácil su manejo, cabe destacar que se pretende hacer uso de las ventajas proporcionadas por diferentes interfaces diseñadas para cada uno de los equipos de laboratorio clínico. Mediante ellas, se implantará una comunicación de manera directa con la base de datos del sistema, simplificando la obtención de los resultados.

A través de los diversos capítulos contenidos en el presente trabajo, se podrá observar el planteamiento, planeación, desarrollo, construcción e implementación de un sistema que cubra las necesidades antes mencionadas.

Dentro del primer capítulo se define de manera general el problema y su magnitud, estableciendo con ello el objetivo de esta tesis.

El capítulo dos, enmarca el proceso de planeación del sistema, originado a partir de los datos proporcionados por el laboratorio clínico con respecto a la tecnología empleada y la compatibilidad con las interfaces de los diferentes equipos con los que cuenta actualmente el laboratorio para realizar sus operaciones.

En el capítulo tercero, se presenta la aplicación de los conocimientos de ingeniería de software, bases de datos, computadoras y programación, comunicaciones digitales, y redes

de computadoras para lograr un diseño robusto y confiable antes de comenzar la construcción de manera física del sistema.

El capítulo cuarto representa la parte más significativa de este trabajo, debido a que explica detalladamente cada uno de los módulos que integran el sistema y sus interfaces.

Para finalizar, el capítulo quinto corresponde a la evaluación y conclusiones del sistema, mostrando el análisis sobre su funcionamiento y aplicación mediante pruebas de campo; por lo que con base en los resultados se puede retroalimentar y desarrollar las adecuaciones necesarias al sistema, lo que redundará en su crecimiento y optimización de nuevas versiones.

CAPÍTULO I.
ANTECEDENTES.

En este primer capítulo se establecen las condiciones que involucran este proyecto de tesis; es decir se pretende familiarizar al lector con el área donde se implantará el sistema, así como las necesidades de éste, resultando esto en la definición del objetivo del presente trabajo.

1.1 Laboratorio clínico.

Para poder comprender perfectamente los temas tratados en la presente tesis es necesario detallar la función del laboratorio.

Los hospitales son la principal institución de la medicina profesional, son entidades muy complejas que requieren de diversas áreas para su funcionamiento, tales como: jefatura, recepción, entrega de resultados, inmunología, microbiología y otras. En el presente proyecto nos concentraremos en la del laboratorio clínico.

La función del laboratorio clínico es efectuar análisis cualitativos y cuantitativos de líquidos del cuerpo, como sangre, orina y líquido cefalorraquídeo, al igual que de residuos fecales y otras sustancias. Para que los resultados de estos análisis sean útiles al médico en el diagnóstico y tratamiento de enfermedades han de efectuarse los análisis lo más exacto posible. Esto implica el uso de métodos analíticos y tecnología de punta.

En la actualidad un laboratorio clínico está estructurado de forma general de la siguiente manera: cuenta con secciones distribuidas en una o más plantas, una sala de espera, salas de extracciones, baños, áreas de recepción, administración, dirección y depósito de muestras, áreas de hemostasia, química clínica, inmunológica, microbiología, bacteriología y urgencias. Así mismo cuenta con un sistema de refrigeración frío/calor central, con la iluminación y ventilación natural requerida para estas instalaciones.

Para el caso del presente trabajo, se diseño una aplicación acorde con las necesidades del laboratorio clínico del Hospital Juárez de México, el cual cuenta con el siguiente equipo: un analizador de muestras sanguíneas *Hitachi 912*, un analizador de muestras de orina *Clinitek 500*, incluye espectrofotómetros digitales y analógicos, microscopios binoculares, balanzas electrónicas, centrífugas, baños termostatizables, heladeras, estufas de cultivo, estufas esterilización y densitómetros.

1.2 Sofilab S.A. de C.V.

Sofilab es una compañía cuya actividad predominante es la compra-venta, importación y distribución de productos químicos para laboratorio y banco de sangre; ha prestado sus servicios a varias instituciones como el *Hospital Juárez de México*, siendo distribuidor de empresas como *Hitachi*, la que ha ofrecido tecnología de punta a sus clientes.

Sofilab ofrece a sus consumidores la ventaja de que en la compra de sus productos se les proporcionará un sistema integral para el laboratorio clínico, así como interfaces para los equipos de laboratorio.

1.3 Planteamiento del problema y valoración de la situación actual.

Actualmente, un paciente que requiera de los servicios de laboratorio primero debe acudir al área de recepción, donde se capturan sus datos, incluyendo las pruebas del laboratorio que el médico le asignó, en días subsecuentes, deberá presentarse para entregar las muestras en la sección destinada para ello, dichas muestras, serán analizadas por el equipo y/o el personal de laboratorio, este personal por lo general, registra los resultados a mano para después ser entregados al paciente, quien deberá devolverlos a su médico, este proceso es muy lento por lo que pueden pasar semanas antes de que el médico tenga los resultados de un paciente.

Los médicos, afrontan muchas dificultades para poder obtener resultados confiables de las pruebas del laboratorio; ocurren muchos procesos, desde el momento en que el médico solicita una prueba hasta que recibe el resultado para su interpretación; es importante recordar que todos estos procesos requieren tiempo y que están sujetos a errores. No obstante, para que sean clínicamente útiles, los resultados de una prueba, deben ser confiables, precisos, oportunos y de fácil de interpretación. Es útil agrupar los errores que pueden ocurrir en la elaboración de los datos de laboratorio de la siguiente forma:

Errores prelaboratorio: pueden ocurrir en las solicitudes de pruebas, la preparación del paciente y la obtención y manejo de las muestras. Estos errores, incluyen, interpretación errónea de las órdenes por parte de la enfermera, solicitudes de los tiempos erróneos de pruebas, identificación errónea del paciente, preparación inadecuada del enfermo y mala conservación de las muestras.

Errores en el laboratorio: es posible que ocurran al llevar a cabo las pruebas, por ejemplo, métodos de laboratorio que no sean exactos ni precisos, errores de cálculo y errores de trascripción.

Errores postlaboratorio: ocurren durante el traslado de los resultados, como ejemplo: informes ilegibles, falta de intervalos de referencia apropiados y formas inadecuadas para el informe de los resultados.

Con base en lo anteriormente mencionado, se puede establecer que las necesidades de un laboratorio clínico promedio, se enfocan en los siguientes puntos:

- Un control de citas y administración de pacientes con fechas definidas o de emergencias.
- Interactuar con los equipos del laboratorio clínico de tal manera que se agilice la obtención de los resultados.
- La obtención de expedientes que permita el control de cada paciente, así como la visualización de su historial clínico.
- La realización de consultas con el fin de obtener un control de calidad de todos los análisis realizados.
- Poseer un módulo de control y generación de resultados, el cual debe tener un acceso protegido que garantice la confiabilidad del proceso.
- Facilitar la generación de respaldos oportunos y confiables.
- La generación de reportes puede ser total o selectiva, indicando el tipo de paciente, por admisión o por exámenes.
- Permitir un manejo histórico referenciado por pacientes o especialidad.
- Facilitar la consulta de datos históricos.

Valoración de la situación actual.

El sistema que actualmente ofrece *Sofilab* esta basado en *Microsoft Access versión 6.0* y esta diseñado para laboratorios pequeños, con el fin de registrar e imprimir las pruebas que se procesan en el laboratorio clínico.

Las crecientes necesidades del laboratorio obligaron al sistema a incrementar sus módulos y funciones, en consecuencia, la capacidad del manejador de datos (*MS Access*) ha sido superada, esto expresó la clara necesidad de un sistema de base de datos, más robusto para el manejo del laboratorio. Para la selección de éste, deben involucrarse factores como calidad, estabilidad, rendimiento y seguridad.

Debido a que el diseño del sistema actual, no contempla el servicio a varios usuarios, es necesario, realizar replicas de la información en cada maquina del laboratorio, lo que provoca que este proceso sea tardado y poco confiable.

La impresión de reportes es lenta cuando el volumen de datos es muy alto, además, el uso de replicas no ofrece una alta confiabilidad de los mismos.

En el sistema actual, no se cubren adecuadamente las necesidades del laboratorio, debido a que no fue diseñado para soportar las grandes cantidades de información que ahí se manejan, ni tampoco, contempla la utilización de interfaces para la interacción con el sistema, además, es necesario implantar una forma más eficiente de obtención de los reportes, así como ofrecer una manipulación de datos con fines estadísticos.

Como resultado del presente estudio, se hace evidente la necesidad de diseñar un sistema que satisfaga los requerimientos específicos del hospital y que fortalezca su control administrativo y operacional.

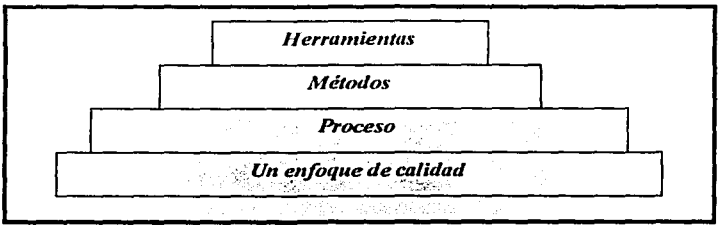
Para resolver estos problemas, el nuevo sistema debe de estar pensado, para poder manejar múltiples usuarios y para aprovechar al máximo la red del laboratorio ya existente. La arquitectura, cliente/servidor, ofrece una mayor confiabilidad y un mejor desempeño general del sistema, además, de que su crecimiento en módulos se realiza de forma casi transparente para el usuario, al realizarse los cambios del manejo de datos en el servidor.

Para establecer un sistema que realmente cumpla con lo anteriormente planteado y se pueda ofrecer un desarrollo profesional del proyecto, nos veremos en la necesidad de apoyarnos en la tecnología que nos ofrece la ingeniería de software, la cual explicaremos con mas detalle en el siguiente punto.

1.4 Ingeniería del software.

La ingeniería del software, en palabras de Fritz Bauer, "es *el establecimiento y uso de principios robustos de la ingeniería a fin de obtener económicamente software que sea fiable y que funcione eficientemente sobre máquinas reales*"[1], aunque esta definición, es aceptable, a primera vista, no dice mucho sobre los aspectos técnicos de la calidad del software, no se enfrenta directamente con la necesidad de la satisfacción del cliente o de la entrega oportuna del producto; omite la mención de la importancia de mediciones y métricas; así como tampoco expresa la importancia de un proceso avanzado. Y sin embargo, la definición de Bauer nos proporciona una línea base. De donde preguntamos, ¿Cuáles son "los principios robustos" de la ingeniería aplicables al desarrollo de software para computadora? ¿Cómo construimos el software "económicamente" para que sea fiable? Éstas son cuestiones, que representan el reto para los ingenieros del software.

La ingeniería de software es una tecnología multicapa (Figura 1). Por lo que, no debemos olvidar que cualquier enfoque de la ingeniería (incluida la ingeniería del software), debe estar basada en un empeño de organización de calidad.



Capas de Ingeniería de Software
Figura 1.

[1] ***Ingeniería del Software***, Pressman Roger, Cuarta edición, Editorial McGraw Hill, Pag. 17-19

De lo expuesto se advierte que, el fundamento de la ingeniería de software es la capa proceso, y el proceso de la ingeniería del software, es la unión que mantiene juntas las capas de tecnología y que permite un desarrollo racional y oportuno de la ingeniería. Por lo que, el proceso en cita, define un marco de trabajo, para un conjunto de *áreas claves de proceso*, que se debe establecer para la entrega efectiva de la tecnología de la ingeniería de software.

Los métodos de la ingeniería de software, indican, como construir técnicamente el software. Estos métodos, abarcan una gran gama de tareas, que incluyen el análisis de requisitos; como los son el diseño; construcción de programas; pruebas y mantenimiento.

Ahora bien, las herramientas de la ingeniería del software, proporcionan un soporte automático o semi-automático para el proceso y para los métodos. Por lo que cuando se integran herramientas para que la información creada por una de éstas herramientas pueda ser utilizar por otra, estableciéndose un sistema de soporte para el desarrollo del software, llamado *Ingeniería del Software Asistida por Computadora (Computer-Aided Software Engineering CASE)*. La que combina software, hardware y una base de datos de ingeniería de software (un depósito que contiene información importante sobre el análisis, el diseño, y la construcción de programas y pruebas), las que crean un entorno de ingeniería del software, que sea análogo a *CAD/CAE (Computer-Aide Desing/Engineering) (Diseño / Ingeniería Asistida por Computadora)* para el hardware.

Para construir la ingeniería del software adecuadamente, se debe de definir un proceso o método de desarrollo, dicho proceso, se puede dividir en tres fases genéricas, con independencia del área de la aplicación, tamaño o complejidad del proyecto.

La primera es la *fase de definición* la cual se centra sobre el *¿qué?*. Es decir, durante la definición, se intenta identificar ¿qué información ha de ser procesada?, ¿qué función y rendimiento se desea?, ¿qué comportamiento del sistema?, ¿qué interfaces van a ser establecidas?, ¿qué restricciones de diseño existen?, y ¿qué criterios de validación se necesitan para definir un sistema correcto?.

Por lo tanto, deben de identificarse los requisitos clave del sistema y del software. Aunque los métodos aplicados durante la fase de definición varían, dependiendo de los paradigmas de ingeniería de software que se apliquen, éstos darán lugar a tres tareas principales: ingeniería de sistemas o de información; planificación del proyecto del software y análisis de los requisitos.

La siguiente es la fase *de desarrollo* que se centra en el *¿cómo?*. Es decir, durante su desarrollo, un ingeniero del software intenta definir ¿cómo han de diseñarse las estructuras de datos?, ¿cómo ha de construirse la función como una arquitectura del software?, ¿cómo han de implantarse detalles sobre los procedimientos?, ¿cómo han de caracterizarse las interfaces?, ¿cómo ha de traducirse el diseño en el lenguaje de programación? y ¿cómo deben de realizarse las pruebas?. Estos métodos aplicados durante la fase de desarrollo, variarán, no obstante que las siguientes tareas ocurrirán siempre; en el diseño del software, la generación de código y la prueba del software.

Finalmente, la *fase de mantenimiento* se centra en el *cambio* que va asociado a la corrección de errores, a las adaptaciones requeridas a medida que evoluciona el entorno del software, y a cambios debidos a las mejoras producidas por los requisitos cambiantes de las necesidades del cliente. Dentro de la fase de mantenimiento, se vuelven a aplicar los pasos de las fases de definición y desarrollo, pero en el contexto del software éstas ya existen. Durante la fase de mantenimiento se encuentran cuatro tipos de cambios.

- Corrección. el mantenimiento correctivo modifica el software para corregir los defectos.
- Adaptación. el mantenimiento adaptativo produce modificación en el software para acomodarlo a los cambios de su entorno externo.
- Mejora. se lleva el software más allá de los requisitos funcionales originales.
- Prevención. también llamado reingeniería del software, es hacer cambios en programas de computadora a fin de que se pueda corregir, adaptar y mejorar más fácilmente.

1.5 Propuesta de solución.

De acuerdo con la valoración de la situación actual predominante, debe decirse que el equipo disponible, hardware y software, es el indicado para determinar la necesidad de un sistema integral que permita satisfacer de manera óptima los procesos del laboratorio clínico, en este sistema se propone una forma de automatización, que apegada a los estándares de desarrollo de la empresa *Sofilab S.A. de C.V.* y a los conocimientos adquiridos en la carrera de ingeniería en computación, ofrezca una solución adecuada a los problemas antes mencionados, utilizando lo último en tecnología de dichos estándares, con lo cual determinamos los siguientes requerimientos:

- Sistema basado en una arquitectura cliente/servidor.
- Plataforma de desarrollo para servidor: *MS SQL Server* bajo *MS Windows NT.*
- Una plataforma de desarrollo para cliente: *MS Visual Basic versión 6* para *Windows 95, 98.*
- Base de datos relacional y normalizada que contenga la información principal del sistema; es decir la información relativa a pacientes, resultados de laboratorio y almacén.
- Ambiente gráfico que desplegará en forma amigable y práctica los grupos de resultados de las diversas áreas con lo que se incluye los diversos exámenes que cada laboratorio realice.
- Manejo de diferentes niveles de usuario: consulta, actualización de resultados, definición y cambio de parámetros, usuarios médicos, químicos encargados de proceso, jefatura y encargado de almacén.
- Interactuar con 10 interfaces que permitirán el mejor flujo de los resultados clínicos con la base de datos mediante la generación automática de listas de producción y sea capaz de enviar esta lista automáticamente a cualquier sistema de análisis clínico.

- Facilitar la exportación de datos hacia productos basados en *Microsoft Office*.
- Poseer un módulo de control y validación de los resultados, el cual puede estar protegido por códigos de seguridad garantizando la confiabilidad del proceso.
- Facilitar la generación de respaldos oportunos y configurables por el usuario.
- Permitir el manejo de citas, control de almacén, y resultados clínicos a través de interfaces para los equipos de laboratorio, lo anterior se realiza interactuando con la base de datos del sistema.
- Generación de reportes para los diversos módulos que maneja el sistema.
- Garantizar una funcionalidad de 24 hrs. al día.

1.6 Objetivo.

Proporcionar la automatización administrativa de las diversas áreas de un laboratorio clínico, mediante la comunicación vía interfaces de los diversos equipos de dicho laboratorio, hacia una base de datos central que permita el control y manejo de la información relativa a pacientes, exámenes, existencia de reactivos en almacén, así como los respectivos reportes que se generen.

CAPÍTULO II.
PLANIFICACIÓN.

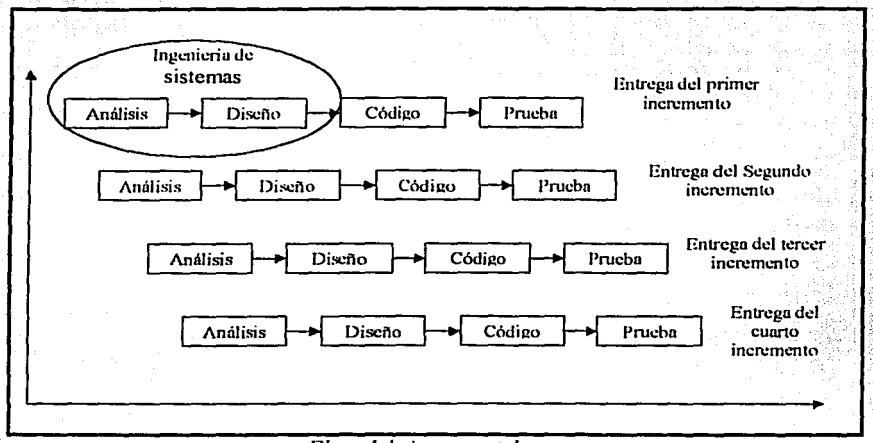
El objetivo de la planificación del proyecto, es proporcionar un marco de trabajo que permita al gestor hacer estimaciones razonables de recursos, costos, y planificación temporal.

2.1 Método de desarrollo.

Como se menciona anteriormente, dentro de la ingeniería de software, es necesario, establecer un método para el diseño del sistema, después de evaluar los diversos métodos que esta tecnología nos ofrece, tales como el modelo lineal, el modelo de prototipos, el modelo de espiral, entre otros, se ha determinado, que el que más se apega a las necesidades del sistema, que a esta tesis concierne, es el modelo incremental, el cual se detalla a continuación.

2.1.1 El modelo incremental.

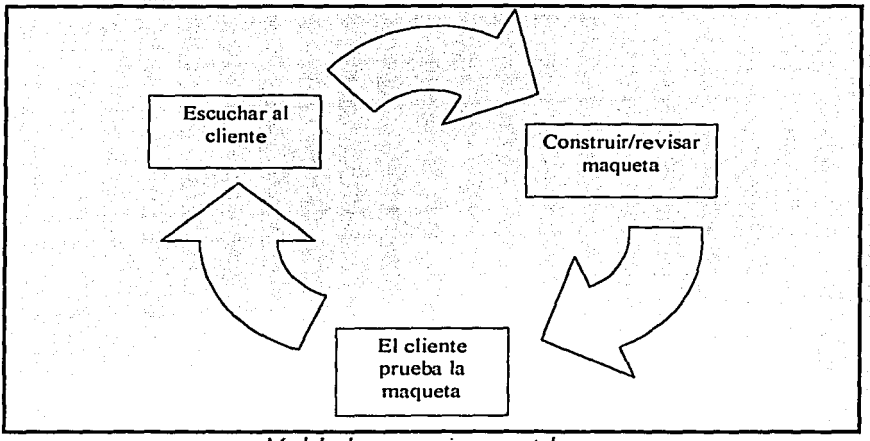
El modelo incremental, combina elementos del modelo lineal secuencial (aplicados respectivamente), con la filosofía interactiva de construcción de prototipos, como se muestra en la figura 2, el modelo incremental, aplica secuencias lineales de manera sorprendente de la misma forma que progresa el tiempo en el calendario, cada secuencia lineal, produce un incremento del software, por ejemplo; el software de tratamiento de texto desarrollado, con el paradigma incremental, podría extraer funciones de gestión de archivos básicos y de producción de documentos, en el primer incremento, funciones de edición más sofisticadas y de producción de documentos; en el segundo incremento, corrección ortográfica y gramatical; en el tercero, debe considerar, que el flujo del proceso de cualquier incremento puede incorporar el paradigma de construcción de prototipos.



El modelo incremental.
Figura 2.

Cuando se utiliza un modelo incremental, el primer incremento o menudo, es un producto esencial (*núcleo*); es decir, se afrontan requisitos básicos, pero muchas funciones suplementarias quedan sin extraer. El cliente utiliza el producto central, como resultado de utilización y/o evaluación, se desarrolla un plan para el incremento siguiente; éste afronta la modificación del producto central, a fin de cumplir mejor las necesidades del cliente y la entrega de funciones y características adicionales, este proceso, se repite siguiendo la entrega de cada incremento, hasta que se elabore el producto completo.

El modelo de proceso incremental, consistente en la construcción de prototipos (Figura 3) y otros enfoques evolutivos, es interactivo por naturaleza, pero a diferencia de la construcción de prototipos, el modelo incremental, se centra en la entrega de un producto operacional con cada incremento, los primeros incrementos, son versiones desmontadas del producto final, pero proporcionan una plataforma para la evaluación por parte del usuario.



Modelo de proceso incremental.
Figura 3.

2.2 Arquitectura cliente/servidor.

El concepto de cliente/servidor proporciona una forma eficiente de utilizar todos estos recursos de máquina, de tal forma, que la seguridad y fiabilidad que proporcionan los entornos *mainframe*[2], se traspasa a la red de área local, a esto hay que añadir la ventaja de la potencia y simplicidad de los ordenadores personales[3].

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

Éste es el modelo de interacción más común entre aplicaciones en una red, debe decirse que no forma parte de los conceptos de la Internet, como los protocolos *IP*, *TCP* o *UDP*, sin embargo, todos los servicios estándares de alto nivel propuestos en Internet funcionan según este modelo.

Los principales componentes del esquema cliente/servidor son entonces, los clientes, los servidores y la infraestructura de comunicaciones.

[2] ***Mainframe:*** Equipos destinados para trabajo pesado.
[3] http://www.mup.es.csi/silice/Global71.html

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

Los clientes interactúan con el usuario comúnmente en forma gráfica, frecuentemente, se comunican con procesos auxiliares que se encargan de establecer conexión con el servidor, envían el pedido, reciben la respuesta, manejan las fallas y realizan actividades de sincronización y de seguridad.

Los clientes realizan generalmente, funciones tales como:

- Manejo de la interfaz del usuario.
- Captura y validación de los datos de entrada.
- Generación de consultas e informes sobre las bases de datos.

Los servidores proporcionan un servicio al cliente y éstos devuelven los resultados, en algunos casos, existen procesos auxiliares que se encargan de recibir las solicitudes del cliente, verificar la protección, activar un proceso servidor para satisfacer el pedido, recibir su respuesta y enviarla al cliente; además, deben manejar los inter-bloqueos, la recuperación ante fallas y otros aspectos afines. Por las razones anteriores, la plataforma computacional asociada con los servidores, es más poderosa que la de los clientes, por esta razón se utilizan PC'S poderosas, siendo éstas estaciones de trabajo, mini computadores o sistemas grandes, además deben manejar servicios como administración de la red, recuperación y contabilidad, mensajes, control y administración de la entrada al sistema ("*login*") y auditoria. Usualmente en los servidores existe algún tipo de servicio de bases de datos, en ciertas circunstancias, este término designará a una máquina, siendo el caso si dicha máquina está dedicada a un servicio particular, por ejemplo: servidores de impresión, servidor de archivos, servidor de correo electrónico, etc.

Por su parte, los servidores realizan entre otras, las siguientes funciones:

- Gestión de periféricos compartidos.
- Control de accesos concurrentes a bases de datos compartidas.
- Enlaces de comunicaciones con otras redes de área local o extensa.
- Siempre que un cliente requiere un servicio lo solicita al servidor correspondiente y éste, le responde proporcionándolo, normalmente, pero no necesariamente, el cliente y el servidor están ubicados en distintos procesadores, los clientes se suelen situar en ordenadores personales y/o estaciones de trabajo y los servidores en procesadores departamentales o de grupo.

Para que los clientes y los servidores puedan comunicarse, se requiere una infraestructura de comunicaciones, la cual proporciona los mecanismos básicos de direccionamiento y transporte, la mayoría de los sistemas cliente/servidor actuales, se basan en redes locales y por lo tanto utilizan protocolos no orientados a conexión, lo cual implica que las aplicaciones deben establecer las verificaciones. La red, debe tener características adecuadas de desempeño, confiabilidad, transparencia y administración.

Entre las principales características de la arquitectura cliente/servidor se pueden destacar las siguientes:

- El servidor presenta a todos sus clientes una interfaz única y bien definida.
- El cliente no necesita conocer la lógica del servidor, sólo su interfaz externa.
- El cliente no depende de la ubicación física del servidor, ni del tipo de equipo físico en el que se encuentra, ni de su sistema operativo.
- Los cambios en el servidor implican pocos o ningún cambio en el cliente.

Como ejemplos de clientes, pueden citarse interfaces de usuario para enviar comandos a un servidor, *API's* (*Application Programing Interface / Interfaz de Programación de Aplicación*), para el desarrollo de aplicaciones distribuidas, herramientas en el cliente para acceder a servidores remotos o aplicaciones que solicitan acceso a servidores para algunos servicios.

Como ejemplos de servidores pueden citarse, servidores de ventanas como *X-Windows*, servidores de archivos como *NFS*, servidores para el manejo de bases de datos (como los servidores de *SQL*), servidores de diseño y manufactura, asistidos por computador, etc.

Una infraestructura cliente/servidor, consta de tres componentes esenciales, todos ellos de igual importancia y estrechamente ligados:

Plataforma operativa: ésta deberá soportar todos los modelos de distribución cliente/servidor, todos los servicios de comunicación y deberá utilizar, preferentemente componentes estándar de la industria para los servicios de distribución. Los desarrollos propios, deben coexistir con las aplicaciones estándar y su integración deberá ser imperceptible para el usuario, igualmente podrán acomodarse programas escritos utilizando diferentes tecnologías y herramientas.

Entorno de desarrollo de aplicaciones: debe elegirse después de la plataforma operativa, aunque es conveniente evitar la proliferación de herramientas de desarrollo, se garantizará que el enlace entre éstas y el "*middleware*[4]" no sea excesivamente rígido, siendo posible utilizar diferentes herramientas para desarrollar partes de una aplicación. Un entorno de aplicación incremental, debe posibilitar la coexistencia de procesos cliente y servidor desarrollados con distintos lenguajes de programación y/o herramientas, así como utilizar distintas tecnologías, por ejemplo, lenguaje procedural, lenguaje orientado a objetos, multimedia, etc. que han sido puestas en explotación en distintos momentos del tiempo.

Gestión de sistemas: estas funciones aumentan considerablemente el costo de una solución, pero no se pueden evitar, siempre deben adaptarse a las necesidades de la organización y al decidir la plataforma operativa y el entorno de desarrollo; es decir en las primeras fases de la definición de la solución.

[4] *Middleware*: Software que se encuentra entre dos o más tipos de software y traduce información entre ellos.

Existe un conjunto de variantes de la arquitectura cliente/servidor dependiendo de dónde se ejecutan los diferentes elementos involucrados:

1. Administración de los datos
2. Lógica de la aplicación
3. Lógica de la presentación.

Presentación distribuida. En esta primera variante que tiene algún interés es donde, tanto la administración de los datos, como la lógica de la aplicación, funcionan en el servidor, y la lógica de la presentación se divide entre el servidor (parte preponderante) y el cliente (donde simplemente se muestra).

Esta opción es extremadamente simple porque generalmente no implica programación alguna ¿qué se obtiene con ello? una mejor presentación, desde el punto de vista estrictamente cosmético y ciertas capacidades mínimas para vincular las transacciones clásicas con el entorno *Windows*; desde el punto de vista del uso de los recursos, esta primera alternativa es similar a la arquitectura centralizada.

Administración remota de datos. Esta es una segunda opción plausible, donde la administración de los datos se hace en el servidor, mientras que tanto la lógica de la aplicación como la de la presentación, funcionan en el cliente.

Desde el punto de vista de las necesidades de potencia de procesamiento, esta variante es la óptima, se minimiza el costo del procesamiento en el servidor (sólo se dedica a administrar la base de datos, no participando en la lógica de la aplicación que, cada vez consume más recursos), mientras que se aumenta en el cliente donde es irrelevante, teniendo en cuenta las potencias de cliente necesarias de todas maneras para soportar el sistema operativo *Windows*.

El otro elemento a tener en cuenta es el tránsito de datos en la red, esta variante podrá ser óptima, buena, mediocre o pésima, de acuerdo a este tránsito.

En el caso de transacciones o consultas activas, donde prácticamente todos los registros seleccionados son necesarios para configurar las pantallas a mostrar, este esquema es óptimo.

Por otro lado, en el caso de los programas de procesamiento por lotes, donde en realidad no se muestra nada, esta alternativa es teóricamente indefendible (no obstante, si el cliente está ligado al servidor por una red de alta velocidad, los resultados prácticos, a menudo, son aceptables). Una variante interesante es la de complementar el procesamiento en el cliente con procesamiento en el servidor.

Este objetivo se puede abordar de dos maneras diferentes:

La primera es el uso de procedimientos almacenados y disparadores asociados al servidor de base de datos.

Como la mayor parte de los mecanismos utilizados en la arquitectura cliente/servidor, estos elementos fueron introducidos por *Sybase* al final de la década de los 80 y consisten en:

Disparadores (Triggers): sucesos que se ejecutan cuando se detectan ciertos estados en la base de datos. Su función es permitir la implementación de reglas permanentes, su uso más típico ha sido el de proteger la integridad referencial de la base de datos (esa función hoy es cumplida por las capacidades declarativas de definir dicha integridad referencial que tienen actualmente todos los servidores importantes). El disparador llama a los procedimientos almacenados que se han programado previamente para atender a cada uno de dichos eventos.

Procedimientos almacenados (Stored Procedures): son programados para cumplir la parte de la lógica de la aplicación que se desea se ejecute en el servidor. Un procedimiento almacenado puede ser llamado por otro de ellos, por un disparador o directamente desde el cliente, mediante una llamada remota ("*remote call*").

Este esquema representa un avance importante en la distribución de la lógica de la aplicación entre el cliente y el servidor que, sin embargo, presenta un conjunto de particularidades indeseables:

No existe un estándar de lenguaje para la formulación de procedimientos almacenados (el original, definido por *Sybase*, es el *Transact SQL* y diferentes derivaciones del mismo son utilizados por *Sybase* y *Microsoft*, *Oracle* cuenta con el llamado *PL/SQL*, *Informix* también tiene el *Informix 4GL*, mientras que *IBM* no tiene un lenguaje específico, etc.).

2.2.1 Ventajas y desventajas de la arquitectura cliente/servidor.

1) Ventajas

1.1) Para el cliente

- Mediante la integración de las aplicaciones cliente/servidor con las aplicaciones personales de uso habitual (como hojas de cálculo y herramientas de acceso a bases de datos), los usuarios pueden construir soluciones particulares que se ajusten a sus necesidades cambiantes.
- Una interfaz gráfica consistente reduce el tiempo de aprendizaje de los usuarios.
- Permiten un mejor aprovechamiento de los sistemas existentes protegiendo la inversión, por ejemplo, la posibilidad de compartir servidores (habitualmente caros) y dispositivos periféricos (como impresoras) entre máquinas clientes, permite un mejor rendimiento del conjunto.
- Los usuarios podrán disponer de la información del sistema en el momento en que lo deseen sin tener que esperar largos periodos de tiempo a que ésta última sea actualizada.

- Proporcionan un mejor acceso a los datos. La interfaz de usuario ofrece una forma homogénea de ver el sistema, independientemente de los cambios o actualizaciones que sean producidos en él y de la ubicación de la información.

- El esquema cliente/servidor contribuye además a proporcionar a las diferentes direcciones de una institución soluciones locales, pero permitiendo además la integración de la información relevante a nivel global.

1.2) Para el desarrollador

- Se pueden utilizar componentes para distintas aplicaciones, tanto de hardware como de software, de varios fabricantes, lo cual contribuye considerablemente a la reducción de costos y favorece la flexibilidad en la implantación y actualización de soluciones.

- El movimiento de funciones desde un ordenador central hacia los servidores o clientes locales, origina el desplazamiento de los costos de ese proceso hacia máquinas más pequeñas y por tanto, más baratas.

- Las arquitecturas cliente/servidor eliminan la necesidad de mover grandes bloques de información por la red hacia los ordenadores personales o estaciones de trabajo para su proceso. Los servidores controlan los datos, procesan peticiones y después transfieren sólo los datos requeridos a la máquina cliente, entonces, la máquina cliente presenta los datos al usuario mediante interfaces amigables, todo esto reduce el tráfico de la red lo que facilita que pueda soportar un mayor número de usuarios.

- Es posible integrar PC's con sistemas medianos y grandes, sin que todas las máquinas tengan que utilizar el mismo sistema operativo.

- Si se utilizan interfaces gráficas para interactuar con el usuario, el esquema cliente/servidor presenta la ventaja, con respecto a uno centralizado, de que no es siempre necesario transmitir información gráfica por la red pues ésta puede residir en el cliente, lo cual permite aprovechar mejor el ancho de banda de la red.

- Tanto el cliente como el servidor pueden escalarse para ajustarse a las necesidades de las aplicaciones, los CPU's utilizados en los respectivos equipos pueden dimensionarse a partir de las aplicaciones y el tiempo de respuesta que se requiera.

- La existencia de varios CPU's proporcionan una red más fiable: una falla en uno de los equipos no significa necesariamente que el sistema deje de funcionar.

- En una arquitectura como ésta, los clientes y los servidores son independientes los unos de los otros, con lo que pueden renovarse para aumentar sus funciones y capacidad de forma independiente sin afectar al resto del sistema.

- La arquitectura modular de los sistemas cliente/servidor, permite el uso de ordenadores especializados (servidores de base de datos, servidores de ficheros, estaciones de trabajo para *CAD*, etc.).

- Es más rápido el mantenimiento y el desarrollo de aplicaciones, pues se pueden emplear las herramientas existentes (por ejemplo los servidores de *SQL* o las herramientas de más bajo nivel como los *sockets* o el *RPC*).

2) Desventajas

2.1) Para los usuarios.

- Existen multitud de costos ocultos (como la formación en nuevas tecnologías, licencias, cambios organizacionales, etc.) que podrían encarecer su implantación.

- A veces los problemas de congestión de la red pueden degradar el rendimiento del sistema por debajo de lo que se obtendría con una única máquina (arquitectura centralizada).

2.2) Para los desarrolladores

- Hay una alta complejidad tecnológica al integrar una gran variedad de productos.

Por una parte, el mantenimiento de los sistemas pude ser más difícil pues implica la interacción de diferentes partes de hardware y de software distribuidas por distintos proveedores, lo cual dificulta el diagnóstico de fallas.

- En algunos casos requiere de un fuerte rediseño de todos los elementos involucrados en los sistemas de información (modelos de datos, procesos, interfaces, comunicaciones, almacenamiento de datos, etc.). Además, se debe determinar la mejor forma de dividir las aplicaciones entre la parte cliente y la parte servidor.

Por un lado, es importante que los clientes y los servidores utilicen el mismo mecanismo (por ejemplo sockets o RPC), lo cual implica que se deben tener mecanismos compatibles que existan en diferentes plataformas.

- Es más difícil asegurar un elevado grado de seguridad en una red de clientes y servidores que en otros sistemas, por ejemplo, el de un único ordenador centralizado, debido a que se deben hacer verificaciones en el cliente y en el servidor, también se puede recurrir a otras técnicas como el encriptamiento.

Un aspecto directamente relacionado con el anterior, es el de cómo distribuir los datos en la red. En el caso de una empresa, por ejemplo, éste puede ser hecho por departamentos, geográficamente, o de otras maneras, además hay que tener en cuenta que en algunos casos, por razones de confiabilidad o eficiencia se pueden tener datos replicados y que puede haber actualizaciones simultáneas.

2.3 Sistema operativo.

En los años sesenta, un sistema operativo se podría haber definido como el software que controla al hardware; sin embargo, actualmente existe una tendencia significativa a la transferencia de las funciones del software al *firmware*[5], es decir, microcódigo. Dicha tendencia se ha pronunciado tanto que es probable que en algunos sistemas las funciones codificadas en *firmware* sobrepasen pronto a aquéllas codificadas en software.

Es evidente que se necesita una nueva definición de sistema operativo, se puede imaginar un sistema operativo como los programas instalados en el software o el *firmware*, que hacen utilizable el hardware. El hardware proporciona la capacidad bruta de cómputo; los sistemas operativos ponen dicha capacidad de cómputo al alcance de los usuarios y administran cuidadosamente el hardware para lograr un buen rendimiento.

En general, se puede decir que un sistema operativo se desea que tenga las siguientes características:

- Eficiencia. Permite que los recursos de la computadora se usen de la manera más adecuada posible.
- Habilidad para evolucionar. Debe construirse de manera que permita el desarrollo, prueba o introducción efectiva de nuevas funciones del sistema sin interferir con el servicio.
- Encargado de administrar el hardware. Le corresponde el optimar los recursos de la computadora en cuanto a hardware se refiere, esto es, asignar a cada proceso una parte del procesador para poder compartir los recursos.
- Relacionar dispositivos (gestionar a través del núcleo (*Kernel*)). Se encarga de comunicar a los dispositivos periféricos entre sí.
- Organizar datos para acceso rápido y seguro.
- Manejar las comunicaciones en red. Permite al usuario manejar con gran facilidad todo lo referente a la instalación y uso de las redes de computadoras.
- Procesamiento por "*bytes*" de flujo a través del bus de datos.

[5] ***Firmware:*** Rutinas de software guardadas en memoria de sólo lectura (ROM).

- Facilitar las entradas y salidas. Debe simplificar al usuario el acceso y manejo de los dispositivos de entrada / salida de la computadora.

- Técnicas de recuperación de errores.

- Evita que otros usuarios interfieran. No permite que los usuarios se bloqueen entre ellos, informándoles si esa aplicación está siendo ocupada por otro usuario.

- Generación de estadísticas.

- Permite que se puedan compartir el hardware y los datos entre los usuarios.

2.4 Bases de datos.

La pretensión primera de una base de datos es almacenar información, exactamente igual que los archivos; pero extiende el sentido de estos porque no solo almacena, sino que también estructuran los datos, organizando la información según un cierto modelo.

Este modelo de datos, define las estructuras básicas de información y las relaciones que entre ellas existen. El modelo se explica al crear la base de datos abstrayendo las características de cada entidad; así como las relaciones y dependencias entre las mismas.

El citado esquema se almacena con la propia base de datos, de manera que cualquier programa que la utilice encontrará en su estructura física, entre otras informaciones, los nombres de cada uno de los elementos de datos y el tipo de información que contienen.

La utilidad del esquema radica en que facilita enormemente el manejo de la información. En primer, lugar resulta mucho más sencillo referirnos a los datos según los términos de una cierta abstracción, como ejemplo; hablar de los nombres de los empleados, que especificar su ubicación física en un eventual archivo. Por otra parte, el modelo también facilita el acceso a la información. En un archivo accederemos a los datos dependiendo de su disposición física, mientras que en una base de datos los solicitaremos mediante consultas escritas.

2.4.1 Objetos básicos.

Las bases de datos, independientemente de cuales sean los mecanismos utilizados para almacenar y obtener su información. Estarán compuestas fundamentalmente por tablas.

- **Tabla.** Es una colección de elementos de información agrupados en filas y columnas. La tabla almacena la información sobre una entidad diferenciada, como por ejemplo, los datos de los empleados de una empresa. La base de datos acaba constituyéndose como una colección de tablas relacionadas entre sí.

- **Fila o registro.** Representa un ejemplar del objeto que abstrae la tabla, esto es una instancia de la entidad real de la cual almacenamos datos. Por ejemplo, si

creamos la tabla EMPLEADOS, la información acerca de cada uno de los empleados se almacena en una fila distinta, de manera que cada empleado se halla asociado a un registro. Una tabla resulta de este modo una colección de filas.

- **Columna**. Comúnmente conocidas como campos, describen cada uno de los atributos o características de los objetos que representan los registros de la tabla. Cada columna almacena información homogénea, esto es, del mismo tipo. En la tabla EMPLEADOS, podemos definir la columna NOMBRE que contiene en cada una de sus filas una cadena de caracteres que incluye el nombre del empleado.

- **Celda**. Es la intersección de una fila y una columna y representa el valor de un determinado atributo, el representado por la columna a la que pertenece la celda, para un registro de la tabla.

2.4.2 Modelo relacional de bases de datos.

El modelo relacional revolucionó el mundo de las bases de datos y permitió que las computadoras personales reemplazaran a las mini computadoras y "*mainframes*" en muchas aplicaciones.

Este modelo se apoya en el cálculo racional desarrollado por E. F. Codd en los años 70[6], Su principal aportación radica en que la relación entre los datos deja de expresarse a través de punteros a los datos que deben almacenarse conjuntamente con ellos.

En el modelo relacional, en cambio, los vínculos entre las entidades de datos se establecen mediante la abstracción de relaciones entre distintas tablas. Estas relaciones se obtienen del análisis de la naturaleza intrínseca de los datos y no de las particulares estructuras físicas de almacenamiento.

Una relación consiste en una liga que puede establecerse entre dos tablas que poseen algunos campos con valores idénticos.

Existen ciertas reglas que deben respetarse a la hora de crear bases de datos relacionales.

- **Las filas son independientes:** ninguna fila de una tabla puede depender de ninguna otra de la misma tabla, por lo que no tiene sentido la ordenación alguna de registros en el interior ésta.

- **Las filas deben ser únicas:** dos filas de la misma tabla deben diferir al menos en un valor.

- **Las columnas son independientes:** análogamente a lo que sucede con las filas, las columnas tampoco poseen ningún orden, esto es; no tiene ningún significado

[6] *Visual Basic 6, Manual Completo de Programación*, Aitken Peter, Pag. 507 a 509

desde el punto de vista del modelo y para la eventual ordenación de las mismas en su presentación.

- **Los valores en las columnas deben ser unitarios:** dentro de una determinada fila, cada columna contendrá un único valor, nunca una fila o lista de ellos.

Es aconsejable, aunque no imprescindible, que cada registro posea una serie de atributos, es decir campos, que lo identifiquen de manera inequívoca, a estos campos los conocemos como claves o llaves y en una base de datos relacional existen los siguientes tipos.

- **Llaves primarias** (*Primary Keys*): una llave primaria es una columna que contiene valores distintos para todas las filas de la tabla. De este modo, una llave primaria identifica inequívocamente a cada registro. Ninguna de las celdas de una clave primaria puede quedar vacía.

- **Llaves candidatas** (*Candidate Keys*): en algunas ocasiones las tablas poseen más de una columna individual o grupo de las mismas que podrían conformar una llave primaria pues sus valores son únicos.

- **Llaves foráneas** (*Foreign Keys*): Una llave foránea es una columna o grupo de columnas en una tabla que contienen valores que coinciden con una llave primaria en otra. Estas llaves foráneas permiten establecer relaciones entre tablas que son la base del modelo. Este proceso de enlazar unas tablas con otras da como resultado una tabla combinada o "*join*".

2.5 Software.

El software son herramientas que nos ayudan en el diseño y la construcción de diversas aplicaciones.

Existen una gran variedad de Software aplicable a las necesidades de desarrollo del sistema que nos ocupa (diversos sistemas operativos, manejadores de bases de datos, lenguajes de programación, etc.), con el fin de apegarnos a la normatividad de la empresa Sofilab S.A. de C.V. y de proporcionar la mejor calidad en el sistema, hemos decidido utilizar el siguiente software.

2.5.1 Sistema operativo Microsoft Windows NT.

Windows NT ha sido desde mucho el sistema operativo de terminado de calidad en la familia Windows. Fue diseñado desde el principio de un sistema operativo amigable, seguro y estable para necesidades de servidor intensivas. *Windows NT* es un sistema operativo mucho más estable que *Windows 95* y *98*. Saca la mayor ventaja posible del hardware en la computadora y funciona en una amplia gama de procesadores.

Una de las características eficaces de *Windows NT* es que puede integrar a un número de sistemas operativos dispares (entornos) en la misma red. Por ejemplo, puede usarse para conectar clientes que ejecutan *Windows 95, 98, NT Workstation, OS/2, Macintosh* y *Unix*.

Microsoft también puso a disposición varios complementos para *Windows NT Server* que proporcionan funciones críticas para empresas con necesidades intensivas específicas. Por ejemplo, está disponible un servidor de base de datos basado en *SQL* para necesidades de bases de datos de alto rendimiento y un servidor de Internet para información de publicaciones en Internet.

Debido a la similitud que su interfaz con el usuario tiene con *Windows 95* y *98*, permite al usuario una rápida y sencilla adaptación, lo que convierte a *Windows NT* la plataforma ideal para la ejecución del sistema.

2.5.2 Visual Basic 6.0.

Con la expansión de las aplicaciones desarrolladas en el lenguaje de programación *Visual Basic (VB)* y la maduración de sus funciones, los mecanismos para conectar *VB* a bases de datos también se han multiplicado y evolucionado. *Visual Basic 6.0* no es ninguna excepción; añade soporte para varias metodologías nuevas y un sin fin de nuevos enfoques a los paradigmas existentes. La presión que han ejercido los desarrolladores sobre *Microsoft* ha dado como fruto gran parte de este rendimiento y mayor productividad especialmente a la hora del acceso a datos.

Cuando se introdujo *VB*, fue la primera herramienta de desarrollo que sacó la programación *Windows* del reino de los especialistas. Las ideas del diseño de interfaces arrastrar/soltar y de la programación orientada a eventos son ahora casi universales.

Visual Basic 6 ofrece una potencia incomparable y una gran facilidad de uso para las necesidades de los desarrolladores actuales. En particular, las utilidades para programación con bases de datos, aplicaciones cliente/servidor y el desarrollo en *Internet* proveen la capacidad que el programador ha solicitado.

Visual Basic permite una programación orientada a objetos *(POO)* real, como *C++* y *Java*, aunque no tiene todas las características de un auténtico lenguaje orientado a objetos. El término objeto se utiliza de forma diferente en *VB*, lo que no implica que sea un lenguaje inferior en algún sentido; se trata simplemente de un enfoque distinto de la programación, podemos definir a *VB* como un lenguaje de programación orientado a eventos.

Los eventos son cosas que hace el usuario con el ratón o el teclado mientras el programa se está ejecutando, *VB* contiene una serie de instrucciones que permite reconocer los diversos movimientos que hace el usuario, lo que permite al programa saber cuando debe de ejecutar algo, es decir, un evento es algo que hace que las cosas sucedan en *VB*.

2.5.3 SQL Server 7.0.

SQL significa *Lenguaje de Consulta Estructurado* (*Structured Query Language*), un lenguaje de manipulación de bases de datos estándar desarrollado originalmente por IBM.

SQL surgió en 1974 en el laboratorio de investigación de *IBM* como Lenguaje de Consulta Estructurado en Ingles *(Structured English Query Language)* o *SEQUEL*, desde entonces, el lenguaje ha evolucionado hasta el *SQL* actual.

Existen varios estándares de *SQL* y varios diseñadores han añadido ampliaciones a sus productos concretos de *SQL* para sus programas de bases de datos.

Al contrario que *Basic* y la mayoria de los lenguajes de programación, *SQL* es un lenguaje "procedimental", lo cual significa que *SQL* no contiene sentencias ni construcciones para controlar las secuencias u orden de ejecución del programa. Las sentencias *SQL* se limitan a expresar lo que el usuario quiere hacer; el programa que ejecuta las instrucciones *SQL* interpreta las sentencias y devuelve el resultado.

SQL Server 7.0 es un manejador de bases de datos (*Data Base Manager System DBMS*), desarrollado por *Microsoft*, la versión 7 esta totalmente reestructurada, a diferencia de las versiones anteriores, ahora es posible ejecutar la herramienta bajo una plataforma de *Windows 95* ó *98*, lo que hace posible el no tener que desarrollar prototipos en *Microsoft Access*, las nuevas interfaces que proporciona el programa para la creación y administración de las bases de datos es mas sencilla, lo que da la oportunidad a los desarrolladores a optimar tiempos.

Dentro de las funciones de un *DBMS* se encuentran:

- Almacena, recupera y modifica datos.
- Guarda la consistencia de los datos.
- Soluciona problemas de concurrencia.

Su objetivo principal es crear un ambiente en que sea posible guardar y recuperar información de la base de datos en forma eficiente.

El manejo de los datos del *DBMS* incluye tanto la definición como la manipulación y seguridad de los mismos.

2.5.4 Sheridan.

Las utilerías *Sheridan* son una serie de herramientas de software creadas para facilitar el desarrollo de aplicaciones, entre estas herramientas podemos mencionar controles para acceso a datos y otras que están enfocadas a la presentación de los mismos y de la aplicación en general.

De estas herramientas seleccionamos las siguientes debido a que resultaron ser las más apropiadas para la aplicación.

2.5.4.1 Data Widgets.

El software *Data Widgets* es un conjunto de controles personalizados que permiten diseñar vistas de despliegue para las aplicaciones de la base de datos de manera sencilla, impulsando y proporcionando fuerza en el desarrollo de una aplicación.

Diseñado con la finalidad de un fácil manejo, *Data Widgets* ahorra tiempo en el desarrollo de aplicaciones que involucran el funcionamiento de base de datos. Lo que antes tomaba horas de desarrollo puede tomar minutos ahora, esto se explica debido a que sólo se necesita modificar algunas propiedades en lugar de agregar un nuevo código.

Data Widgets incluye seis controles que permiten personalizar las vistas de la base de datos, cada uno diseñado para una manipulación de datos específica, trabajando en los formatos de 16-bits y 32-bits, basados en *OLE (Object Linking and Embedding/ Vinculación e Incrustación de Objetos)* y formato *OCX (Ole Custom Control / Control Personalizado OLE).*

2.5.4.2 Visual Assist.

La totalidad de los componentes del software *Visual Assist* fueron creados con el fin de asistir al diseñador durante la realización del programa, se dice que refuerza el ambiente de trabajo y esta enfocado a las siguientes categorías.

- Generación de código.
- Manejo de controles.
- Conexiones a bases de datos.
- Herramientas de ambiente.

Una de las principales razones por la que se seleccionó ésta herramienta es debido a que *Visual Assist* facilita el arreglo de los diversos objetos que son visibles en la aplicación, es decir; permite que el desarrollador dedique menos tiempo en procesos como alineación de etiquetas y pueda concentrarse en el desarrollo del código que da funcionalidad al software.

2.5.5 Crystal Reports 8.0.

El software para la elaboración de reportes *Seagate Crystal Reports* está diseñado para trabajar con la base de datos para ayudar a analizar e interpretar información importante. *Seagate Crystal Reports* facilita la creación de informes simples y dispone también de herramientas poderosas para generar informes complejos o especializados.

Esté software incorpora diversos asistentes que facilitan la generación de informes, así como la implementación de fórmulas, tablas cruzadas, subinformes y formatos que permiten una mejor comprensión de los datos.

La flexibilidad de *Seagate Crystal Reports* no termina con la creación de informes, ya que éstos se pueden publicar en una variedad de formatos que incluyen Microsoft Word y Excel, Correo Electrónico e incluso en el Internet. La elaboración avanzada en el Internet permite a otros miembros de su grupo de trabajo ver y actualizar informes compartidos en sus exploradores.

2.5.6 True DBGrid 6.0.

True DBGrid pro 6.0 es un control *ActiveX*[7] para datos diseñado para *Microsoft Visual Studio*. Desarrollado por *APEX Software Corp. True DBGrid pro 6.0* es la última actualización para el control de Visual Basic *DBGrid* adaptado para funcionar con las últimas versiones de *Visual Basic* y *Visual C++*.

True DBGrid permite a los usuarios mostrar, editar, agregar y borrar datos en forma tabular. Usando las últimas tecnologías para la construcción en *Visual Studio*, incluyendo *OLE DB (Object Linking and Embedding for Data Bases Vinculación e Incrustación de Objetos para Bases de Datos)* *True DBGrid* complementa la interfaz con la base de datos permitiendo a los desarrolladores concentrarse en tareas específicas de mayor importancia.

2.6 Hardware.

Está sección pretende dar a conocer de forma general los puntos de aplicación de la presente tesis, como son: las redes y las interfaces.

Debemos recordar que esté trabajo esta enfocado en una arquitectura cliente/servidor y además pretende explotar la capacidad de interfazar con los equipos de laboratorio clínico por lo que se hace necesario ahondar en dichos temas.

2.6.1 Redes de computadoras.

Cuando en 1981 *IBM* presenta la computadora personal (PC), la palabra *personal* era un adjetivo adecuado. Estaba dirigido a las personas que deseaban disponer de su propia computadora, sobre la que ejecutan sus propias aplicaciones, y sobre la que administran sus archivos personales en lugar de utilizar las mini computadoras y grandes sistemas que estaban bajo el estricto control de los departamentos de informática. Los usuarios de las computadoras personales comenzaron pronto a conectar sus sistemas formando redes, de una forma que podrán compartir los recursos como las impresoras. Ocurriendo entonces algo curioso; en 1985 las redes se hicieron tan grandes y complejas que el control volvió a los departamentos de informática. Actualmente las redes no son elementos simples y fáciles, a menudo se llegan a extender fuera de la oficina local, abarcan el entorno de una

[7] ***ActiveX:*** Un conjunto de tecnologías que permiten a los componentes de software integrarse entre sí en un entrono de red, independientemente del lenguaje en el que los componentes fueron creados.

ciudad o uno mayor y necesitan entonces expertos que puedan tratar los problemas derivados de las comunicaciones telefónicas, con microondas o vía satélite.

La más simple de las redes conecta dos computadoras, permitiéndoles compartir archivos e impresoras. Una red mucho más compleja conecta todas las computadoras de una empresa o compañía en el mundo. Para compartir impresoras basta con un conmutador, pero si se desea compartir eficientemente archivos y ejecutar aplicaciones de red, hace falta tarjetas de interfaz de red (*NIC, NetWare Interface Cards*) y cables para conectar los sistemas. Aunque se pueden utilizar diversos sistemas de interconexión vía los puertos seriales y paralelos, estos sistemas baratos no ofrecen la velocidad e integridad que necesita un sistema operativo de red seguro y con altas prestaciones que permita manejar muchos usuarios y recursos.

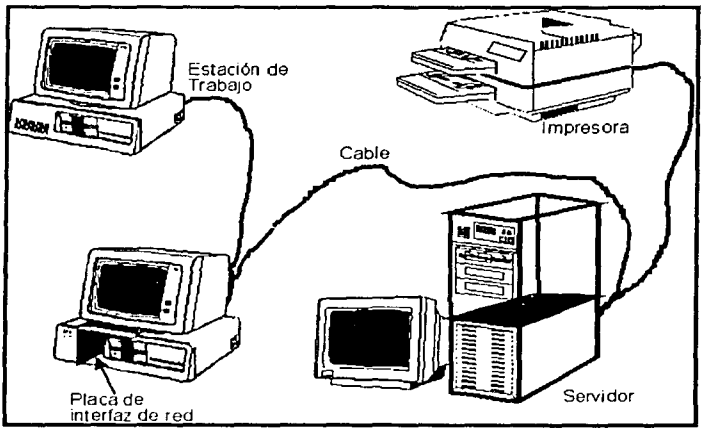


Figura 4.
Componentes típicos de un sistema en red.

Una vez instalada la conexión se ha de instalar el sistema operativo de red (*NOS, Network Operating System*). Hay dos tipos básicos de sistemas operativos de red: punto a punto y con servidor dedicado.

NOS Punto a Punto: este es un tipo de sistema operativo que le permite a los usuarios compartir los recursos de sus computadoras y acceder a los recursos compartidos de las otras computadoras. Microsoft Windows, Novell Lite son sistemas operativos punto a punto.

NOS Servidor dedicado: es un sistema operativo con servidor reservado, como son NetWare de Novell, Windows NT y otros, una o mas computadoras se utilizan como servidores de archivos no pudiendo ser utilizados de otra manera.

2.6.1.1 Componentes de una red.

Una red de computadoras esta constituida tanto por hardware como por software. El hardware incluye las tarjetas de interfaz de red y los cables que las unen, el software incluye los controladores (programas que se utilizan para gestionar los dispositivos y el sistema operativo de red que gestiona la red. A continuación se listan los componentes de una red.

Servidor: este ejecuta el sistema operativo de red y ofrece los servicios de red a las estaciones de trabajo.

Estaciones de trabajo: cuando una computadora se conecta a una red, la primera se convierte en un nodo de la ultima y se puede tratar como una estación de trabajo o cliente. Las estaciones de trabajo pueden ser computadoras personales con el *DOS, Macintosh, Unix, OS/2* o estaciones de trabajo sin discos.

Tarjetas o placas de interfaz de red: toda computadora que se conecta a una red necesita de una tarjeta de interfaz de red que soporte un esquema de red específico, como *Ethernet, ArcNet* o *Token Ring*. El cable de red se conectará a la parte trasera de la tarjeta.

Sistema de cableado: el sistema red está constituido por el cable utilizado para conectar entre si el servidor y las estaciones de trabajo.

Recursos y periféricos compartidos: entre los recursos compartidos se incluyen los dispositivos de almacenamiento ligados al servidor, las unidades de discos ópticos, las impresoras, los ruteadores y el resto de equipos que puedan ser utilizados por cualquiera en la red.

2.6.1.2 Ethernet y Fast Ethernet.

Ethernet es la capa física más popular de la tecnología *LAN (Local Area Network /Red de Área Local)* usada actualmente. Otros tipos de *LAN* incluyen *Token Ring, Fast Ethernet, FDDI, ATM y LocalTalk. Ethernet* es popular porque permite un buen equilibrio entre velocidad, costo y facilidad de instalación. Estos puntos fuertes, combinados con la amplia aceptación en el mercado y la habilidad de soportar virtualmente todos los protocolos de red populares, hacen a *Ethernet* la tecnología ideal para la red de la mayoría de los usuarios de la informática actual. La norma de *Ethernet* fue definida por el Instituto para los Ingenieros Eléctricos y Electrónicos (*IEEE*) como *IEEE Standard* 802.3. Adhiriéndose a la norma de *IEEE*, los equipos y protocolos de red pueden interactuar eficazmente.

Para redes *Ethernet* que necesitan mayores velocidades, se estableció la norma *Fast Ethernet (IEEE 802.3u)*. Esta norma elevó los límites de 10 Mega bits por segundo (Mbps.) de *Ethernet* a 100 Mbps. con cambios mínimos a la estructura del cableado existente. Hay

tres tipos de *Fast Ethernet*: *100BASE-TX* para el uso con cable *UTP* de categoría 5, *100BASE-FX* para el uso con cable de fibra óptica, y *100BASE-T4* que utiliza un par de cables más para permitir el uso con cables *UTP* de categoría 3. La norma *100BASE-TX* se ha convertido en la más popular debido a su íntima compatibilidad con la norma *Ethernet 10BASE-T*. En cada punto de la red se debe determinar el número de usuarios que realmente necesitan las prestaciones más altas, para decidir que segmentos del troncal necesitan ser específicamente reconfigurados para *100BASE-T* y seleccionar el hardware necesario para conectar dichos segmentos "rápidos" con los segmentos *10BASE-T* existentes.

2.6.2 El modelo OSI.

OSI (*Open Systems Interconnection / Interconexión de Sistemas Abiertos*) es una norma común que permite la comunicación entre computadores que adopten este modelo de red.

Este modelo propone dividir en diferentes niveles todas las tareas que se llevan a cabo en la comunicación entre equipos de cómputo.

Consta de siete capas:

- *Nivel Físico*

Se definen las especificaciones eléctricas y mecánicas necesarias para mantener la comunicación entre computadores (dimensiones de los conectores, tipo de señal eléctrica a transmitir, tipos de cable).

- *Nivel de Enlace*

Establece el flujo de información que estará disponible a los usuarios de la red. Controla la ocurrencia de errores y su respectiva corrección. En esta parte del modelo se incluye información acerca del formato de los bloques de datos, los códigos de corrección, la forma de detección y corrección de errores.

- *Nivel de Red*

Se encarga de decidir la ruta por la que los datos serán transmitidos a través de la red. Incluye la administración y gestión de datos, la emisión de mensajes y la regulación del tráfico de la red.

- *Nivel de Transporte*

Asegura la transferencia de la información a pesar de los fallos que pudieran ocurrir en cualquiera de los niveles anteriores. Incluye la detección de bloqueos, las caídas del sistema y la búsqueda de rutas alternativas.

- *Nivel de Sesión*

Organiza las funciones que permiten que dos usuarios se comuniquen a través de la red. Incluye las tareas de seguridad, contraseñas de usuarios y la administración del sistema.

- *Nivel de Presentación*

Traduce la información del formato de la máquina a un formato entendible por los usuarios. Incluye el control de las impresoras, la emulación del tipo de terminal y los sistemas de codificación.

- *Nivel de Aplicación*

Se encarga del intercambio de información entre los usuarios y el sistema operativo. En esta parte se incluye a los programas de aplicación.

2.6.3 Protocolos de comunicación.

Son normas que permiten a los sistemas comunicarse. Un protocolo define la forma en que los sistemas deben identificarse entre si en una red, la forma en que los datos deben transitar por la red y cómo ésta información debe procesarse una vez que alcanza su destino final. Los protocolos también definen procedimientos para gestionar transmisiones o tramas perdidas o dañadas. *IPX (para Novell NetWare)*, *TCP/IP* (para *UNIX*, *Windows NT*, *Windows 95/98* y otras plataformas), *DECnet* (para conectar una red de ordenadores *Digital*), *AppleTalk* (para los ordenadores *Macintosh*), y *NetBIOS/NetBEUI* (para redes *LAN Manager* y *Windows NT*) son algunos de los protocolos más populares en la actualidad.

Aunque cada protocolo de la red es diferente, todos pueden compartir el mismo cableado físico. Este concepto es conocido como *independencia de protocolos*, lo que significa que los dispositivos que son compatibles en las capas de los niveles físico y de datos permiten al usuario ejecutar muchos protocolos diferentes sobre el mismo medio físico.

La presente tesis esta enfocada principalmente a los protocolos que permiten la comunicación con los diferentes equipos de laboratorio clínico, debido a que cada fabricante establece un protocolo particular no es posible establecer un método de análisis estándar para los datos recibidos.

2.6.4 Interfaces.

Una Interfaz es un punto en el que se establece una conexión entre dos elementos, que les permite trabajar juntos. En el campo de la informática se distinguen diversos tipos de interfaces que actúan a diversos niveles, desde las claramente visibles, que permiten a las personas comunicarse con los programas, hasta las imprescindibles interfaces hardware, a menudo invisibles, que conectan entre sí los dispositivos y componentes dentro de los ordenadores o computadoras. Las interfaces de usuario cuentan con el diseño gráfico, los

comandos, mensajes y otros elementos que permiten a un usuario comunicarse con un programa. Las microcomputadoras disponen de tres tipos básicos de interfaces de usuario (que no necesariamente son excluyentes entre sí): la interfaz de línea de comandos, reconocible por los símbolos A o C del sistema *MS-DOS*, que responde a los comandos introducidos por el usuario; la interfaz controlada por menús utilizada en muchas aplicaciones (por ejemplo *Lotus 1-2-3*) ofrece al usuario una selección de comandos, permitiéndole elegir uno de ellos presionando la letra correspondiente, desplazando el cursor con las teclas de dirección o apuntando con el ratón (*mouse*); y la interfaz gráfica de usuario, una característica de los equipos *Apple Macintosh* y de los programas basados en ventanas, representa visualmente los conceptos, por ejemplo un escritorio, y permite al usuario no sólo controlar las opciones de los menús, sino también el tamaño, la posición y el contenido de una o más ventanas que aparezcan en pantalla.

En el interior de las computadoras, donde el software funciona a niveles menos visibles, existen otros tipos de interfaces, como las que hacen posible que los programas trabajen con el sistema operativo y las que permiten al sistema operativo trabajar con el hardware de la computadora.

En hardware se entienden por interfaces las tarjetas, los conectores y otros dispositivos con que se conectan los diversos componentes a la computadora para permitir el intercambio de información. Existen, por ejemplo, interfaces estandarizadas para la transferencia de datos, como el *RS-232-C* y el *SCSI*, que permiten interconectar computadoras e impresoras, discos duros y otros dispositivos.

2.7 Planificación de tiempos.

2.7.1 Selección de tareas.

Para desarrollar una planificación temporal del proyecto, se debe de distribuir un conjunto de tareas a lo largo de su duración. El conjunto de tareas se desplegara dependiendo del tipo de proyecto y del grado de rigor. Cada tipo de proyecto puede enfocarse usando un modelo de proceso lineal secuencial, iterativo, o evolutivo. En algunos casos, un tipo de proyecto fluye suavemente hacia el siguiente, por ejemplo, los proyectos de desarrollo de concepto que tienen éxito evolucionan a menudo en nuevos proyectos de desarrollo de aplicación. Esta progresión es natural y predecible y ocurrirá sea cual sea el modelo de proceso que adopte una organización.

2.7.2 Red de tareas.

Las tareas y subtareas individuales tienen interdependencias basadas en su secuencia. Además, cuando hay más de una persona implicada en un proyecto, es probable que las actividades de desarrollo y tareas se realicen en paralelo. Cuando esto ocurre las tareas concurrentes deben coordinarse de manera que estén finalizadas cuando tareas posteriores requieran sus resultados.

Una red de tareas es una representación gráfica del flujo de las mismas en un proyecto. Se emplea a veces como el mecanismo a través del cual se introduce la secuencia de actividades y las dependencias en una herramienta de programación automática de la planificación temporal del proyecto. En su forma más sencilla, la red de tareas muestra las actividades principales de ingeniería del software.

2.7.3 Planificación temporal.

La planificación temporal de un proyecto de software no difiere mucho del de cualquier esfuerzo de ingeniería multitarea. Por tanto, se pueden aplicar herramientas de planificación temporal de proyectos y técnicas generales al software con pocas modificaciones.

Las técnicas de evaluación y revisión del programa (*PERT*) y el método de camino crítico (*CPM*), son dos métodos de la planificación temporal de un proyecto que puede aplicarse a desarrollo del software. Ambas técnicas son dirigidas por la información ya desarrollada en actividades anteriores de la planificación del proyecto.

Las tareas, a veces denominadas estructura de descomposición del trabajo del proyecto, se definen para el producto como un todo o para las funciones individuales.

2.7.4 Gráfico de tiempo.

Cuando se crea una planificación temporal de un proyecto de software, el planificador empieza un conjunto de tareas. Si se emplean herramientas automáticas, la descomposición del trabajo es introducida como una red de tareas o esquemas de tareas. El esfuerzo, duración y fecha de inicio son las entradas de cada tarea. Además, se asignan las tareas a individuos específicos.

Como consecuencia de esta entrada, se genera un gráfico de tiempo, también denominado gráfico de Gantt. Se puede desarrollar un gráfico de tiempo para todo el proyecto, alternativamente; se pueden desarrollar diferentes gráficos para cada función del proyecto o para cada individuo que trabaje en el mismo.

La Figura 5 ilustra el formato de una gráfica de tiempo. Muestra una parte de la planificación temporal de un proyecto de ingeniería de software.

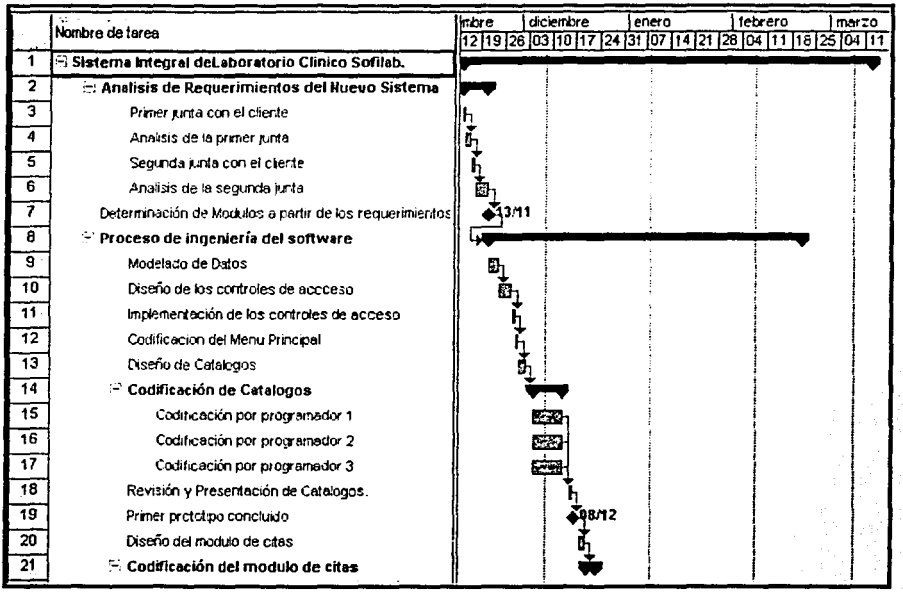


Figura 5.
Gráfica de tiempo.

2.7.5 *Seguimiento de la planificación temporal.*

La planificación temporal del proyecto le proporciona al gestor un "mapa de carreteras". Si se ha desarrollado apropiadamente, define las tareas e hitos[8] que deben seguirse y controlarse a medida que progresa el proyecto. El seguimiento se puede hacer de diferentes maneras:

- Realizando reuniones periódicas del estado del proyecto las que todos lo miembros del equipo de trabajo informa del progreso y de los problemas.

- Evaluando los resultados de todas las revisiones realizadas a lo largo del proceso de la ingeniería de software.

[8] ***Hito:*** Punto de referencia que marca acontecimientos importantes en un proyecto y que se utiliza para controlar el progreso del proyecto. Las tareas con duración cero se muestran como hitos.

- Determinando si se han conseguido los hitos formales del proyecto en las fechas programadas.

- Comparando la fecha real de inicio con las previstas para cada tarea del proyecto. Reuniéndose informalmente con los profesionales del software para obtener sus valoraciones subjetivas del progreso hasta la fecha y los problemas que se avecinen.

CAPÍTULO III.
INGENIERÍA.

3.1 Estimación de tiempos.

Como vimos en el capítulo anterior, la estimación o cálculo del tiempo de desarrollo de un sistema es parte fundamental de su planeación y además representa el primer paso para la construcción del sistema.

A continuación se muestra la forma de distribución de las diversas tareas a lo largo de todo el proyecto, dichas tareas se encuentran divididas en varios puntos principales con la finalidad de mostrar las diversas etapas de desarrollo, entre las que resaltan el análisis de requerimientos del sistema, el proceso de ingeniería de software y codificación así como también la fase de pruebas de campo.

Tarea	*Tiempo estimado*
Sistema Integral de Laboratorio Clínico Sofilab.	86 días
Análisis de requerimientos del nuevo sistema	6 días
Primer junta con el cliente	1 día
Análisis de la primer junta	2 días
Segunda junta con el cliente	1 día
Análisis de la segunda junta	2 días
Determinación de módulos a partir de los requerimientos.	0 días
Proceso de Ingeniería del Software	66 días
Modelado de datos	3 días
Diseño de los controles de acceso	2 días
Implementación de los controles de acceso	1 día
Codificación del menú principal	1 día
Diseño de catálogos	2 días
Codificación de Catálogos	7 días
Codificación por programador 1	7 días
Codificación por programador 2	7 días
Codificación por programador 3	7 días
Revisión y presentación de catálogos.	1 día
Primer prototipo concluido	0 días
Diseño del módulo de citas	2 días
Codificación del Módulo de Citas	3 días
Codificación por programador 1	3 días
Codificación por programador 2	3 días
Codificación por programador 3	3 días
Revisión de citas	1 día
Segundo prototipo concluido	0 días
Diseño de admisiones	1 día
Codificación de admisiones	2 días
Revisión de admisiones	1 día
Tercer prototipo concluido	0 días

Diseño del reporte de resultados	1 día
Módulo de resultados	1 día
Codificación del Módulo de Captura de Resultados Manuales	2 días
Codificación por programador 1	2 días
Codificación por programador 2	2 días
Revisión del área de resultados	1 día
Cuarto prototipo concluido	0 días
Diseño del programa de control y monitoreo para usuarios	2 días
Codificación del Programa de Control y Monitoreo	4 días
Codificación por programador 1	4 días
Codificación por programador 2	4 días
Codificación del módulo de mantenimiento.	1 día
Revisión del programa de control	1 día
Quinto prototipo concluido	0 días
Codificación de reportes	6 días
Codificación por programador 1	6 días
Codificación por programador 2	6 días
Codificación por programador 3	6 días
Diseño de la primera interfaz	2 días
Codificación de la Primera Interfaz	6 días
Codificación por programador 1	6 días
Codificación por programador 2	6 días
Prueba de la primera interfaz	2 días
Sistema concluido	0 días
Fase de Pruebas de Campo	11 días
Implementación del hardware y software requerido	4 días
Instalación del sistema y base de datos	1 día
Importación de los datos existentes en el viejo sistema	1 día
Pruebas de funcionamiento preliminar	1 día
Puesta en marcha	1 día
Liberación del sistema	0 días

Figura 6.
Planificación temporal.

Cabe resaltar que varias de las actividades descritas están contempladas para ser realizadas de manera simultánea, por lo que fue necesaria la participación de tres programadores para poder completar la tarea en el tiempo requerido.

El tiempo estimado para la realización del sistema fue de 89 días, aproximadamente 3 meses, es necesario indicar que dicho tiempo es en si, un cálculo aproximado, que se basa en proyectos similares desarrollos por *Sofilab*.

A continuación se presenta la red de tareas que muestra la relación de todas las tareas planificadas incluyendo las tareas simultáneas e hitos del proyecto.

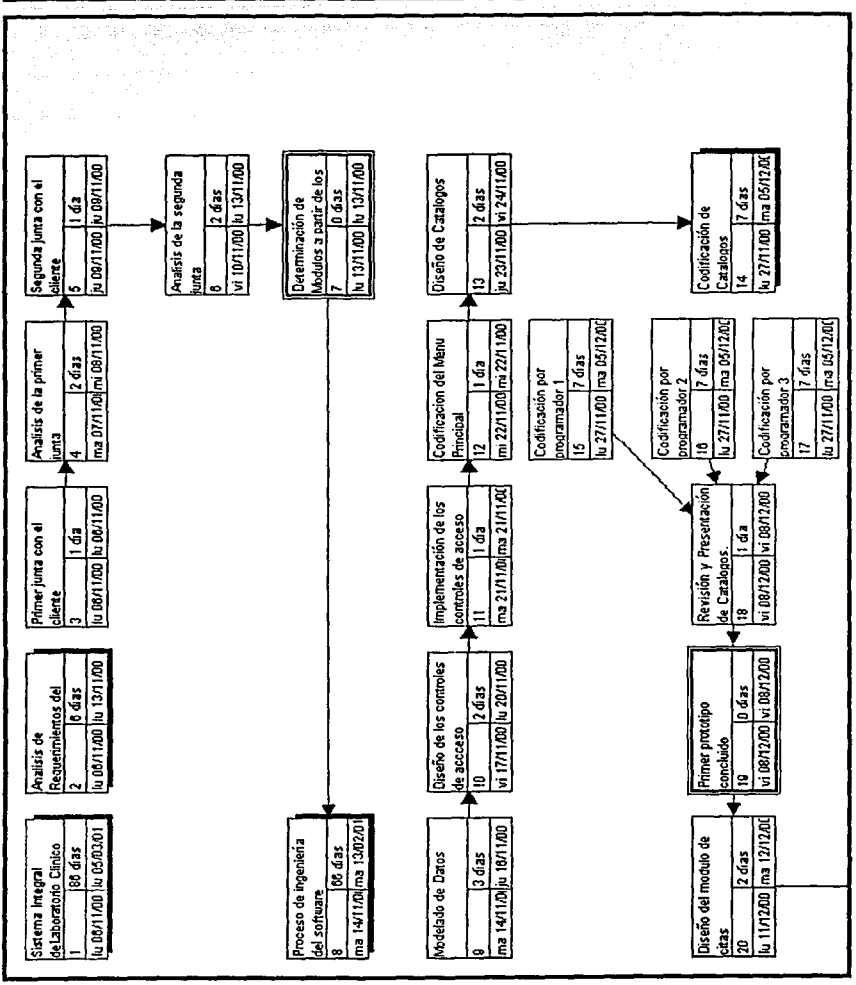


Figura 7.
Red de tareas 1.

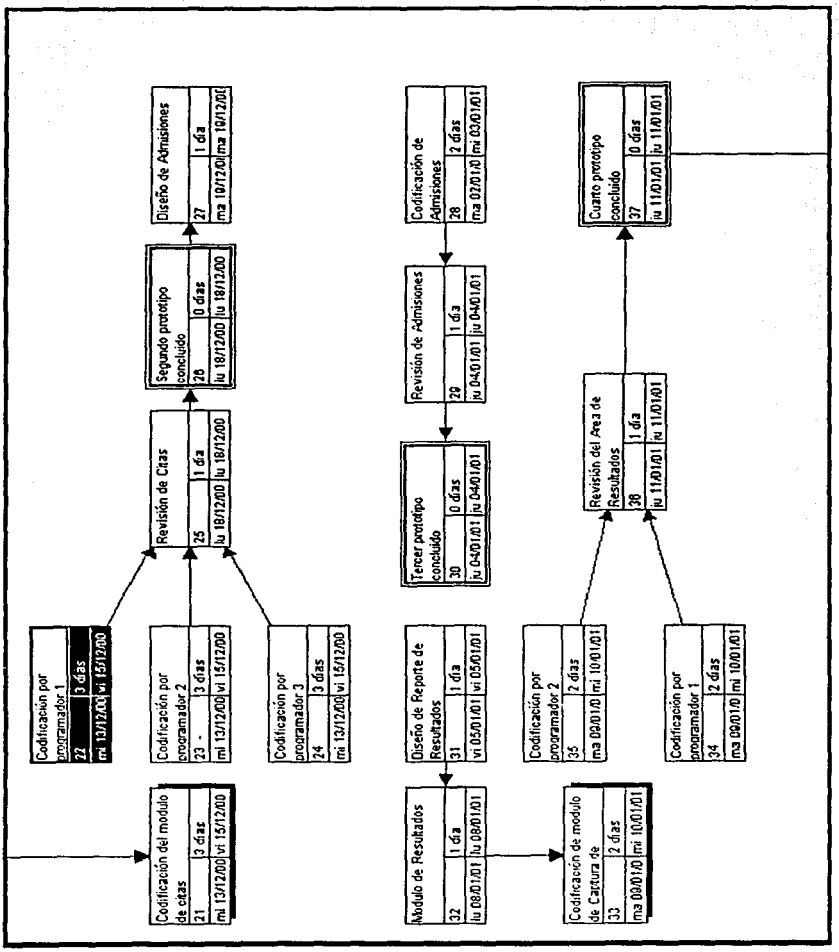


Figura 8.
Red de tareas 2.

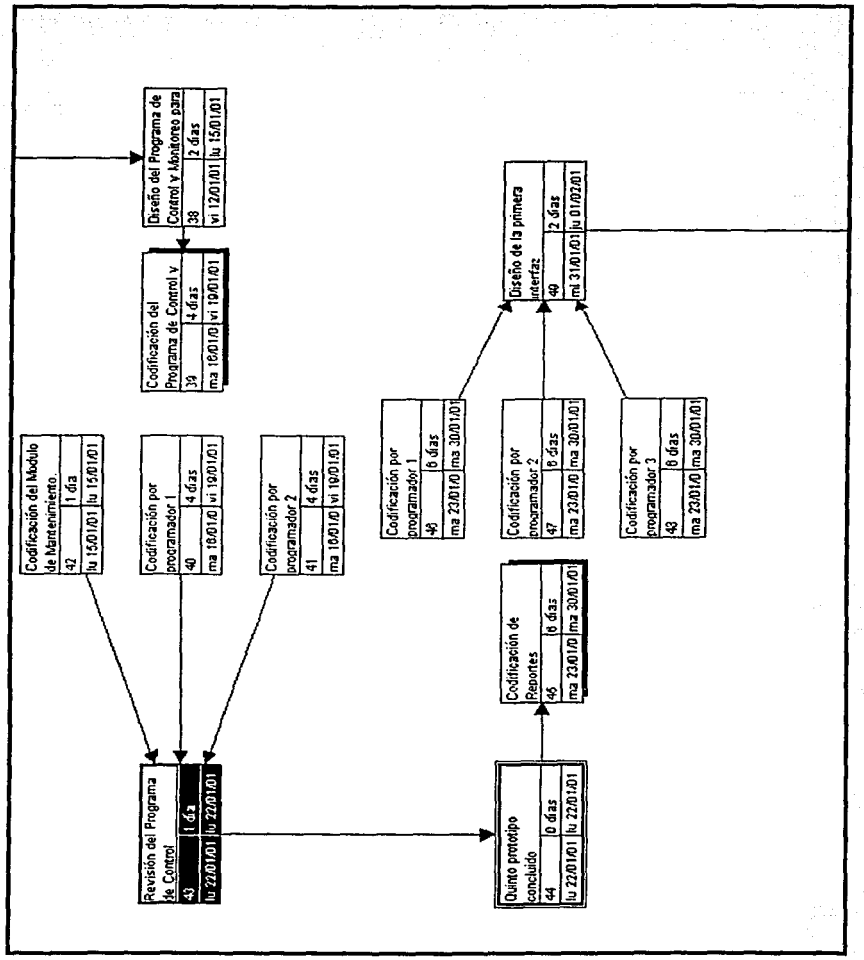


Figura 9.
Red de tareas 3.

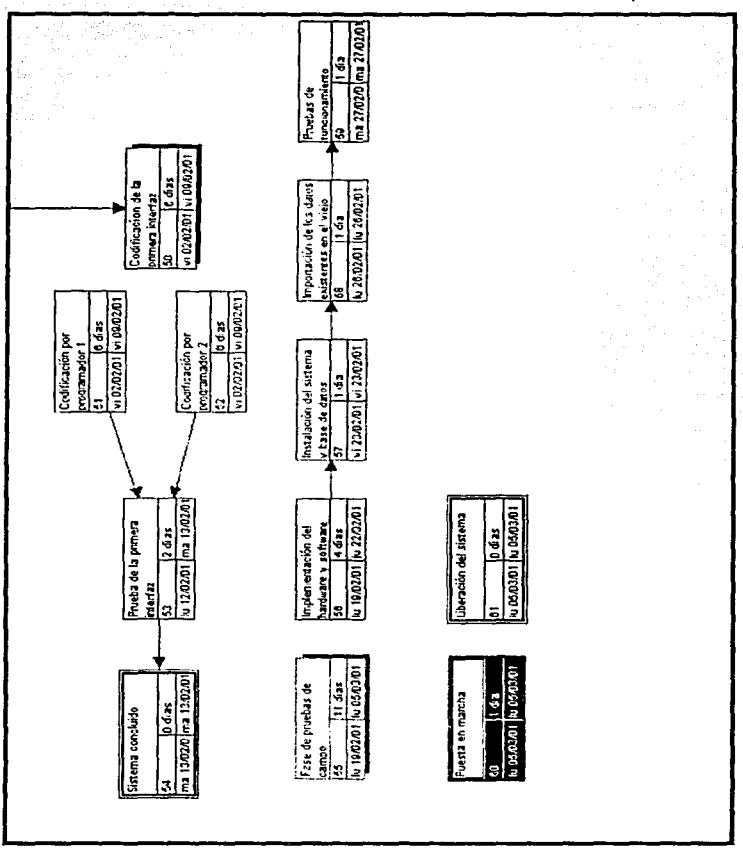


Figura 10.
Red de tareas 4.

Como forma complementaria generamos el gráfico de Gantt que permite la visualización en forma general de las actividades a lo largo del tiempo de desarrollo estimado, esté gráfico será consultado continuamente con el fin de lograr un seguimiento adecuado del sistema a realizar.

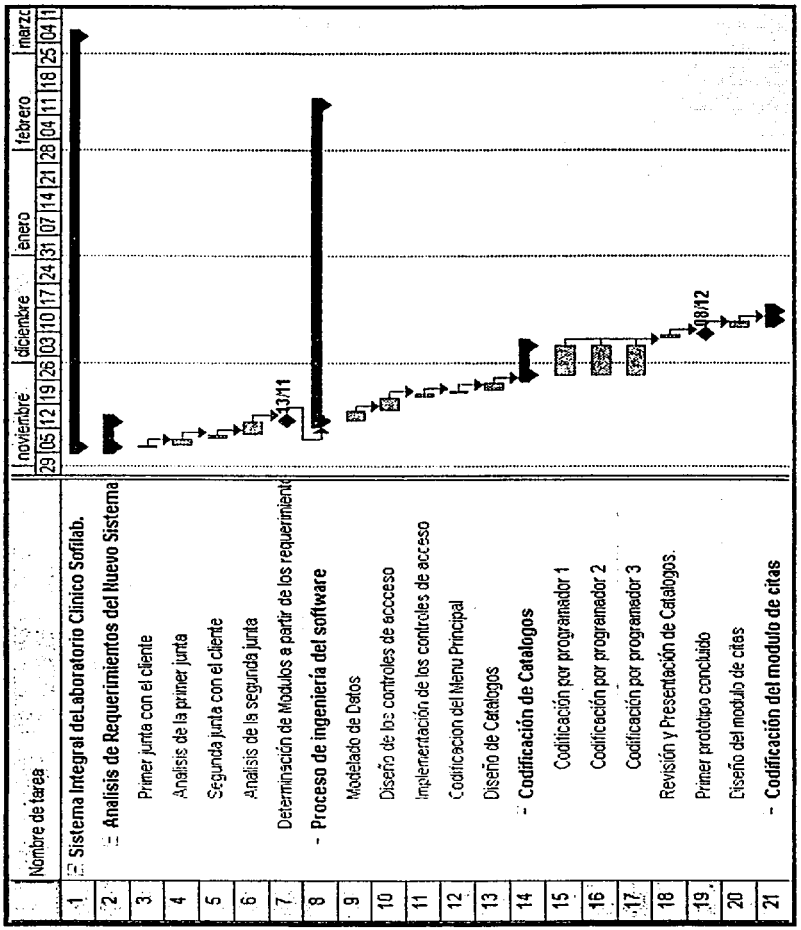


Figura 11.
Diagrama de Gantt 1.

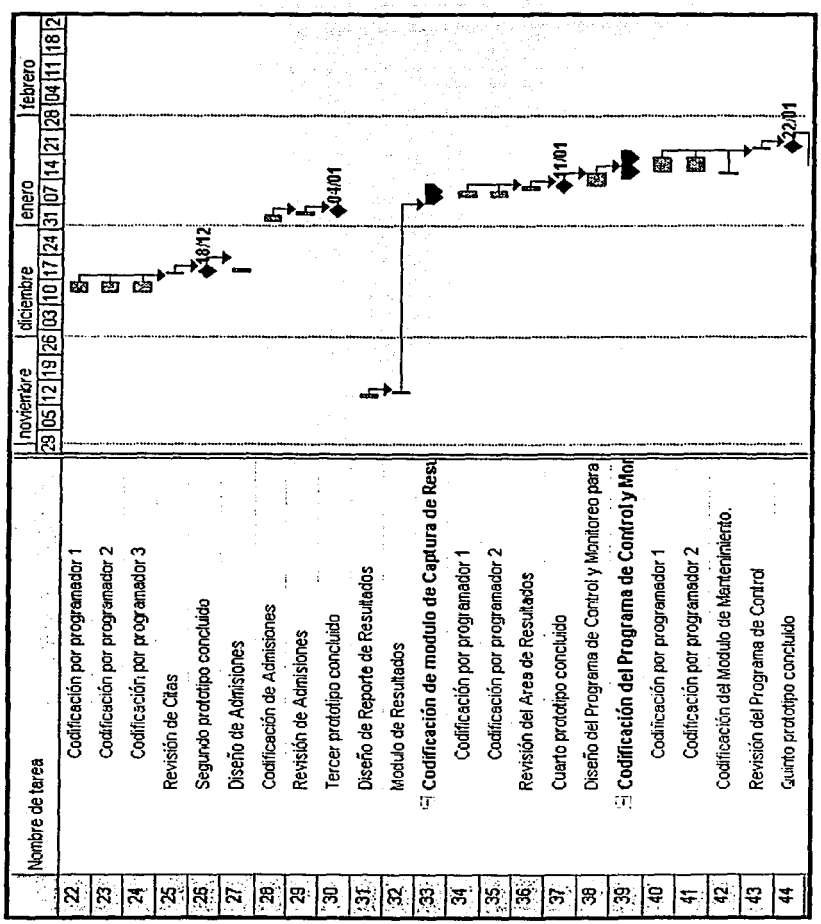


Figura 12.
Diagrama De Gantt 2.

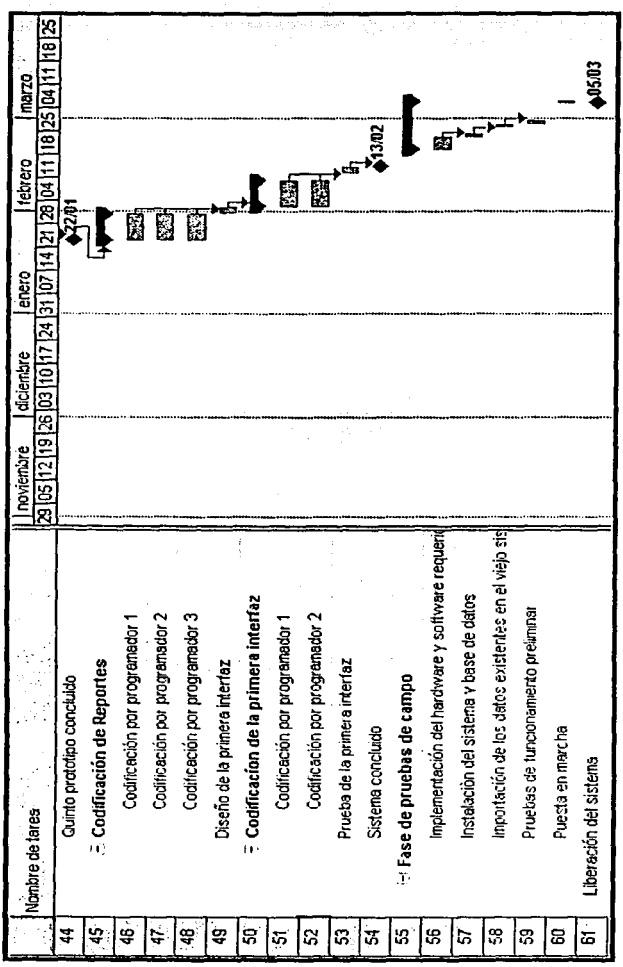


Figura 13.
Diagrama de Gantt 3.

3.2 Modelado de datos.

El modelo de datos responde a una serie de preguntas importantes para cualquier aplicación del procesamiento de datos, ¿cuales son los objetos de datos primarios que va a procesar el sistema?, ¿cuál es la composición de cada objeto de datos y qué atributos describe el objeto?, ¿cuál es la relación entre los objetos y los procesos que los transforman?

Es necesario plantear primero qué datos serán procesados por el sistema, para ello recordemos que un dato es la unidad mínima de información, que representa un atributo con características propias dentro del mismo sistema de información y está compuesto de una longitud, de un tipo y de un rango de valores que puede ser discreto o continuo. En especifico nuestros datos están relacionados con las actividades propias de un laboratorio clínico.

Los métodos de modelado de datos hacen uso de los diagramas de flujo de datos (*DFD*) y entidad relación (*DER*), dentro de los mismos observaremos la composición, atributos y las relaciones existentes entre los diversos objetos de datos que componen nuestro sistema.

3.2.1 Diagrama de flujo de datos (DFD).

El diagrama de flujo de datos es la representación gráfica de los flujos de información de un sistema de información, en otras palabras es el seguimiento del origen y destino de la información.

A continuación se define la simbología utilizada dentro del *DFD* para facilitar la comprensión del *DFD* del sistema.

- *Entidad externa:* Representa los orígenes y destinos de los datos del sistema de información.

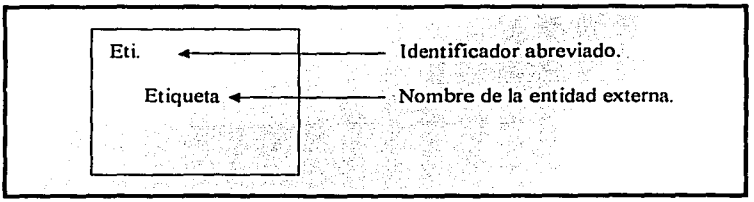


Figura 14.
Entidad externa.

- *Flujo de datos:* Representa el movimiento de los datos dentro del sistema

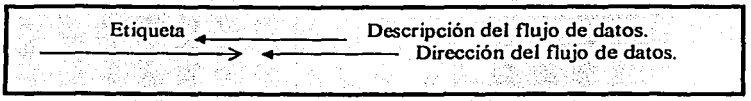


Figura 15.
Flujo de datos.

- *Procesos:* Representa los puntos dentro del sistema donde la información sufre modificaciones.

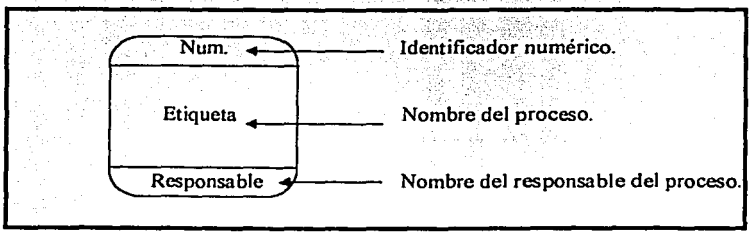


Figura 16.
Procesos.

- *Almacenamiento de datos:* Es el lugar físico dentro del sistema donde se guarda la información (en nuestro caso la base de datos).

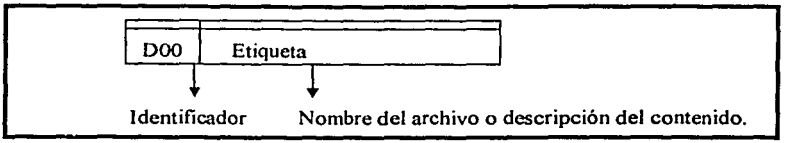


Figura 17.
Almacenamiento de datos.

Basados en los antecedentes mencionados en el capítulo I donde se trató el funcionamiento del laboratorio clínico, se separaron las diversas etapas en las cuales se procesa la información con el fin de lograr un esquema que describa de mejor manera el funcionamiento del laboratorio, permitiéndonos así tener una visión más clara de cada proceso y de cada flujo de datos. El siguiente diagrama relaciona de forma general lo anteriormente mencionado.

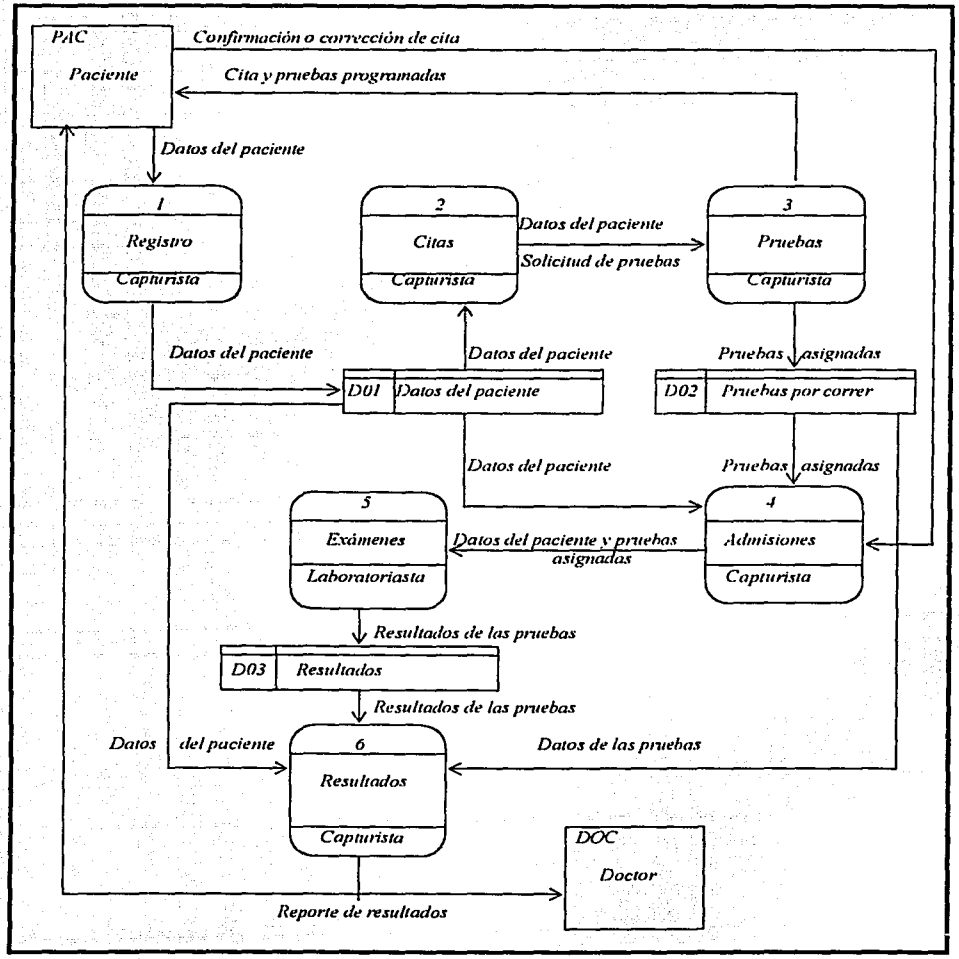


Figura 18.
Diagrama de flujo de datos general.

En el diagrama anterior, se puede observar que la información proporcionada por la entidad *Paciente (PAC)*, debe pasar por el proceso de *Registro (1)* donde al final del proceso los datos se almacenarán de forma física en el área esquematizada como *Datos del paciente (D01)*, siendo responsable de ello un capturista del laboratorio del área de admisiones. El siguiente paso que debe de seguir el flujo normal de información es a través del proceso de *Citas (2)*, este proceso solicita los datos del paciente del registro *D01*, dentro del cual se establece la fecha para la misma, información que es enviada al proceso de *pruebas (3)* el cual se encarga de asignar las diversas pruebas y/o perfiles registrándolas en la sección *pruebas por correr (D02)* y a su vez de proporcionarle a la entidad *PAC* los datos de la cita y las pruebas programadas. Una vez capturados los datos del paciente y asignada una cita, el proceso *Admisiones (4)* es el encargado de confirmar los datos y autorizar las pruebas mandando el conjunto de datos obtenidos al proceso de *Exámenes (5)*, el cual se encargará de generar los resultados que serán almacenados en la sección *Resultados (D03)* dejando disponibles dichos datos para el proceso *Resultado (6)* en donde se crearán los diversos reportes los que serán entregados a las entidades *PAC y Doctores (DOC)*.

Cada proceso que compone el *DFD* general esta compuesto por diversos subprocesos, los cuales serán detallados a continuación.

DFD del proceso de Registro (proceso 1).

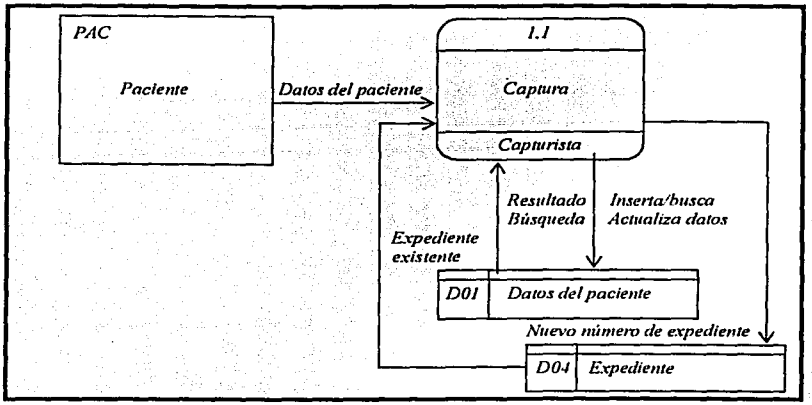


Figura 19.
DFD proceso de Registro.

El proceso de registro recibe los datos directamente de la entidad externa *PAC*, entre los que destacan nombre, edad, sexo, interno o externo, etc. Los datos son recibidos por el proceso *Captura (1.1)*, en donde se realiza una búsqueda para verificar si los datos de la

entidad *PAC* se encuentran almacenados, de ser así se envía en respuesta los datos y el número de expediente existente de la entidad almacenados en las secciones *D01 y Expedientes (D04)*, de lo contrario se realiza una inserción de datos y se genera un nuevo número de expediente.

DFD del proceso Citas (Proceso 2)

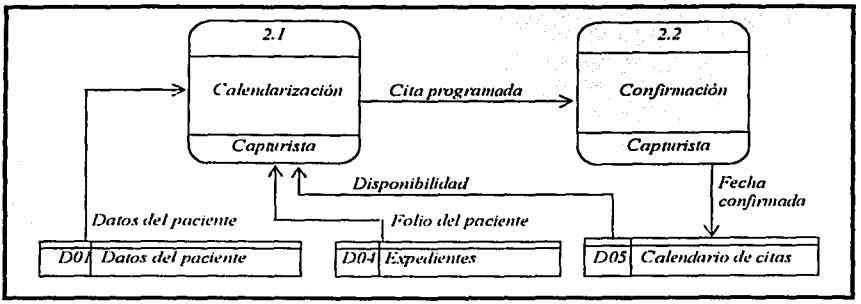


Figura 20.
DFD proceso de Citas.

Recordemos que el proceso de citas es el encargado de establecer la fecha de la cita, es necesario que se ejecute un proceso de *Calendarización (2.1)*, el cual obtiene los datos necesarios de los registros *D01 y D04* y a su vez obtiene la disponibilidad del laboratorio para una fecha especifica del registro *Calendario de citas (D05)* la cual es validada en el proceso de *Confirmación (2.2)* que básicamente consiste en no programar citas los días sábado y domingo y a su vez no programar más de 20 citas en un día.

DFD de Pruebas (Proceso 3).

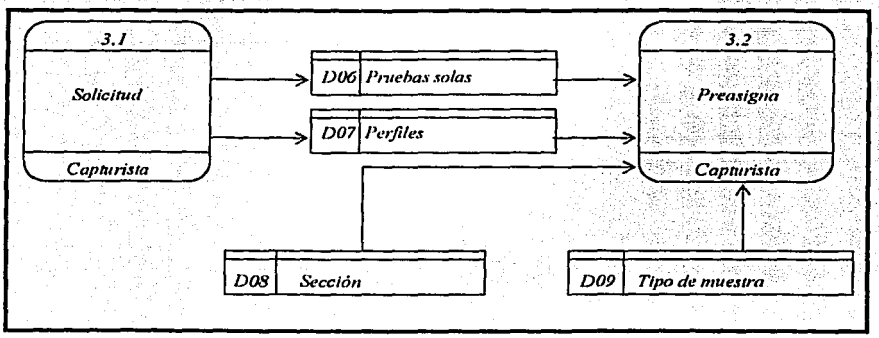


Figura 21.
DFD proceso de Pruebas.

Una vez establecida la fecha de la cita, es necesario asignar las pruebas y/o perfiles correspondientes. Mediante el proceso de *Solicitud (3.1)* se obtienen todas las posibles pruebas y/o perfiles que pueden ser asignados de los registros *Pruebas (D06) y Perfiles (D07)*, este conjunto de pruebas son procesadas en el proceso *Preasigna (3.2)* en donde se asignan de manera definitiva a una sección de laboratorio y se establece el tipo de muestra necesarias de los registros *Sección (D08) y Tipo de muestra (D09)* una vez asignadas las pruebas definitivas y las fechas de las citas se envía la información al proceso de *Admisiones* y a la entidad *PAC*.

DFD de Admisiones (Proceso 4).

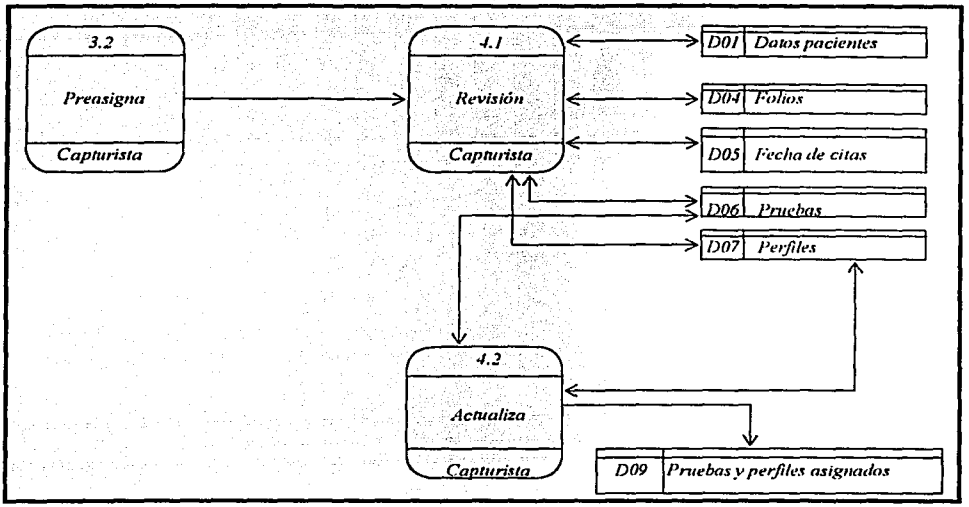


Figura 22.
DFD proceso de Admisiones.

El proceso de *Admisiones* se encarga de confirmar los datos de una cita realizando una consulta a los registros *D01, D04, D05, D06 y D07* y a su vez confirmando los datos de la entidad *PAC* que de ser necesario solicitará un cambio en la fecha de la cita o en las pruebas y/o perfiles asignados, enviando estos datos al registro *Pruebas asignadas (D09).*

DFD de Exámenes (Proceso 5).

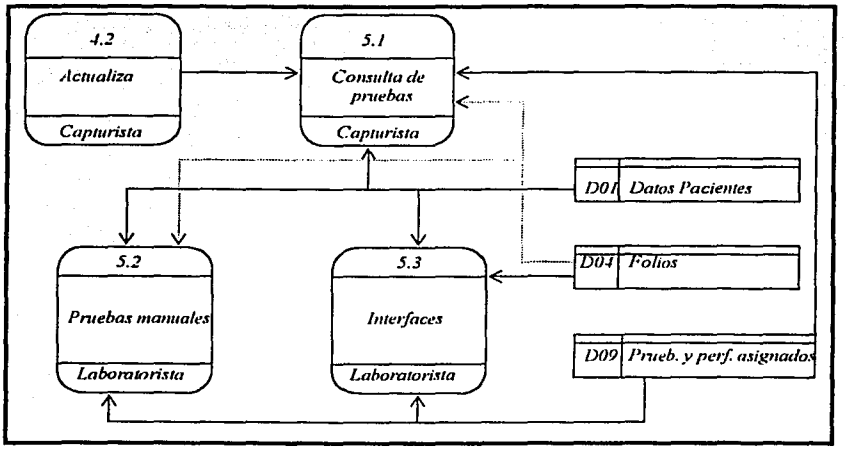


Figura 23.
DFD proceso de Exámenes.

El proceso de *Exámenes* generará los resultados de todas las pruebas solicitadas por la entidad *PAC* , para ello el proceso *Consulta de pruebas (5.1)* obtendrá los datos de los registros *D01, D04, y D09*, esta consulta será utilizada por los procesos *Resultados manuales (5.2)* e *Interfaces (5.3)*, los cuales se encargarán de obtener los resultados de cada prueba que les corresponda, en el proceso *5.1* se realizarán y actualizarán los resultados de todas aquellas pruebas que no corresponden a un aparato de análisis clínico interfazado o a una prueba de carácter manual, correspondiendo las antes mencionadas al proceso *5.2* en donde se reciben los resultados directamente de los aparatos del laboratorio mediante las interfaces conectadas a ellos. Los resultados obtenidos serán almacenados en el registro correspondiente *Resultados (D03)*.

DFD Resultados (Proceso 6).

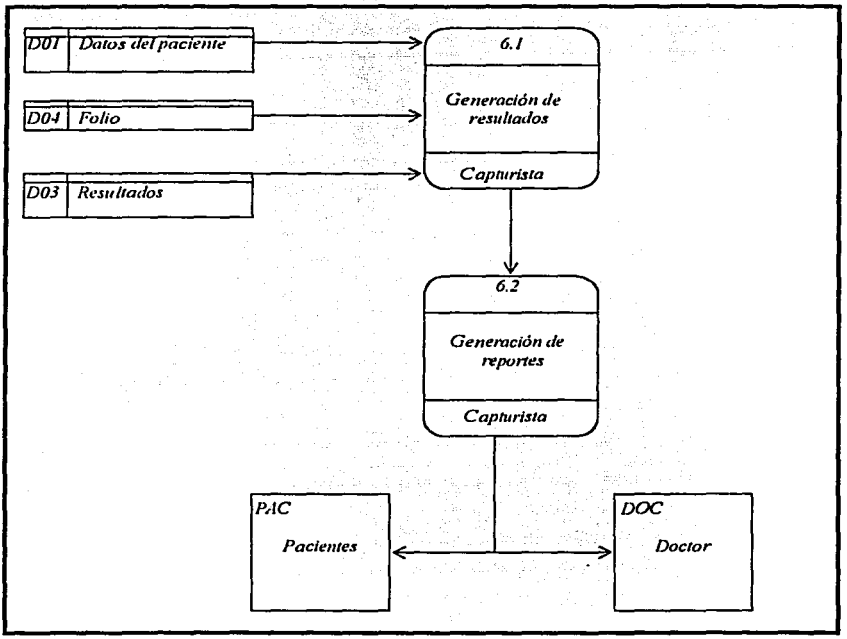


Figura 2-4.
DFD proceso de Resultados.

Este es el último proceso que se ejecuta dentro del sistema de laboratorio, dentro de este proceso se encuentra el subproceso *Reportes (6.1)* el cual se encargará de generar los reportes necesarios que contengan los datos de los registros *D01, D03 y D04*, así mismo estos reportes podrán tener un formato general y de fines estadísticos recordando que los reportes principales serán los entregados a las entidades *PAC y/o DOC.*

3.2.2 Diagrama entidad – relación. (DER).

Se basa en la percepción del mundo real, que consiste en un conjunto de objetos llamados entidades y las relaciones entre estas, representa la estructura lógica general de la base de datos. Ahora se describen los objetos que se utilizan en un diagrama entidad-relación *(DER)* así como su tipos.

Las entidades son objetos concretos como un libro o abstractos como un dia festivo. Por ejemplo: Juan López.

Una entidad está representada por un conjunto de características propias o (atributos). Para cada atributo existe un rango de valores permitidos llamados dominio del atributo, por ejemplo: El dominio del atributo "*número de cuenta*" podria ser el conjunto de todos los enteros positivos. Así un conjunto de entidades es un grupo de entidades del mismo tipo.

Ejemplo: Las personas que tienen cuenta en un banco.
Sucursal (Nombre_Sucursal, Ciudad_Sucursal, Activo)
Cliente (Nombre_Cliente, Seguro_Social, Calle)

Una relación es la asociación entre entidades. Por ejemplo Asociación del Cliente Juan López con número de cuenta 610. Un conjunto de relaciones es un grupo de relaciones del mismo tipo. Por ejemplo: Las relaciones Cliente-Cuenta para denotar la Relación Cliente y Cuenta.

Tipos de Relaciones.

Las Relaciones entre entidades pueden ser de los siguientes tipos:

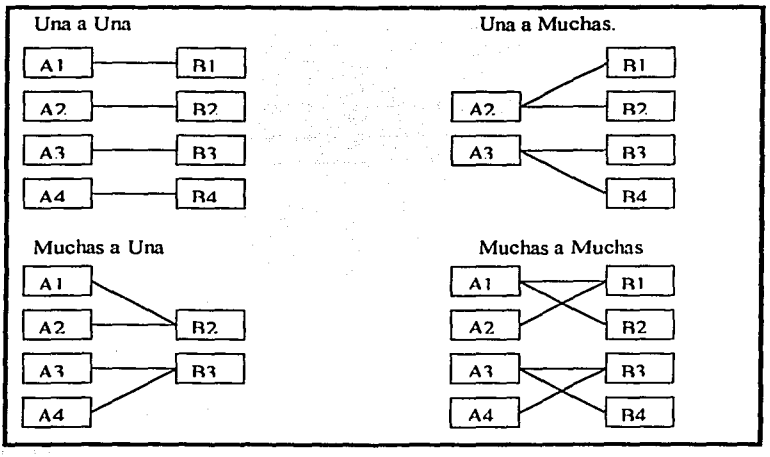


Figura 25.
Tipos de relaciones.

Estas definiciones junto con lo ya mencionado sobre bases de datos fueron el principal elemento para la creación del *DER* el cual se muestra a continuación de manera general.

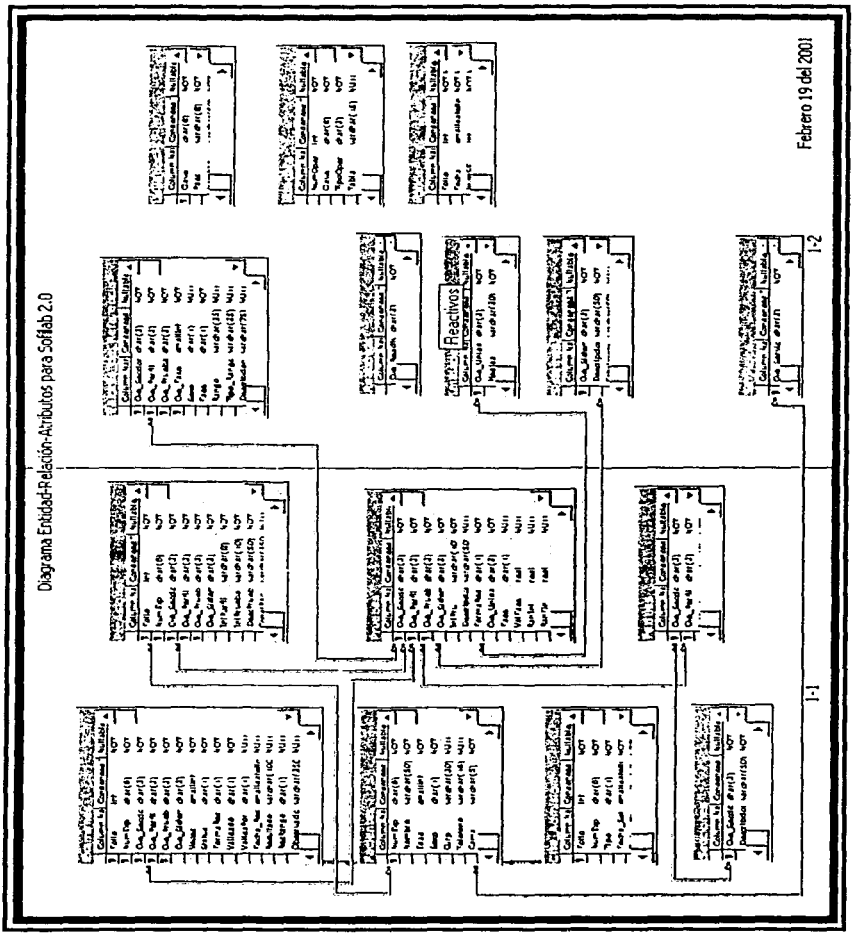


Figura 26.
Diagrama entidad relación vista general.

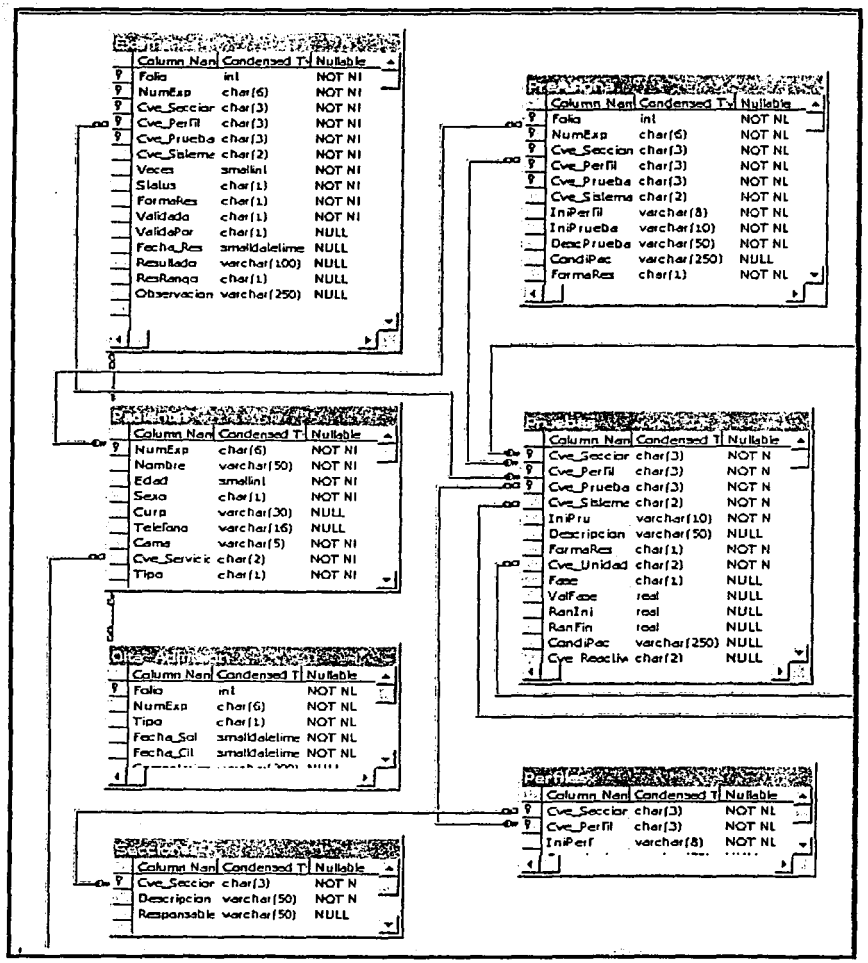


Figura 27.
Diagrama DER vista de la parte Izquierda

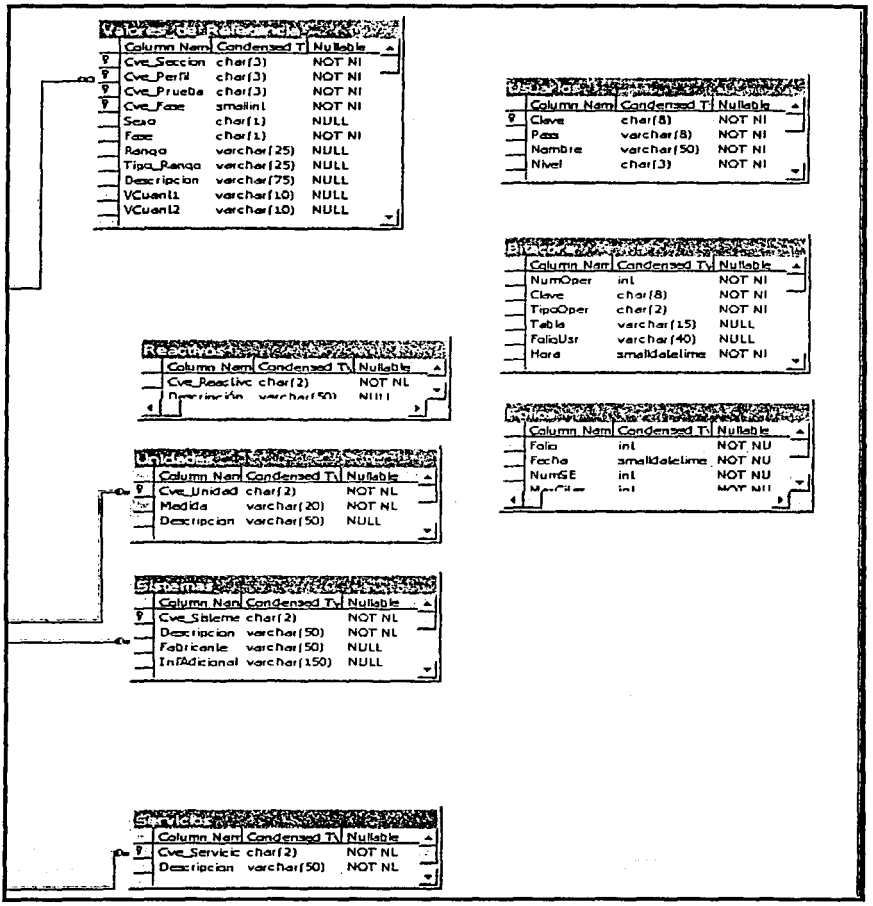


Figura 28.
Diagrama DER vista de la parte derecha

En el diagrama se puede apreciar la existencia de tablas que parecen no estar unidas al sistema sin embargo son importantes para el sistema en si. La Primera de ellas es la tabla de *Usuarios* la cual contiene la información especifica de cada usuario del sistema, ésta tabla se utilizará para restringir el acceso a personal no autorizado o delimitar los accesos para el personal del laboratorio de acuerdo con sus necesidades. La tabla de *Bitácora* esta relacionada con la tabla de *Usuarios* en una relación de uno a muchos, con el fin de que en la *Bitácora* sólo se contenga información de usuarios reales, es decir, la integridad del sistema. *Bitácora* como su nombre lo indica, contiene un registro de la actividad general del sistema. La tabla de *Folio* sólo es utilizada para llevar una relación de los folios asignados y los registros sin expediente, es por esto que no se relaciona con ninguna otra tabla.

Ahora procedamos con las tablas que corresponden a los catálogos del sistema, la primera es *Sistemas* la cual tiene una relación de uno a muchos con la tabla *Pruebas* a través de su llave *Cve_sistema*, La relación de todos los equipos existentes en el laboratorio clínico esta en la tabla *Sistemas*. La tabla de *Unidades* contiene el tipo de unidades de medida para las pruebas realizadas por lo que está relacionada de una a muchas con la tabla de *Pruebas*. La tabla que se utiliza para el control de los reactivos y similares es la de *Reactivos* que también tiene relación de uno a muchos con la tabla de *Pruebas*. *Valores_de_Referencia* es usada para poder establecer lo que en *Pruebas* es el rango el cual indicará si es un valor normal o anormal en el reporte de *Resultados*, la idea es que se pueda automatizar dicha parte para calcular automáticamente el rango correspondiente a cada prueba, su llave principal es compuesta por *Cve_Sección*, *Cve_Perfil*, *Cve_Prueba* y *Cve_Fase*; esto para poder contener la llave foránea con Pruebas de tal forma que su relación es de muchos a muchos, es decir, que un rango de referencia puede pertenecer a varias pruebas y a su vez varias pruebas pueden definir un rango o valores de referencia.. El *Catálogo de Servicios* es donde se definen los tipos de servicios que brinda el laboratorio, su tabla se relaciona de uno a muchos con la tabla *Pacientes* lo que significa que cada paciente solo puede recibir un servicio. *Secciones* identifica cada sección del laboratorio clínico, su relación directa es con *Perfiles* a través de su clave que es una abreviatura de su nombre, es conveniente detallar que a través de *Perfiles* se relaciona con *Pruebas* para mantener la integridad del sistema esto debido a que no existe una prueba que no pertenezca a un perfil, aún que sea el 000 con descripción sin perfil, y a su vez ningún perfil esta libre de *Sección*. Así, *Perfiles* es el catálogo más complejo debido a que define sus características de nombre y sección de pertenencia y principalmente una o varias pruebas que le corresponden por lo que su relación con Pruebas es uno a muchos a través de la llave compuesta por *Cve_Sección* y *Cve_Perfil*.

Pruebas es la tabla que contiene todas la pruebas del laboratorio clínico, tiene la llave principal compuesta por las claves de *Sección*, *Perfil* y *Cve_Prueba*, esto debido a que dichos campos en conjunto son únicos para cada prueba, esto es que pueden existir pruebas con claves iguales y con el mismo perfil pero diferente sección, secciones y perfiles iguales pero con prueba diferente, pruebas y secciones iguales pero pertenecientes a diferente perfil, que además de estar relacionada con las tablas antes mencionadas *Pruebas* tiene una relación de muchos a muchos con la tabla de *Exámenes*, su relación con la tabla *Preasigna* es de muchos a muchos y existe con la finalidad de que no puedan asignarse pruebas ficticias a los folios de *Citas*.

Pacientes es la tabla que contiene la información demográfica de cada paciente así como su número de expediente el cual es su llave principal, al igual que *Pruebas* esta relacionada con la tabla de *Preasigna* sólo que su relación es de uno a muchos, a un solo paciente le corresponden varias pruebas asignadas para cita.

Preasigna es la tabla que contiene las pruebas programadas para todos los pacientes antes de ser admitidas, esto con el fin de agilizar y controlar con mayor facilidad el manejo de estas pruebas (altas, bajas), sus únicas relaciones son con *Pacientes* y *Exámenes* como se explicó anteriormente para asegurar la integridad de los datos.

Exámenes es la tabla donde están contenidos todos los resultados de las pruebas aplicadas por el laboratorio clínico, es aquí donde se concentra la información más importante del sistema ya que se utiliza para generar los reportes para cada paciente, sus relaciones son de muchos a muchos con las tablas de *Pruebas* y *Pacientes*.

3.2.3 Diccionario de datos.

El diccionario de datos es un listado organizado de todos los elementos de datos que son pertinentes para el sistema con definiciones precisas y rigurosas que permiten que el usuario y el analista del sistema tengan una misma comprensión de las entradas, salidas, y de los cálculos intermedios.

Aunque el formato del diccionario de datos varía entre las distintas herramientas, la mayoría contiene la siguiente información:

- *Nombre:* el nombre principal del elemento de datos o de control, del almacén de datos o de una entidad externa.
- *Alias:* otros nombres usados para el nombre.
- *Dónde se usa / cómo se usa:* un listado de los procesos que usan el elemento de datos o de control y cómo lo usan.
- *Descripción del contenido:* el contenido representado mediante una notación.
- *Información adicional:* otra información sobre los tipos de datos, los valores implícitos, las limitaciones y las restricciones, etc.

Tabla de Valores de referencia: Esta tabla hace referencia a las claves de los diversos perfiles o pruebas a realizarse.

CAMPO	LLAVE	NOMBRE	ABREVIATURA	TIPO	LONGITUD	NULO
1	Si	Clave de sección	Cve_Sección	Char	3	N
2	Si	Clave del perfil	Cve_Perfil	Char	3	N
3	Si	Clave de la prueba	Cve_Prueba	Char	3	N
4	Si	Clave de la fase	Cve_Fase	smallint	variable	N
5		Sexo del paciente	Sexo	Char	1	S
6		Fase de la prueba	Fase	Char	1	N
7		Rango de valores de la prueba	Rango	Varchar	25	S
8		Tipo de rango de la prueba	Tipo_Rango	Varchar	25	S
9		Descripción del tipo de rango	Descripción	Varchar	75	S
10		Valor cuantitativo 1	Vcuant1	Varchar	10	S
11		Valor cuantitativo 2	Vcuant2	Varchar	10	S

Tabla de Usuarios: Guarda las claves de acceso, contraseñas (*passwords*) y niveles de acceso de cada usuario.

CAMPO	LLAVE	NOMBRE	ABREVIATURA	TIPO	LONGITUD	NULO
1	Si	Clave de usuario	Clave	Char	8	N
2		Contraseña del usuario	Pass	Varchar	8	N
3		Nombre del usuario	Nombre	Varchar	50	N
4		Nivel de acceso	Nivel	Char	3	N

Reactivos: Guarda los datos relacionados a los tipos de reactivos que se utilizan para cada prueba, el número necesario en cada prueba y su cantidad en existencia.

CAMPO	LLAVE	NOMBRE	ABREVIATURA	TIPO	LONGITUD	NULO
1	Si	Clave de reactivo	Cve_reactivo	Char	2	N
2		Descripción del reactivo	Descripción	Varchar	50	S
3		Unidad del sistema MKS del reactivo	ReaUnidad	Char	10	S
4		Cantidad requerida por prueba	ReaCanPru	Char	4	S
5		Cantidad en existencia	ReaCanExi	Char	4	S
6		Cantidad máxima requerida	ReaCanMax	Char	4	S
7		Cantidad mínima requerida	ReaCanMin	Char	4	S

Bitácora: Contiene la información acerca de las operaciones que se realizan en el sistema.

CAMPO	LLAVE	NOMBRE	ABREVIATURA	TIPO	LONGITUD	NULO
1		Número de operación	NumOper	Int	Variable	N
2		Clave del usuario	Clave	Char	8	N
3		Tipo de operación	TipoOper	Char	2	N
4		Nombre de la tabla en la que se realiza la operación	Tabla	Varchar	15	S
5		Folio del usuario	FolioUsr	Varchar	40	S
6		Hora de la operación	Hora	Smalldatatime	Variable	N

Unidades: Contiene la información de los diferentes tipos de unidades empleadas en el laboratorio clínico.

CAMPO	LLAVE	NOMBRE	ABREVIATURA	TIPO	LONGITUD	NULO
1	Si	Clave de unidad	Cve_Unidad	Char	2	N
2		Medida en sistema MKS u otro	Medida	Varchar	20	N
3		Descripción de la unidad	Descripción	Varchar	50	S

Folios: Guarda la información referente al número de folio de cada paciente.

CAMPO	LLAVE	NOMBRE	ABREVIATURA	TIPO	LONGITUD	NULO
1		Número de folio	Folio	Int	Variable	N
2		Fecha de creación	Fecha	Smalldatatime	Variable	N
3		Número de seguro	NumSE	Int	Variable	N
4		Número de citas máximo	MaxCitas	Int	Variable	N

Sistemas: Contiene la información de los diversos aparatos utilizados para realizar los análisis clínicos.

CAMPO	LLAVE	NOMBRE	ABREVIATURA	TIPO	LONGITUD	NULO
1	Si	Clave del sistema	Cve_Sistema	Char	2	N
2		Descripción del sistema	Descripción	Varchar	50	N
3		Fabricante del aparato	Fabricante	Varchar	50	S
4		Información respecto a su unción	InfAdicional	Varchar	50	S

Servicios: Contiene la información referente a los diversos servicios que presta el laboratorio.

CAMPO	LLAVE	NOMBRE	ABREVIATURA	TIPO	LONGITUD	NULO
1	Si	Clave del servicio	Cve_servicio	Char	2	N
2		Descripción del tipo de servicio	Descripción	Varchar	50	S

Secciones: Contiene la información acerca de las diversas secciones del hospital como bacteriología, etc.

CAMPO	LLAVE	NOMBRE	ABREVIATURA	TIPO	LONGITUD	NULO
1	Si	Clave de Sección	Cve_Sección	Char	3	N
2		Descripción	Descripción	Varchar	50	N
3		Responsable	Responsable	Varchar	50	S

Perfiles: Contiene la información de los perfiles existentes en el laboratorio clínico para su aplicación a los pacientes.

CAMPO	LLAVE	NOMBRE	ABREVIATURA	TIPO	LONGITUD	NULO
1	Si	Clave de Sección	Cve_Sección	char	3	N
2	Si	Clave de Perfil	Cve_Perfil	char	3	N
3		Iniciales del Perfil	IniPerf	varchar	8	N
4		Descripción	Descripción	varchar	50	S

Tabla de Exámenes: Aquí se encuentran todos los resultados de las pruebas para la generación de reportes.

CAMPO	LLAVE	NOMBRE	ABREVIATURA	TIPO	LONGITUD	NULO
1	Si	Folio	Folio	int	Variable	N
2	Si	Número de expediente	NumExp	char	6	N
3	Si	Clave de sección	Cve_Sección	char	3	N
4	Si	Clave de Perfil	Cve_Perfi	char	3	N
5	Si	Clave de Prueba	Cve_Prueba	char	3	N
6		Clave de sistema	Cve_sistema	char	3	N
7		Veces	Veces	smallint	Variable	N
8		Status	Status	char	1	N
9		Forma de resultado	FormaRes	char	1	N
10		Válidado	Validado	char	1	N
11		Válidado Por	ValidadoPor	char	1	S
12		Fecha de resultado	Fecha_Res	smalldatetime		S
13		Resultado	Resultado	varchar	100	S
14		Rango de Resultado	ResRango	char	1	S

15		Observaciones	Observaciones	varchar	250	S

Tabla de Pacientes: Esta tabla guarda los datos demográficos de cada paciente que registra el sistema, así como su expediente y tipo de servicio que recibe.

CAMPO	LLAVE	NOMBRE	ABREVIATURA	TIPO	LONGITUD	NULO
1	Si	Número de Expediente	NumExp	char	6	N
2		Nombre	Nombre	varchar	50	N
3		Edad	Edad	smallint	variable	N
4		Sexo	Sexo	char	1	N
5		Curp	Curp	varchar	30	S
6		Teléfono	Teléfono	varchar	16	S
7		Cama	Cama	varchar	5	N
8		Clave de Servicio	Cve_Servicio	char	2	N
9		Tipo	Tipo	char	1	N

Tabla de Pruebas: Aqui se conservan las especificaciones de cada prueba que el sistema registra.

CAMPO	LLAVE	NOMBRE	ABREVIATURA	TIPO	LONGITUD	NULO
1	Si	Clave de sección	Cve_Sección	char	3	N
2	Si	Clave de perfil	Cve_Perfi	char	3	N
3	Si	Clave de prueba	Cve_Prueba	char	3	N
4		Clave de sistema	Cve_sistema	char	3	N
5		Iniciales de prueba	IniPru	varchar	10	N
6		Descripción	Descripción	varchar	50	S
7		Forma del resultado	Formares	char	1	N
8		Clave de unidad	Cve_Unidad	char	2	N
9		Fase	Fase	char	1	S
10		Valor de fase	ValFase	real	variable	S
11		Rango inicial	RanIni	real	variable	S
12		Rango final	RanFin	real	variable	S
13		Condiciones del paciente	CondiPac	varchar	250	S
14		Clave de reactivo	Cve_Reactivo	char	2	S

Tabla de PreAsigna: Esta tabla se utiliza para facilitar y agilizar las modificaciones en las citas que se programan.

CAMPO	LLAVE	NOMBRE	ABREVIATURA	TIPO	LONGITUD	NULO
1	Si	Folio	Folio	int	Variable	N
2	Si	Número de expediente	NumExp	char	6	N
3	Si	Clave de sección	Cve_Sección	char	3	N
4	Si	Clave de Perfil	Cve_Perfi	char	3	N
5	Si	Clave de Prueba	Cve_Prueba	char	3	N
6		Clave de sistema	Cve_sistema	char	3	N
7		Iniciales del Perfil	IniPerfl	varchar	8	N
8		Iniciales de Prueba	IniPru	varchar	10	N
9		Descripción de Prueba	DescPrueba	varchar	50	N
10		condiciones del paciente	CondiPac	varchar	250	S
11		Forma del Resultado	FormaRes	char	1	S

Tabla de Cita_Admision: Registra todas las citas de cada paciente; así como la fecha en la que es admitido.

CAMPO	LLAVE	NOMBRE	ABREVIATURA	TIPO	LONGITUD	NULO
1	Si	Folio	Folio	int	Variable	N
2		Número de expediente	NumExp	char	6	N
3		Tipo	Tipo	char	1	N
4		Fecha de solicitud	Fecha_sol	smalldatetime		N
5		Fecha de cita	Fecha_Cit	smalldatetime		N
6		Comentarios	Comentarios	varchar	200	S

3.3 Diseño de módulos.

Toda arquitectura de software conlleva modularidad; es decir, el software se divide en componentes identificables y tratables por separado, denominados módulos, que están integrados para satisfacer los requisitos del programa, cada módulo realiza una función específica dentro del sistema y al tener el sistema dividido se facilita su programación o más bien dicho, disminuye la complejidad del mismo.

Dentro de la presente sección, mostraremos de manera general la forma en que será estructurado el sistema dentro de sus diversas pantallas y funciones las cuales conforman los módulos de nuestro sistema.

Módulo de Pacientes.

El módulo de pacientes será una pantalla en donde se mostrarán los datos de los pacientes registrados en el sistema, los datos serán el número de expediente, nombre, edad, CURP, teléfono, número de cama (si es que está internado), sexo, el tipo de paciente (si es de rutina o urgencias) y el tipo de servicio, esta pantalla a su vez permitirá realizar búsquedas de pacientes tomando como criterios de búsqueda los datos contenidos en la tabla de la base de datos, lo que facilitará la búsqueda de los pacientes registrados en el hospital. Esta pantalla contará con un conjunto de botones que darán al usuario las opciones de ingresar un nuevo paciente, actualizar los datos de un paciente existente o darlo de baja, la forma tendrá la siguiente apariencia:

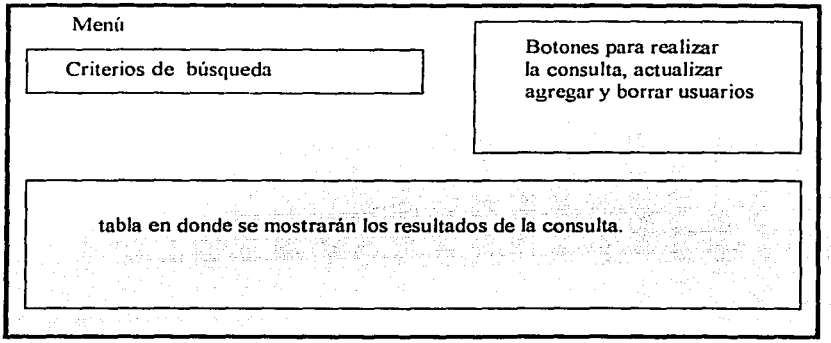


Figura 29.
Módulo de pacientes 1.

Las altas y actualizaciones de los pacientes se realizarán en una pantalla diferente que contendrá los campos a llenar o actualizar, un conjunto de botones que permitirá aceptar o cancelar el alta o actualización y para el caso del dato tipo de servicio un control proporcionado por el software *Sheridan Data Widgets* denominado *SSOleDBCombo* que permitirá mostrar el catálogo de servicios facilitando la elección del servicio a realizar sin tener la necesidad de memorizar las claves de los mismos, la pantalla tendrá más o menos la siguiente apariencia:

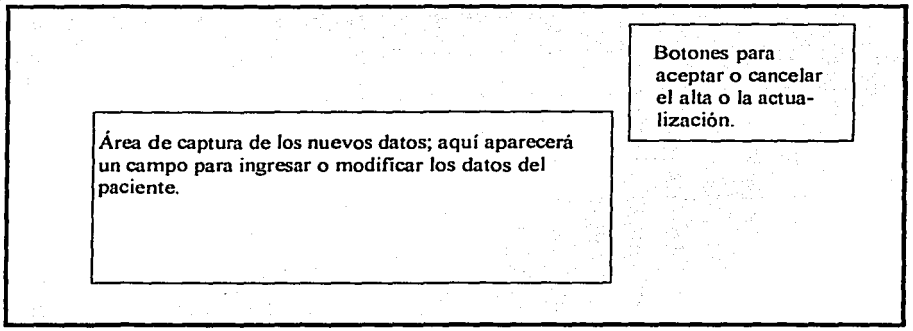


Figura 30.
Módulo de pacientes 2.

Módulo de Usuarios.

El módulo de usuarios será utilizado para monitorear y asignar los niveles y claves de acceso a los diversos usuarios que tendrá el sistema, debemos de recordar que dentro del manejo de un laboratorio clínico existen personas encargadas de la admisión de pacientes, de la ejecución de las pruebas, de la asignación de pruebas y administradores del mismo laboratorio, debido a las diversas funciones de los usuarios dentro del laboratorio es necesario tener un control de acceso al sistema, permitiendo a unas personas hacer una cosa y no permitirles hacer otras, para ello al momento de asignar una clave a un usuario nuevo es necesario asignarle un nivel de acceso, para ello se crea el módulo de usuarios que será desde el administrador del sistema, mismo que se encargue de asignar dichos niveles de acceso a los diversos usuarios del mismo.

La forma básica de la pantalla de usuarios es la siguiente

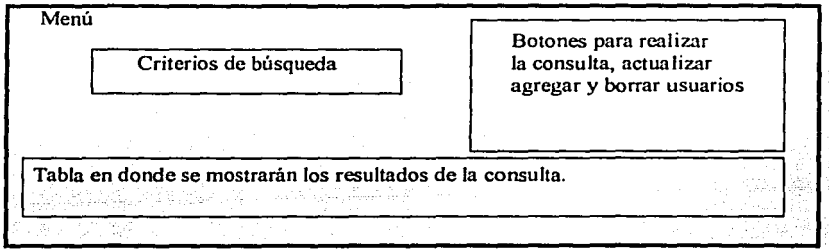


Figura 31.
Módulo de usuarios 1.

Un control *ComboBox* mostrará las categorías existentes dentro del sistema lo que facilitará al administrador la búsqueda de los usuarios, así mismo podrá buscar por nombre o realizar una consulta general de los usuarios existentes, la tabla mostrará el resultado de la búsqueda indicando la clave del usuario, su nombre, categoría, puesto y nivel de acceso, también existirán un conjunto de botones que permitirán agregar un nuevo usuario, actualizar el registro existente o eliminar al usuario del sistema, la actualización y alta de un usuario se realiza en una pantalla que mostrará los diversos niveles de acceso que pueden ser concedidos, los datos del usuario y la clave asignada, así como un conjunto de botones para proceder con la actualización o el alta del usuario o cancelar el proceso, ésta pantalla tendrá más o menos la siguiente apariencia:

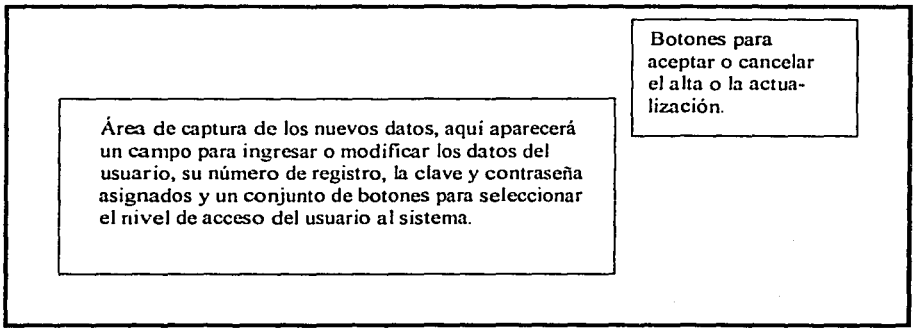


Figura 32.
Módulo de usuarios 2.

Módulo de Catálogos.

Existen varios catálogos dentro del sistema *Sofilab*, estos corresponden a las diversas áreas y tareas que se realizan en los laboratorios y principalmente serán utilizados para asistir en la selección de tipos de pruebas, perfiles y búsqueda de usuarios, entre otras aplicaciones. Se ha determinado la creación de catálogos de Sistemas, Unidades, Perfiles, Pruebas, Reactivos, Secciones y Servicios.

La forma principal de cada catálogo se verá de la siguiente manera:

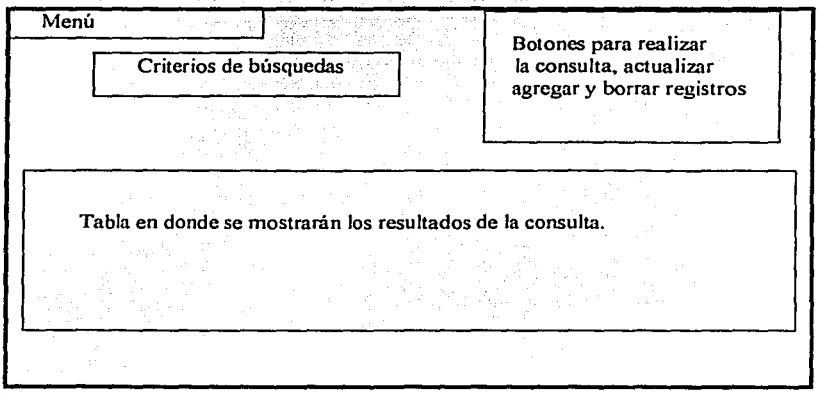


Figura 33.
Módulo de catálogos 1.

Dentro del menú se encontrarán las opciones de impresión y guardado a archivo de los registros obtenidos de la consulta realizada, el *ComboBox* proporcionará al usuario una forma amigable de seleccionar el tipo de consulta que desee realizar, por ejemplo podrá realizar consultas por número de registro, por descripción, por nombre, etc. Estas opciones aparecerán de antemano en el *ComboBox* con lo cual el usuario solo tendrá que hacer "*clic*" con el ratón o presionar la tecla "*Entrar*" para realizar la consulta. Cada catálogo le permitirá al usuario realizar una actualización de un registro, ingresar uno nuevo o borrar alguno, dependiendo del nivel de acceso asignado a ese usuario.

Cada catálogo se actualizará de manera similar, es decir al momento de ingresar, borrar o actualizar un registro, se mostrará una nueva forma en donde aparecerá el número de registro correspondiente, el nombre o descripción de dicho registro, en el caso de catálogos como el de sistemas, pruebas y perfiles aparecerán nuevos campos para su llenado como la de nombre del fabricante, función e incluso una nueva tabla en donde se seleccionará por ejemplo para el caso de perfiles, el nombre y tipo de pruebas que conformarán el nuevo perfil, la forma básica de esta pantalla es la siguiente:

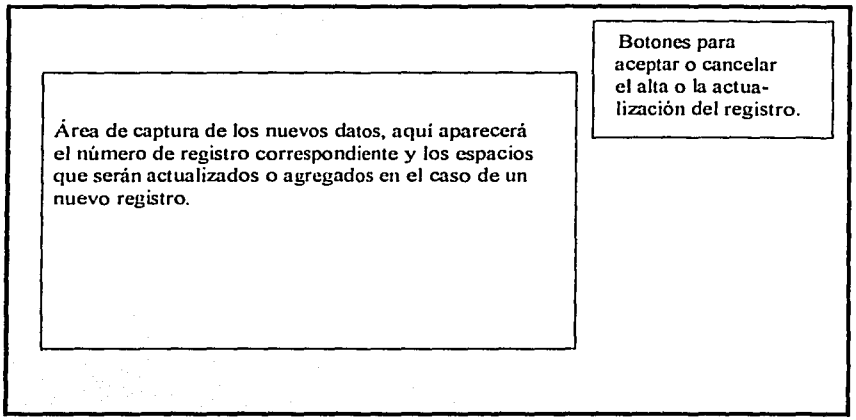


Figura 3.4.
Módulo de catálogos 2.

Módulo de Mantenimiento.

Este es el módulo más sencillo del sistema, en el sentido de que el trabajo a realizar en éste módulo realmente lo realizará el *SQL Server*, las funciones a efectuar serán

- Respaldo de la base de datos.
- Truncar el "*LOG*" de transacciones, es decir, limpiar el registro donde se realiza la transferencia de información de la base de datos.
- Actualizar los índices de las tablas.
- Eliminar los registros de las citas no realizadas.
- Determinar el número máximo de citas diarias.

Estas opciones se mostrarán con un conjunto controles *Option_Buttons* de *Visual Basic*, lo que facilitará la selección de la función de mantenimiento a realizar, la pantalla tendrá más o menos la siguiente forma:

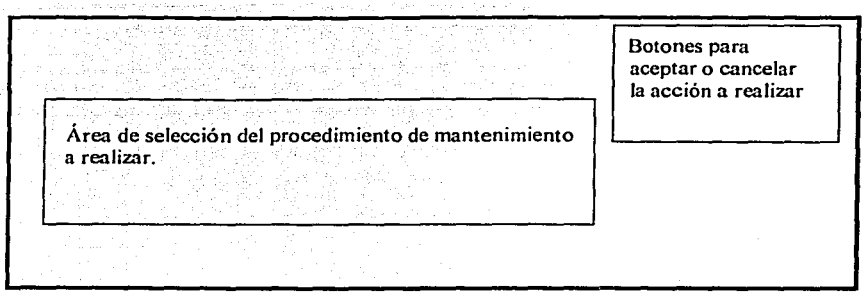


Figura 35.
Módulo de mantenimiento.

Módulo de Citas.

Es aquí donde se encuentra la pantalla principal de captura, dentro de la cual se pretende agilizar la captura de datos generales de los pacientes como la programación de citas.

El módulo de citas deberá estar constituido por cuatro partes. La primera permitirá una búsqueda rápida para un paciente registrado o la posibilidad de agregarlo en ese momento, para estos también existirá una distinción de los que ya tienen asignado un número de expediente y los que no.

Una vez indicado el paciente se procederá a mostrar otra pantalla los datos demográficos de éste con el fin de corregir o registrar estos, en esta misma pantalla se indicará además el tipo de servicio correspondiente para este paciente.

La pantalla siguiente permitirá la asignación de pruebas, las cuales pueden ser solas o pertenecientes a uno o varios perfiles.

Por último otra pantalla indicará la fecha de la cita, así como el folio que le corresponde.

La primera pantalla de citas.

Como se nuestra en la siguiente figura, se tendrá una caja de texto para recibir el número de expediente, indicadores de la fecha y hora para referencia al usuario, Un botón para la no asignación de un número de expediente. Una búsqueda por nombre de los pacientes, así como los botones de salir y siguiente.

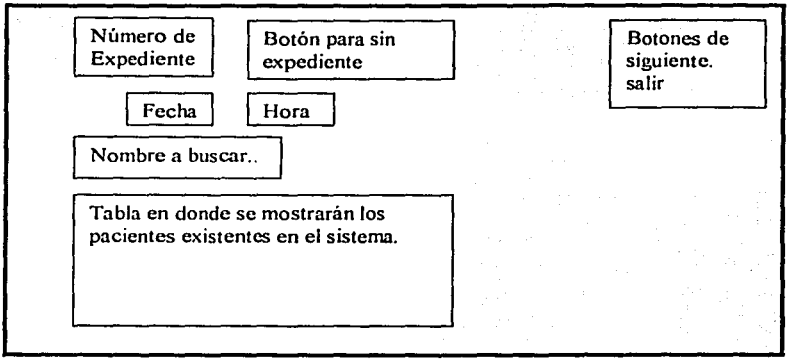


Figura 36.
Módulo de citas 1.

Si se selecciona siguiente aparecerá la pantalla mostrada en la Figura 37 donde apreciamos que está diseñada para capturar los datos demográficos del paciente

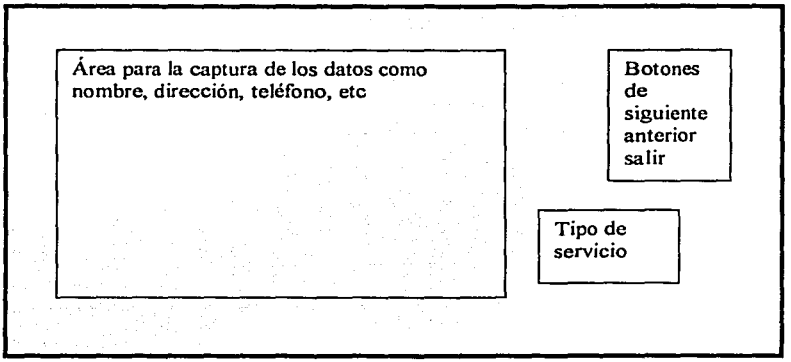


Figura 37.
Módulo de citas 2.

La pantalla de asignación de pruebas mostrará tanto la búsqueda de pruebas como las ya asignadas en dos vistas de datos diferentes permitiendo a través de estas el alta y baja de pruebas para ese paciente.

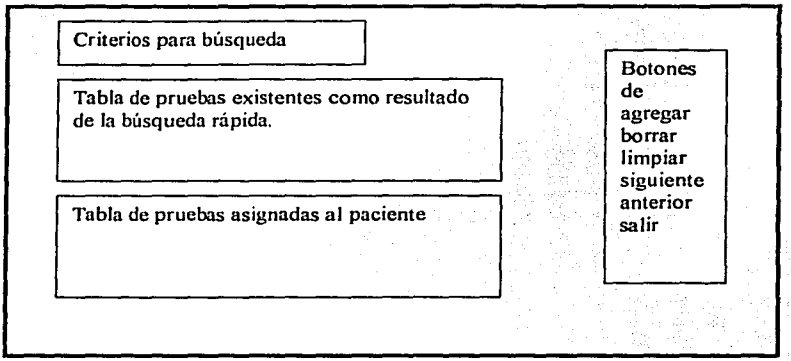


Figura 38.
Módulo de citas 3.

Una última pantalla mostrará la fecha actual y la de la cita sugerida en un pequeño calendario y el folio en una caja de texto.

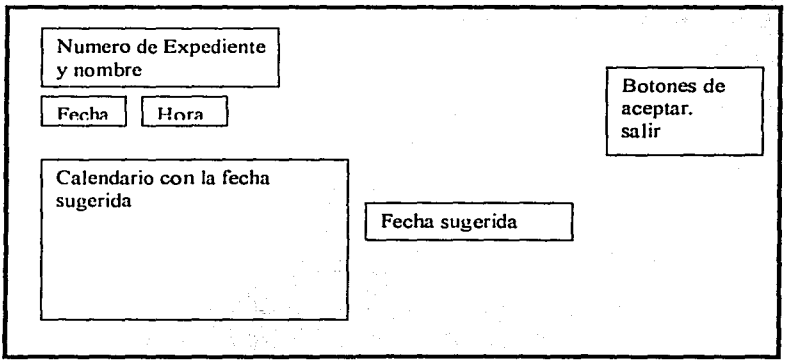


Figura 39.
Módulo de citas 4.

Módulo de Admisiones.

El módulo de admisiones permite visualizar cada cita asignada de forma tal que se pueda admitir a un solo paciente confirmando visualmente sus pruebas y haciendo clic en un botón o a un grupo seleccionado con un solo clic sin necesidad de verificar las pruebas.

Se deberá poder modificar las pruebas de una cita, así como también poder admitir directamente a un paciente sin necesidad de una cita previa.

La pantalla principal de este módulo mostrará las citas pendientes del día y la opción para diversos tipos de búsqueda de folio, número de expediente, nombre y fecha. Con solo cambiar una opción se deberán mostrar las admisiones ya procesadas para así corregir, de ser necesario, las pruebas de un paciente.

La siguiente figura muestra que la primera pantalla de citas está compuesta por una tabla de búsqueda general, un control de *VB CheckBox* para habilitar y deshabilitar la opción de los de hoy, una *ComboBox* despliega las opciones para la búsqueda, a través de dos botones de opción se habilitará el despliegue de admisiones de citas, los botones de la derecha permitirán dependiendo de si es Citas o Admisiones, habilitar aceptar para la búsqueda, admitir, reprogramar, eliminar, actualizar pruebas y salir del módulo.

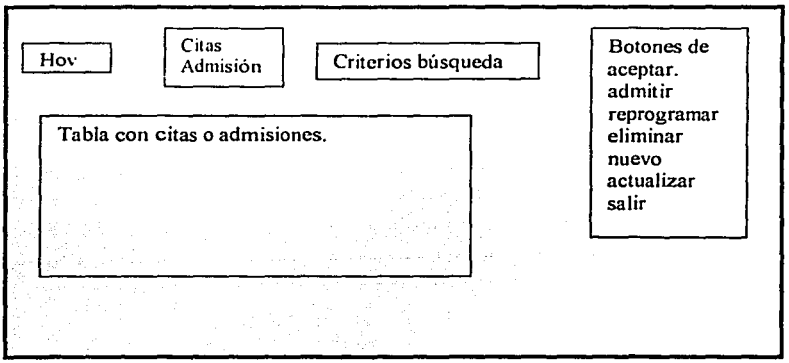


Figura 40.
Módulo de admisiones 1.

La pantalla de Admitir muestra las pruebas asignadas en una tabla y las existentes en otro para poder modificar de ser necesario así como los botones correspondientes de Agregar, Eliminar y Limpiar, el botón Aceptar admite la cita mostrada.

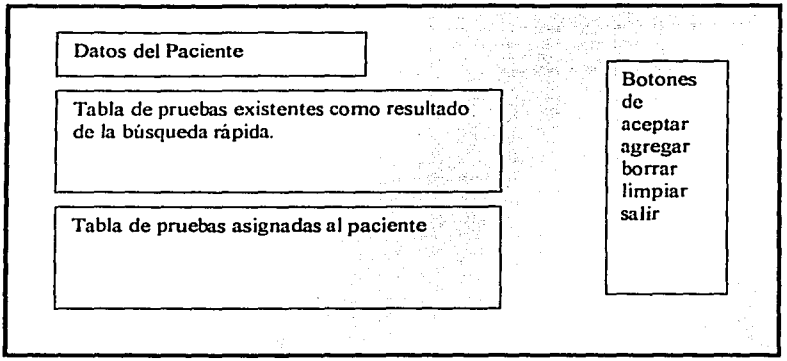


Figura 41.
Módulo de admisiones 2.

La pantalla mostrada por el botón reprogramar trae el calendario y la fecha de la cita, además mostrará datos básicos del paciente como nombre, expediente y folio.

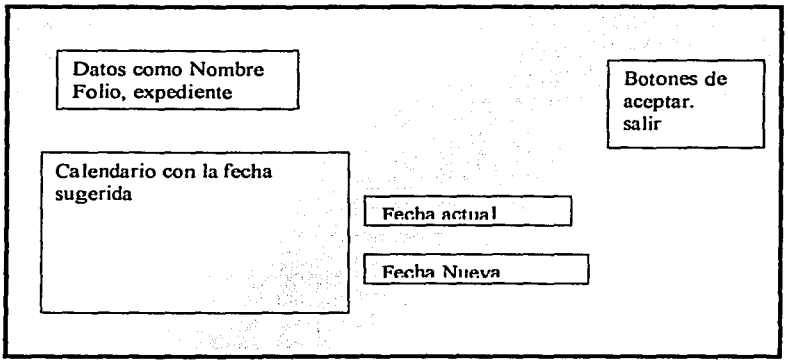


Figura 42.
Módulo de admisiones 3.

El botón de Nuevo traerá las mismas pantallas del módulo de citas con la diferencia que al terminar el paciente será admitido.

Módulo de Reportes Generales y de Resultados.

Módulos sencillos que básicamente están constituidos por un menú para seleccionar el reporte requerido e imprimirlo, se decidió elegir una pantalla completa para estos reportes debido a que sería más fácil para el usuario la búsqueda y selección de un reporte específico.

Los reportes que pueden imprimirse en ésta pantalla son para uso administrativo del laboratorio. Básicamente son tres grupos de reportes: Citas, Admisiones y Relaciones enviadas al archivo.

La pantalla muestra botones de opción con los tipos de reporte.

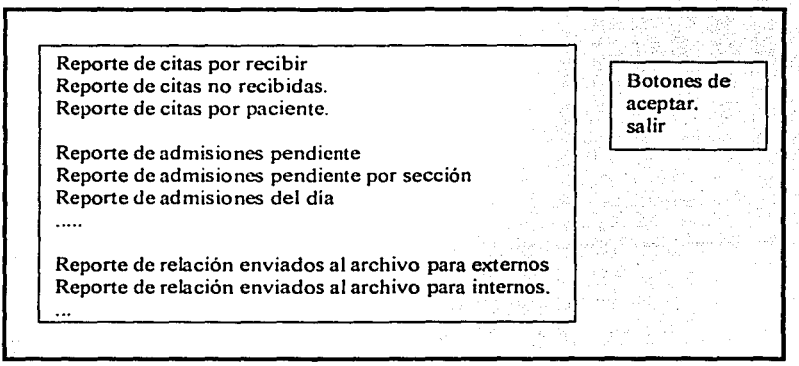


Figura 43.
Módulo de reportes generales.

Los Reportes de Resultados como su nombre lo indica son usados para ser entregados al paciente que a su vez entregará al médico, estos reportes se imprimen con opción de fecha o paciente, existe por requerimientos del laboratorio una sección de reportes para la sección de bacteriología.

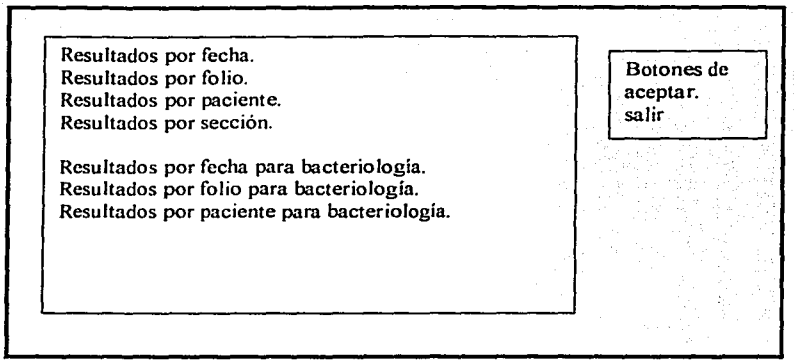


Figura 44.
Módulo de reporte de resultados.

Módulo Registro de resultados.

Uno de los módulos más importantes del sistema, se utiliza para poder capturar manualmente aquellas pruebas que no se obtengan vía interfaces, este módulo es utilizado principalmente por los químicos y personal de las secciones del laboratorio, les permite además asignar y corregir las pruebas de un folio específico.

La primera pantalla muestra una lista de los folios del día y opciones de búsqueda como son por Folio, Número de Expediente, Nombre, Fecha y Sección. Es posible seleccionar un folio específico y actualizar sus pruebas y asignar o cambiar de ser necesario a través de la pantalla de asignación mostrada en los módulos anteriores (figura 45).

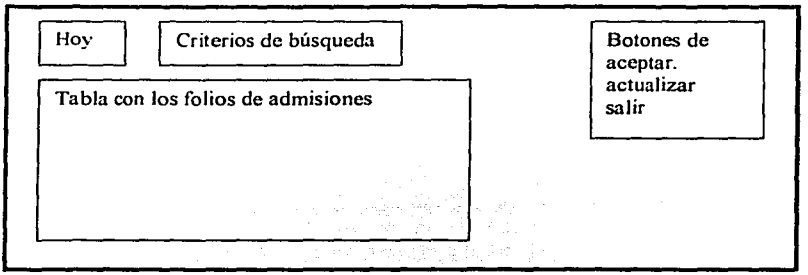


Figura 45.
Módulo de registro de resultados 1.

La pantalla de la figura 46 aparece al seleccionar actualizar los resultados en esta se muestra una tabla donde pueden actualizarse los resultados de las pruebas así como los correspondientes botones de aceptar y salir, otro botón trae la pantalla de asignación de pruebas.

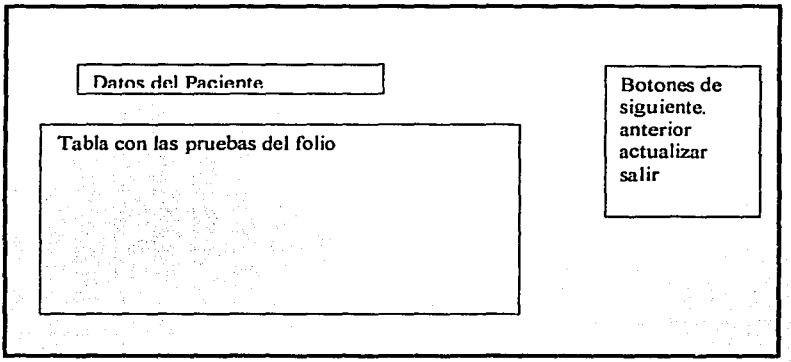


Figura 46.
Módulo de registro de resultados 2.

Módulo de estadísticas.

En el módulo de estadísticas se contempla realizar un control estadístico de los diversos accesos es decir, se pretende observar el número de veces que los usuarios accedan al sistema durante el día, la hora de entrada, los tipos de operaciones que realizaron como dar de alta pacientes, pruebas, ver resultados, etc. Todo esto con la finalidad de llevar un control estricto de los procesos realizados en el laboratorio, con esto será posible verificar por ejemplo cuando se pierda un registro de resultados en la recepción, si las pruebas del paciente ya fueron corridas, quien las corrió y a que hora y día se realizaron, a su vez se podrá evaluar que módulos son los más accesados, cuales los menos utilizados y cuales no lo son, esto con la finalidad de que pensando en futuras versiones del sistema, observar que mejoras se pueden realizar, cuales módulos se podrían omitir y que módulos mejorar, en la Figura 47 se esquematiza el tipo de pantalla que se mostrará al usuario, esta contará con una tabla que mostrará los resultados de los diversos criterios de búsqueda del usuario, por ejemplo una vista general de que usuarios están activos en el sistema, a que hora accesaron y que están haciendo, los criterios de búsqueda se mostrarán en un control de *VB* llamado *ComboBox* en donde el usuario seleccionará el que más le convenga, si por ejemplo el usuario quisiera realizar una búsqueda por nombre, se desplegará un control *VB Text Box* en donde el usuario pondrá el nombre a buscar, los resultados se podrán imprimir en un reporte diseñado en *Sheridan Crystal Reports* que contendrá lo desplegado en la tabla, la pantalla es más o menos como se muestra en la siguiente figura.

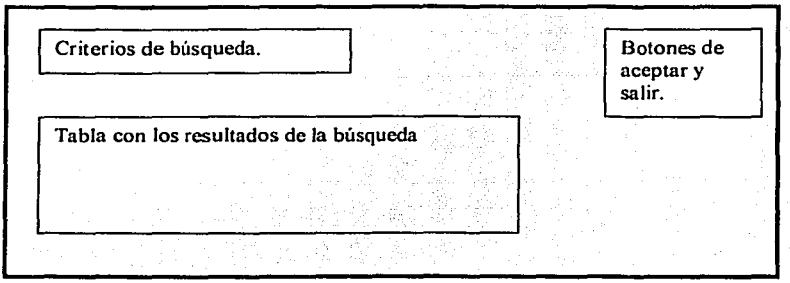


Figura 47.
Módulo de estadísticas.

3.4 Estructura de la Interfaz.

Las interfaces para el *Sistema Integral De Laboratorio Clínico Sofilab* representan la base de la automatización debido a que, gracias a estas aseguramos un manejo efectivo y rápido de los resultados obtenidos de los diversos equipos del laboratorio clínico. Una interfaz como ya mencionamos es un punto en el que se establece la conexión entre dos elementos que les permite trabajar juntos, en éste caso refiriéndonos al sistema de laboratorio clínico y a los equipos que éste contiene. Básicamente la interfaz interpreta los resultados de una o varias pruebas y los transfiere al sistema, la siguiente figura muestra una representación general de una interfaz conectada al sistema.

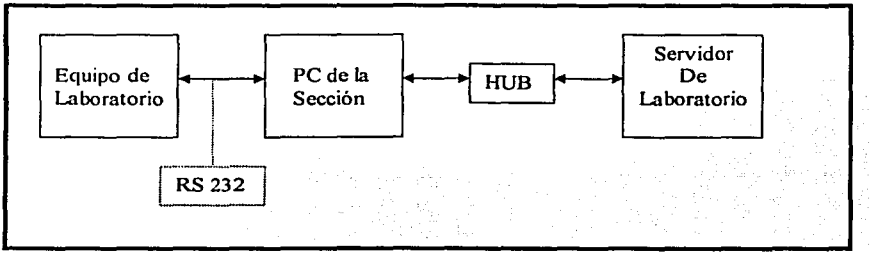


Figura 48.
Estructura de una interfaz.

Cada interfaz varía para cada equipo, debido a que por lo general cada equipo utiliza un protocolo de comunicación diferente aunque muchas veces la interfaz gráfica es la misma, dependiendo de los requerimientos de la sección de laboratorio en la que se encuentre.

Una interfaz para el sistema integral de laboratorio clínico debe basarse como cualquier otro medio de comunicación de red en las siete capas del modelo *OSI*, es así que para su diseño requerimos involucrarnos en todas las capas de una u otra forma.

En la capa física se diseñó el cable de acuerdo con las especificaciones del fabricante del equipo. En la capa de enlace es la más difícil de diseñar debido a que es necesario aplicar un formato delicado para integrarla a las tramas de comunicación debidos y su transmisión y recepción también deben estar bajo los estándares de cada equipo. La capa de Red y de Transporte se utilizan en la comunicación de la interfaz con la red local y hacia el servidor, debido a que la comunicación entre el equipo de laboratorio y la estación de trabajo se realiza de punto a punto. En lo que concierne a la capa de sesión podemos destacar que esto está directamente ligado con el uso del equipo a través de los usuarios del laboratorio al tener que validarse cuando ingresan al sistema y equipo de laboratorio. Las capas de presentación y aplicación están constituidas dentro del código y la presentación de la interfaz en sí mismos.

CAPÍTULO IV.
CONSTRUCCIÓN Y ADAPTACIÓN.

El Sistema Integral de Laboratorio Clínico Sofilab, es un sistema de gran magnitud con muchos procedimientos almacenados diferentes, esto en consecuencia limitó la estructuración del contenido del presente capítulo al no ser posible presentar de manera integra el código y los procedimientos almacenados utilizados para la construcción del sistema, por lo tanto se presentan algunas partes del código y algunos procedimientos almacenados que corresponden a las partes más importantes del sistema, así mismo se presenta de manera integra un ejemplo del código de programación utilizado para el desarrollo de las interfaces que acompañan al sistema.

El sistema está desarrollado en *Microsoft Visual Basic 6* y la base de datos fue construida sobre *Microsoft SQL Server 7* y gracias a las nuevas tecnologías de conexión para bases de datos, la probabilidad de haber utilizado alguna otra plataforma de desarrollo para la base de datos del sistema es bastante amplia, pudiéndose haber desarrollado la base en un *Access* o en un *ORACLE* en lugar de *SQL Server*, esto es debido a que *Microsoft* ha considerado a las bases de datos como una de las principales aplicaciones de *Visual Basic*. Con el transcurso de los años se le han ido agregando a *Visual Basic* nuevas herramientas para que los desarrolladores dispongan en todo momento de las últimas tecnologías al alcance de sus manos; sin embargo a medida que se le han agregado nuevas herramientas, las anteriores no han sido descartadas, sino que se han mantenido como parte de Visual Basic para poder asegurar la compatibilidad con los programas antiguos, esto es necesario ya que conforme se introducen nuevos modelos para acceder a datos no es posible desprendernos de los antiguos sin más, ya que éstos son necesarios para soportar las aplicaciones antiguas que fueron escritas utilizando éstas tecnologías.

En el capítulo II de la presente tesis se mencionaron algunos aspectos concernientes a las bases de datos, a continuación se explican algunas de las tecnologías que utiliza *Visual Basic* para realizar la conexión a los diversos manejadores de bases de datos existentes.

OLE DB .- Son un juego de interfaces *COM* (*Component Object Model / Componentes de Objetos Modelo*) que permite acceder a los programadores a diversas fuentes de información. El objetivo de un *OLE DB* es proveer acceso de datos universal (acceso que puede manejar cualquier tipo de datos, independientemente de su formato, y que no está restringido a determinadas fuentes). Las interfaces *OLE DB* soportan las posibilidades de manejo de datos de cada tipo de fuente de datos. No se accede directamente a *OLE DB* desde Visual Basic, sino que se hace indirectamente vía *ADO* (*Active X Data Objects / Objetos de Datos Active X*).

ADO .- (*Active X Data Objects Objects / Objetos de Datos Active X*) provee a los programadores de Visual Basic una interfaz a nivel de aplicación a *OLE DB* por lo que se puede inferir que *ADO* es una interfaz directa a la fuente de datos. *ADO* es una herramienta de recién incorporación a *Visual Basic* y es considerada por *Microsoft* como su tecnología cumbre de acceso a datos, las versiones anteriores de *Visual Basic* utilizaban herramientas *RDO* (*Remote Data Objects / Objetos de Datos Remotos*) y *ADO* (*Data Access Objects /*

Objetos de Acceso a Datos). Estas tecnologías siguen siendo admitidas por *Visual Basic* como mencionamos de manera previa para hacer compatible el código de versiones anteriores de *Visual Basic* con la actual.

ODBC .- (*Open Database Connectivity / Conectividad Abierta para Base de Datos*) provee una interfaz de programación de aplicaciones (*API*) que contiene procedimientos para realizar diversas tareas de manipulación de datos. Un programa llama a estos procedimientos *API*, cuando es necesario, y el manejador del controlador *ODBC* pasa las llamadas al controlador adecuado, esto significa que hace una llamada al controlador que está designado para la fuente de datos particular que está en uso. De aquí es de donde viene la parte "*OPEN*" (Abierta) del nombre *ODBC*. Cualquier fabricante de bases de datos puede escribir *ODBC* para su propio formato de datos.

ODBC es considerado generalmente como obsoleto. La única razón de utilizar *ODBC* en un proyecto de bases de datos nuevo es cuando necesitamos acceder a una fuente de datos que está en un formato poco conocido, no soportado por *ADO*, para el cual existe un controlador *ODBC*. Aún en esta situación es probable que siga siendo mejor utilizado *DAO*, porque admite bases de datos *ODBC*.

DAO o ADO .- (*Data Access Objects / Objetos de Acceso a Datos*) Es similar a *ODBC* en el sentido que provee una *API* que un programa puede llamar para realizar tareas de manipulación de datos. *DAO* se diferencia de *ODBC* en que utiliza el motor de bases de datos *JET* de Microsoft, en lugar de un controlador proporcionado por el fabricante de bases de datos. *DAO* está optimizado para trabajar con archivos de fuentes de datos con formato *MDB* (el mismo formato usado por el programa de bases de datos *Microsoft Access*). Sin embargo, *DAO* también soporta bases de datos ODBC, así como fuentes de datos en diversos formatos, incluyendo *PARADOX, Fox Pro, dBase, Excel* y *Lotus 1-2-3*.

RDO .- El modelo de programación *RDO (Remote Data Object / Objetos de Datos Remotos*) fue desarrollado específicamente para tratar con los requerimientos especiales del acceso a fuentes de datos vía red, referido como acceso de datos remoto. *RDO* funciona añadiendo una capa encima de *ODBC* para manejar las necesidades especiales del acceso remoto, como establecer conexiones y crear grupos de resultados, en algunas situaciones *RDO* trabaja con *DAO* para realizar las tareas exigidas. *RDO* puede implantarse totalmente con código a través del control *Remote Data*.

Estas son las tecnologías que utiliza *Visual Basic* para realizar la conexión a las diversas fuentes de datos existentes, para el caso específico del *Sistema de Laboratorio Clínico Sofilab* se utilizó *ADO* con *OLE DB* para realizar las conexiones a *SQL*, esto es simple, se utilizó la última tecnología disponible de *Microsoft*, además de que como se mencionó anteriormente, *ADO* permite la conexión con mucha más facilidad a otras bases de datos como *ORACLE*.

4.1 Creación y estructuración de la base de datos.

Recordando que el tipo de modelo de desarrollo del sistema es el incremental, la estructura de la base de datos ha sufrido un crecimiento principalmente en los procedimientos almacenados apegándose a las crecientes demandas del laboratorio, manteniendo su base en el diseño inicial.

Para la construcción de la base de datos se partió del diccionario de datos previamente diseñado y en el diagrama entidad – relación mostrados en el capítulo anterior, para ello se utilizó la herramienta de *MSQL Server* llamada *SQL Enterprise Manager.*

Para una mejor comprensión de los pasos a seguir para la creación de la base de datos listaremos las tareas realizadas a través de esta herramienta.

- Creación de la base de datos vacía (estructura principal del manejador).
- Creación de las tablas.
- Construcción de índices y restricciones.
- Creación de usuarios.
- Creación de procedimientos almacenados.
- Llenado de las tablas con información.
- Asignación de permisos a usuarios.

A continuación se detallaran cada uno de dichos puntos.

4.1.1 Creación de la base de datos vacía.

La interfaz gráfica de *SQL Enterprise Manager* permite utilizar un asistente para la creación de la base de datos que se muestra en la siguiente figura, en donde a través del nombre de dicha base de datos se crean dos archivos principales denominados *Log* y *Data*, que contendrán la información referente al manejo de la base de datos (*Transaction Log*) y la estructura y contenido de la información (*Data*).

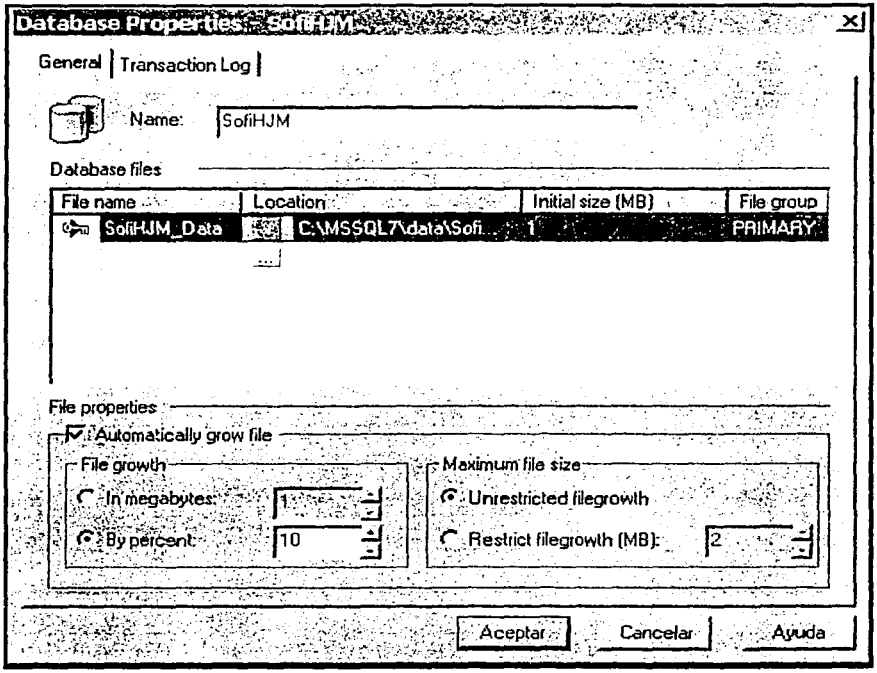
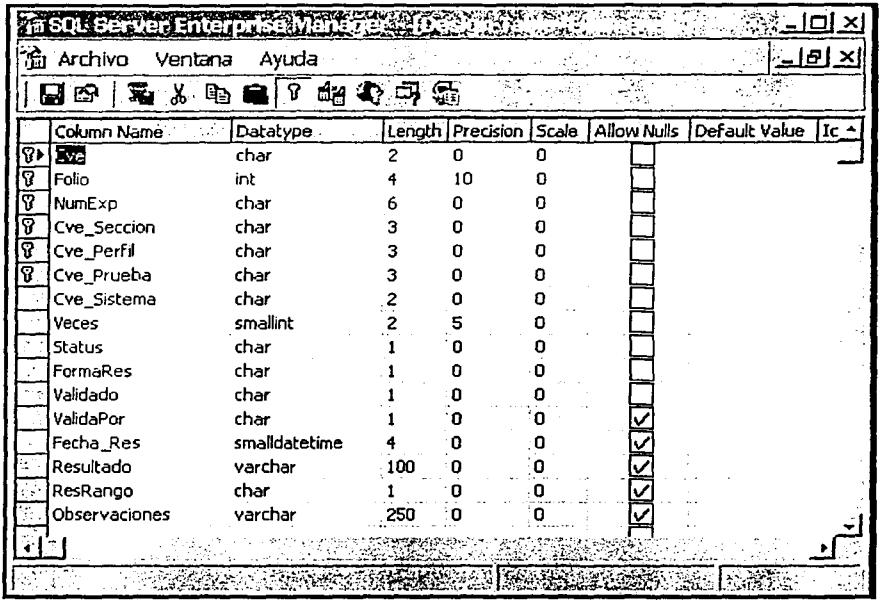


Figura 49.
Propiedades de la base de datos.

En esta pantalla es posible asignar el tamaño inicial de la base de datos, el cual puede ser fijo o variable, para el presente caso se decidió tomar los valores predeterminados de *SQL* y dejar que la base de datos ajustara el tamaño de la misma conforme vaya creciendo.

4.1.2 Creación de las tablas.

Una vez realizado lo anterior, se procede a crear las tablas pertenecientes a esta nueva base de datos, para ello se utiliza una pantalla que se muestra a continuación, en donde se establecen los nombres de los campos, el tipo de dato de cada campo, tamaño, si permite datos nulos o no, si contiene valores predeterminados, si permite valores repetidos o no, entre otras cosas.



Column Name	Datatype	Length	Precision	Scale	Allow Nulls	Default Value	Ic
Cve	char	2	0	0			
Folio	int	4	10	0			
NumExp	char	6	0	0			
Cve_Seccion	char	3	0	0			
Cve_Perfil	char	3	0	0			
Cve_Prueba	char	3	0	0			
Cve_Sistema	char	2	0	0			
Veces	smallint	2	5	0			
Status	char	1	0	0			
FormaRes	char	1	0	0			
Validado	char	1	0	0			
ValidaPor	char	1	0	0	✓		
Fecha_Res	smalldatetime	4	0	0	✓		
Resultado	varchar	100	0	0	✓		
ResRango	char	1	0	0	✓		
Observaciones	varchar	250	0	0	✓		

Figura 50.
Pantalla de diseño de tablas.

En la figura anterior, se puede observar en el lado izquierdo de algunos campos un icono en forma de llave, el cual representa que dichos campos forman la llave primaria de la tabla.

4.1.3 Construcción de índices y restricciones.

Los índices de cada tabla se crean automáticamente en esta versión de *SM SQL Server* pero es necesario que en algunos se alteren propiedades como la de único o agrupado, para ello se utiliza la siguiente pantalla de la herramienta *SQL Enterprise Manager*

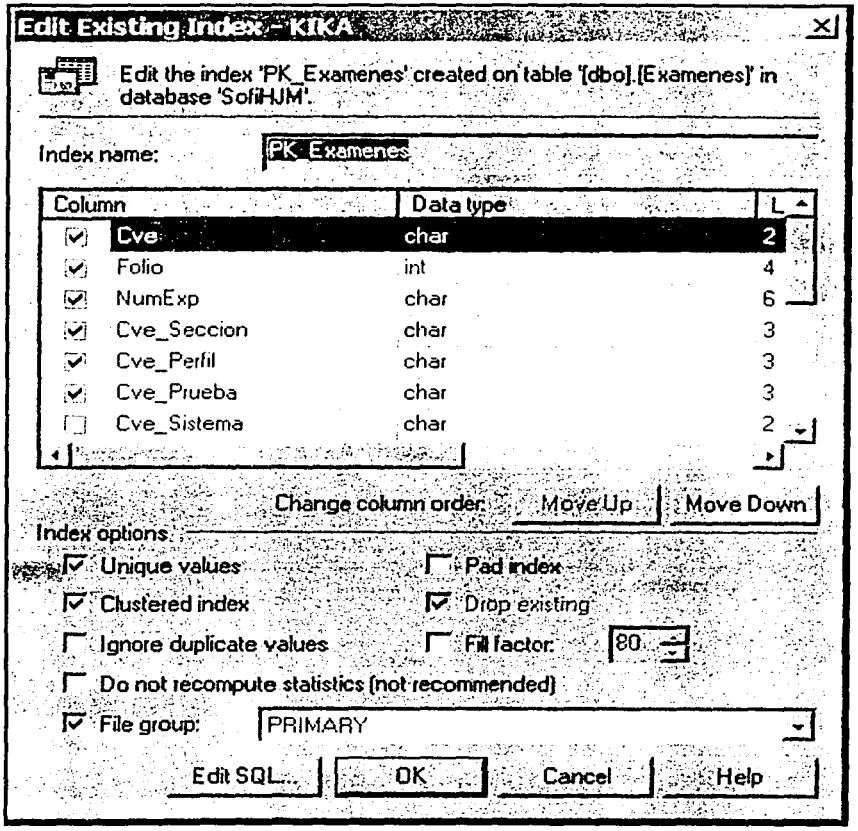


Figura 51.
Pantalla de creación de indices.

Las llaves primarias creadas en cada tabla y las características de los campos como por ejemplo el de si puede o no ser nulo, junto con las llaves foráneas se denominan restricciones las cuales permiten mantener la integridad de la información. Para crear las llaves foráneas se utilizó la interfaz gráfica para diagramas de la base de datos en donde es posible indicar la relación entre tablas, la siguiente figura ejemplifica la creación de una llave foránea mediante dicha interfaz gráfica.

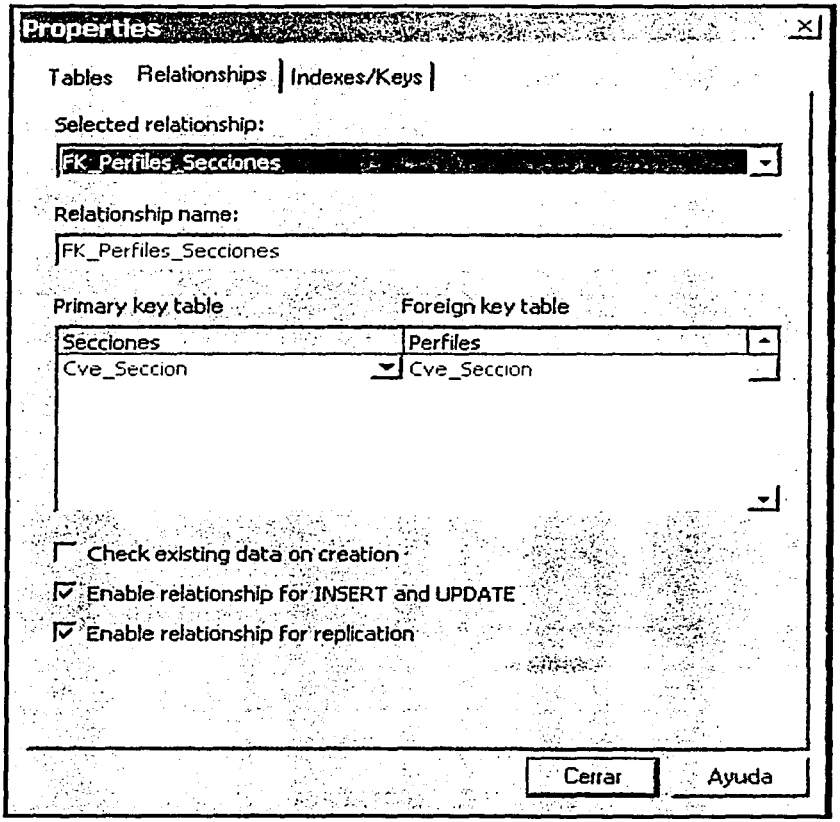


Figura 52.
Pantalla de llaves foráneas de tablas.

4.1.4 Creación de usuarios.

La creación de usuarios es importante en el desarrollo de una base de datos, ya que a partir de ellos se determinarán los diversos niveles de acceso que podrán tener dentro de la base de datos.

La creación de usuarios dentro del *SQL Enterprise Manager* es relativamente simple, de igual forma que en los pasos anteriores se utiliza una herramienta que nos auxilia en la creación de los usuarios, gracias a esta herramienta se evita la creación de comandos específicos en lenguaje *SQL* y en consecuencia se agiliza esta actividad, la siguiente figura muestra la herramienta utilizada.

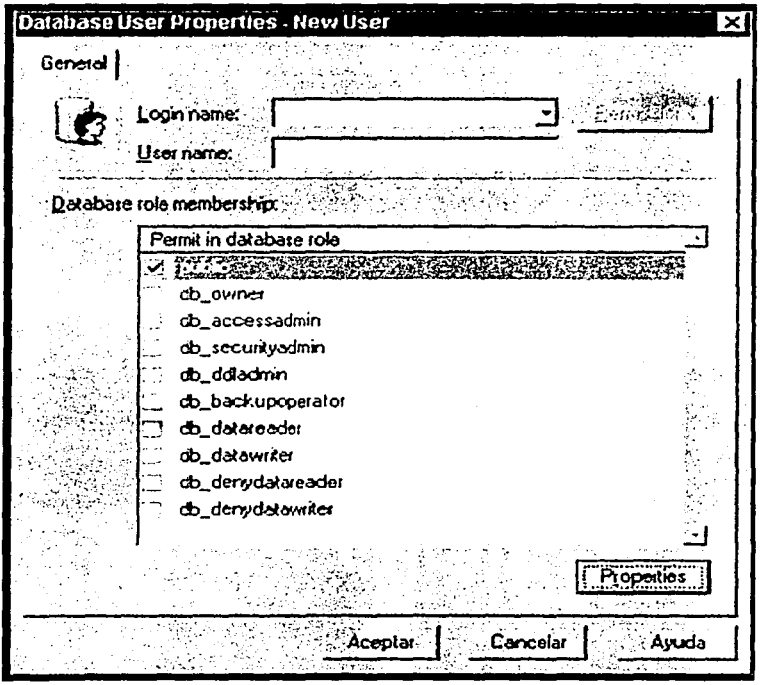


Figura 53.
Pantalla de alta de usuarios.

En esta pantalla es posible ingresar el nombre que el usuario utilizará para acceder a la base de datos (*Login Name*) y su nombre real para poder identificarlo (*User Name*), así mismo, se puede definir el tipo de usuario que será (administrador, público, lector de datos, etc.) y los permisos que tendrá sobre los diversos objetos existentes en la base de datos.

4.1.5 Creación de procedimientos almacenados.

Una parte fundamental del sistema son sin duda los procedimientos almacenados, éstos son instrucciones de *SQL* precompiladas que realizan ciertos tipos de consultas predefinidas por el usuario. Dentro de la versión 7 de *Microsoft SQL Server* existen dos herramientas para crear los procedimientos almacenados, el *SQL Query Analyzer* y una opción existente dentro de la herramienta *Enterprise Manager* que permite su creación o edición.

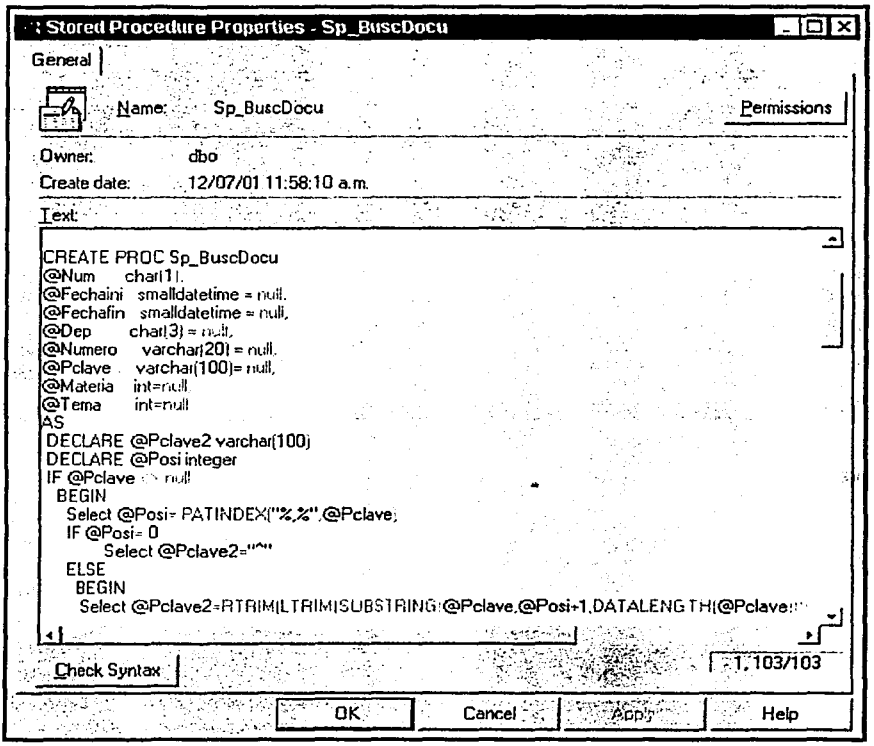


Figura 54.
Herramienta de edición del Enterprise Manager.

Dentro de la pantalla del *Enterprise Manager* es posible editar o crear un nuevo procedimiento almacenado, así mismo cuenta con un botón que realiza una revisión en la sintaxis del procedimiento almacenado auxiliándonos en la elaboración de los mismos.

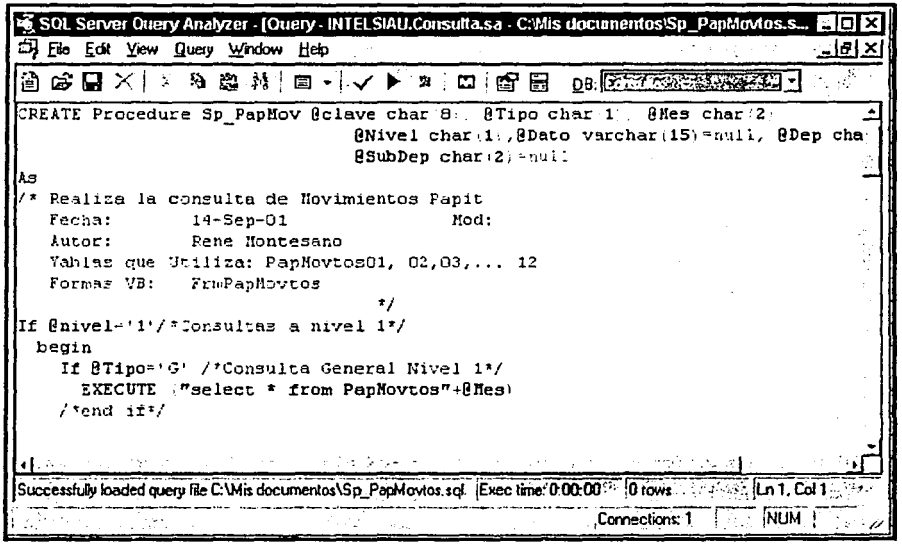


Figura 55.
SQL Query Analyzer.

Dentro de la herramienta *SQL Query Analyzer* es posible redactar el procedimiento almacenado, más no modificarlo sin antes borrar el ya existente, la ventaja de esta herramienta consiste en que es posible ejecutar las instrucciones que compondrán el procedimiento almacenado permitiéndonos una mejor manipulación de las instrucciones asegurando así su funcionalidad.

4.1.6 Llenado de las tablas con información.

Una vez terminada la creación de las tablas y procedimiento almacenados de la base de datos es necesario empezar a llenar las tablas con información, esto es posible hacerlo de diversas maneras, exportando los datos de un archivo de texto o de otra base de datos, de forma manual dentro del *Enterprise Manager* o mediante la instrucción *Insert* de *SQL* detallada a continuación.

```
INSERT [Into ]
    (
     table_name WITH ( <table_hint_limited> [...n])
    | view_name
    | rowset_function_limited
    )

    (       [(column_list)]
            [ VALUES ( (    DEFAULT
                                        |       NULL
                                        |       expression
                                        )[,...n]
                    )
            | derived_table
            | execute_statement
            )
    )
    | DEFAULT VALUES
```

Figura 56.
Estructura de la instrucción Insert.

La importación de datos por medio de archivos de texto es conocida también como "*Bulk Copy*", en este archivo los datos se encuentran separados por un caracter en especial como un espacio en blanco o una @, la pantalla que realiza la importación de datos es la siguiente.

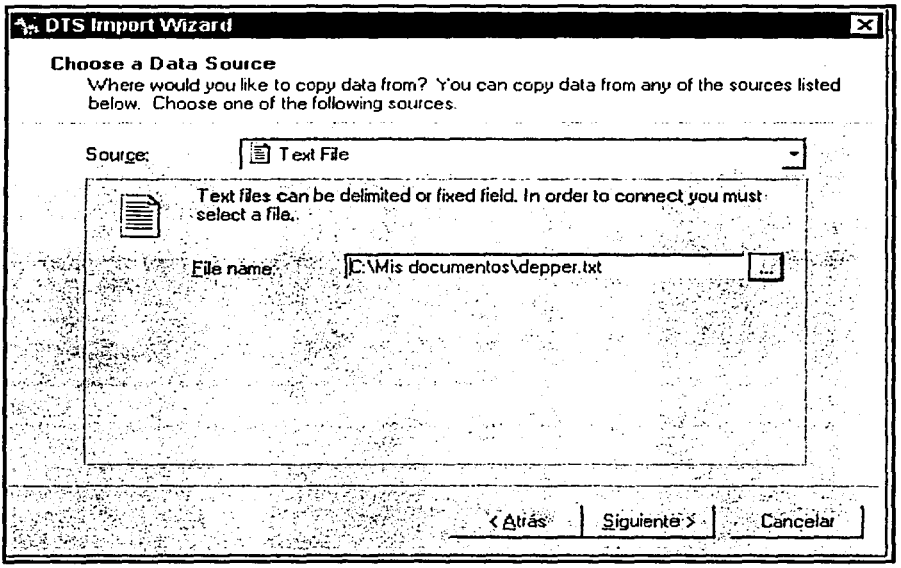


Figura 57.
Pantalla de importación de datos.

Es importante mencionar que *MSQL Server 7* nos permite importar datos no solamente de archivos de texto, esta herramienta nos permite importar datos de hojas de cálculo como *Excel*, de otras bases de datos como *Oracle* y *MS Access* y de otras fuentes de datos.

En nuestro caso particular fue necesaria la creación de un pequeño programa que adaptará los datos existentes del sistema basado en *MS Access* a las tablas del sistema.

4.1.7 Asignación de permisos a usuarios.

Los permisos dentro de una base de datos son esenciales para su buena administración, los usuarios existentes de la base de datos deben de ser clasificados de manera jerárquica, esto es, debe de existir un usuario capaz de crear tablas, procedimientos almacenados, índices y todos los pasos de la creación de una base de datos, así mismo existen usuarios que sólo pueden consultar datos, otros que pueden ejecutar los procedimientos almacenados y consultar las tablas, etc. *SQL Enterprise Manager* nos permite manipular y asignar los permisos que los usuarios tendrán dentro de la base de datos dando así, un nivel extra de seguridad en la aplicación.

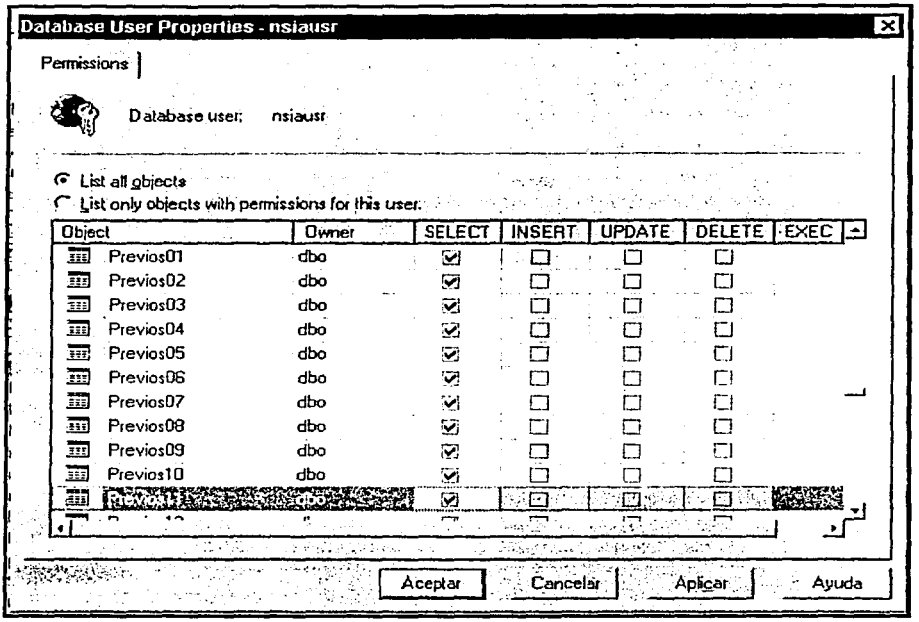


Figura 58.
Pantalla de manejo de permisos del SQL Enterprise Manager.

4.2 Desarrollo de procedimientos almacenados para las consultas.

Como se mencionó anteriormente los procedimientos almacenados son instrucciones *SQL* almacenadas en la base de datos, dichas instrucciones han sido compiladas y optimizadas por la base de datos con lo que su ejecución es mucho mas rápida que la ejecución directa de una sentencia *SQL* permitiendo así, aumentar el desempeño general del sistema, además cada procedimiento almacenado funciona como un pequeño programa que recibe y devuelve datos específicos permitiendo el control de la integridad de los mismos, la seguridad sobre los datos es otra de las ventajas de utilizar procedimientos almacenados en el desarrollo de un sistema debido a que a los usuarios se les restringen permisos de inserción, actualización o borrado de datos de forma directa siendo posible sólo a través de los procedimientos almacenados.

El sistema integral de laboratorio clínico se basa en procedimientos almacenados para realizar la gran mayoría de sus funciones con la base de datos, es así como se generan consultas almacenadas para las principales actividades tales como inserciones, búsquedas, actualizaciones, borrado de datos y en algunas consultas para reportes.

4.2.1 Ejemplos de procedimientos almacenados.

A continuación se muestran los procedimientos almacenados correspondientes al catálogo de reactivos.

```
CREATE PROCEDURE Sp_ReacBusca
@Opci char(1),@Datos varchar(25) = null
As
  If @Opci = "G" /* Busca general*/
    Select * from Reactivos
/*End If*/
  If @Opci = "C" /*Busca por clave*/
    Select * from Reactivos
    Where Cve_Reactivo=@Datos
 /*End If*/
   If @Opci= "D" /*Busca por descripción*/
     Select * from Reactivos
      Where Descripcion like "%" + @Datos + "%"
```

Figura 59.
Procedimiento almacenado de búsqueda de reactivos.

Lo anterior es un fragmento del procedimiento almacenado *Sp_ReacBusca* utilizado para realizar todas las consultas en lo que a reactivos se refiere, como podemos ver el procedimiento recibe como parámetros la opción de consulta y el dato particular para la búsqueda, la opción de "G" es utilizada para desplegar todos los reactivos existentes en la tabla correspondiente, la opción "C" realiza una búsqueda por clave en donde ésta se encuentra en el parámetro de datos, de forma similar la búsqueda "D" por descripción, busca un reactivo a partir de su descripción.

Para el caso de inserción se llama el siguiente procedimiento almacenado.

```
CREATE PROCEDURE Sp_ReacAltas @ReaCve char(2),@Descripcion
varchar(50),@unidad char(10),@reacanpru char(4),@reacanexi
char(4),@reacanmax char(4),@reacanmin char(4)
AS
Insert Reactivos Values
(@ReaCve,@Descripcion,@unidad,@reacanpru,@reacanexi,@reacanmax,
@reacanmin)
```

Figura 60.
Procedimiento almacenado de alta de reactivos.

Sp_ReacAltas recibe como parámetros los campos que serán insertados en la tabla de Reactivos, obsérvese que cada parámetro esta definido de un tipo específico de dato con lo que no es posible tratar de insertar un dato no válido en la tabla.

Sp_ReacBajas es el encargado de borrar un dato específico de la tabla de reactivos para lo que únicamente requiere la clave del reactivo que será eliminado.

```
CREATE PROCEDURE Sp_ReacBajas
@clave char(2)
As
   Delete from Reactivos
   Where Cve_Reactivo=@clave
```

Figura 61.
Procedimiento almacenado de baja de reactivos.

El manejo del resto de los catálogos es semejante en esencia al del catálogo de reactivos, en algunos fue necesario un procedimiento almacenado especial para la generación de reportes.

Algunos de los módulos principales del sistema como es el caso de las citas y el manejo de exámenes requieren de procedimientos almacenados más elaborados, en donde se considera también el orden de ejecución. A continuación veremos un par de procedimientos almacenados que ejemplifican el manejo de los puntos específicos del sistema.

```
CREATE Procedure Sp_CitCheca @Dato varchar(10)=null, @Res int OUTPUT
As
declare @MaxCitas int, @NumCitas int

select @MaxCitas=MaxCitas from Folios

select @Numcitas=(select count(*) from Cita_Admision where Tipo='C' and
Substring(Convert(Char(10),Fecha_Cit,101),1,10)=@Dato)

If @NumCitas < @MaxCitas
  Select @Res=0  /* Continua */
else
  Select @Res=1  /* Se excede */
/*Ebdif*/
Return
```

Figura 62.
Procedimiento almacenado de revisión de citas.

Sp_CitCheca verifica que las citas asignadas para un dia no excedan el máximo permitido por el laboratorio; de ser así, el procedimiento almacenado devuelve RES = 1 que es utilizado para notificar al usuario que debe cambiar la fecha de la cita.

El siguiente fragmento corresponde al procedimiento almacenado *Sp_CitReportes* el cual se utiliza para desplegar los campos de la tabla de citas, recibiendo como parámetro el tipo de reporte requerido, para éste caso las citas de un paciente especifico. De forma similar despliega las citas para un rango de fechas, una sección, etc.

```
CREATE Proc Sp_CitReportes @Num char(1), @Fech1 char(10)=null, @Fech2
char(10)=null, @Dato varchar(6)=null,  @CVE char(2)=null
As
If @Num='2'  /* Citas Por Paciente */
 select (max(C.cve)+cast(C.Folio as varchar(6))) as Folio, max(A.Nombre) as
Nombre, max(A.Cama) as Cama,
          Tipo=Case max(C.Cve)
          When 'E-' Then 'Externo'
          When 'I-' then    'Interno'
          else  'Urgencias'
           End,
      max(C.NumExp) as Expediente, max(C.Fecha_Cit) as 'Fecha de Cita',
      max(R.Cve_Seccion)as Seccion,  Perfil=Case max(P.IniPerf)
                                  When 'S/P' then max(U.IniPru)
                                  else max(P.IniPerf)
                                 end,
                     Descripcion=Case max(P.Descripcion)
                                  When 'SIN PERFIL' then max(U.Descripcion)
                                  else max(P.Descripcion)
                                 end
  from Cita_Admision C, Preasigna R, Perfiles P, Pacientes A, Pruebas U
  where C.Tipo = 'C'    and C.Folio=R.Folio    and C.NumExp=R.NumExp
    and R.NumExp=A.Numexp    and R.Cve_Seccion=P.Cve_Seccion
    and P.Cve_Seccion=U.Cve_Seccion    and R.Cve_Perfil=P.Cve_Perfil
    and P.Cve_Perfil=U.Cve_Perfil    and R.Cve_Prueba=U.Cve_Prueba
    and C.NumExp=@Dato
    group by C.Folio, R.Cve_Seccion
/* End If */
```

Figura 63.
Procedimiento almacenado para la generación del reporte de citas.

En el sistema existen aproximadamente 100 procedimientos, los más recientes cambios en éste aspecto han sido para implantar el manejo de externos, internos y urgencias.

4.3 Construcción de módulos.

En el capítulo anterior se presentó la concepción teórica y esquemática de las pantallas que componen los módulos del sistema, a continuación presentaremos la versión definitiva de dichas pantallas, así como parte del código empleado para su funcionamiento.

Antes que nada, debemos de hacer que *Visual Basic* reconozca los procedimientos almacenados que el sistema utilizará de forma integral para realizar su trabajo, para ello es necesario crear un entorno de datos o "*Data Environment*" (*DE*) dentro de la sección de

diseño (*Designer*) de *Visual Basic*, dentro de este *DE* realizamos la conexión a la base de datos que contiene las tablas y los procedimientos almacenados que utilizaremos, la siguiente figura muestra como está conformado el entorno de datos.

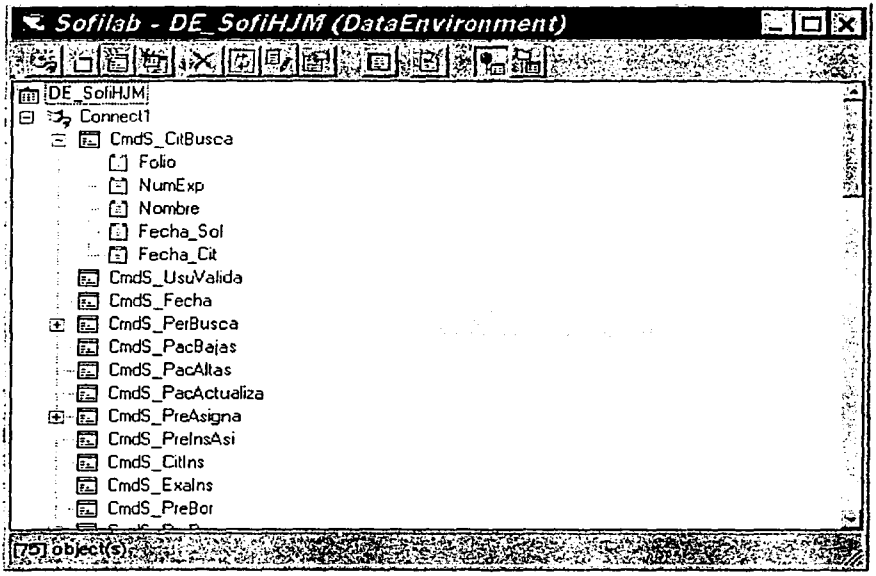


Figura 64.
Entorno de datos (Data Environment).

DE_SofiHJM es el nombre lógico del entorno de datos de la aplicación y a través del *Connect1* se tiene acceso a tablas y procedimientos almacenados como por ejemplo *Cmds_Citbusca*. El cual despliega los campos de Folio, NumExp, Nombre, Fecha_Sol, Fecha_Cit.

4.3.1 Citas.

El módulo de citas es básicamente en donde los usuarios dan de alta a los pacientes en el sistema y programan sus citas, este módulo tiene como pantalla principal el siguiente diseño.

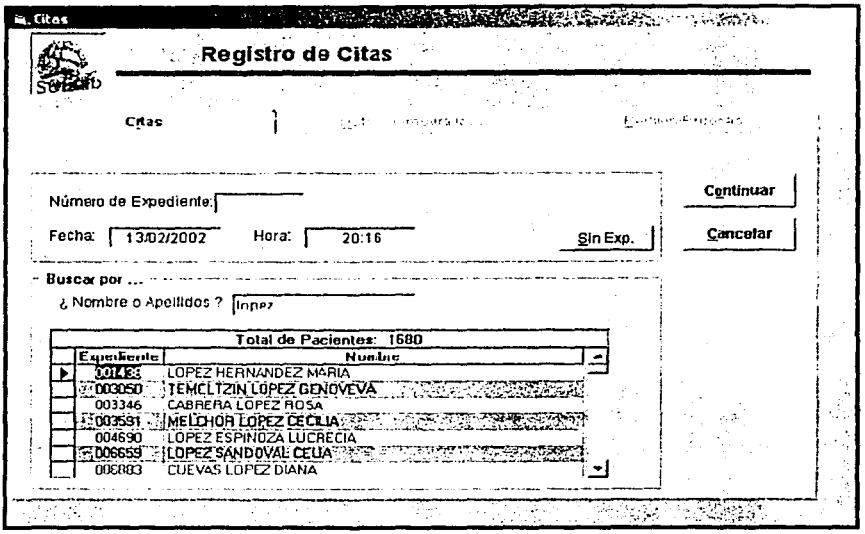


Figura 65.
Pantalla de citas 1.

Esta pantalla consta de tres partes principales, en la primera parte se realiza la búsqueda del paciente para verificar si ha tenido citas previas, esto se hace con el objeto de utilizar el número de expediente del paciente generado con anterioridad y así mismo evitar volver a capturar los datos del paciente.

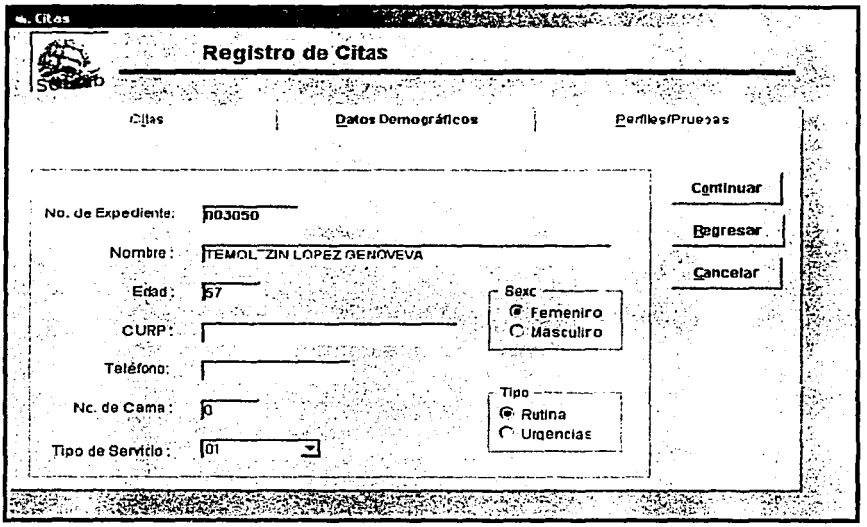


Figura 66.
Pantalla de citas 2.

La segunda parte de la pantalla mostrada en la figura, es donde se capturan los datos generales del paciente, si el paciente ya existe dentro de nuestra base de datos la información es generada de manera automática.

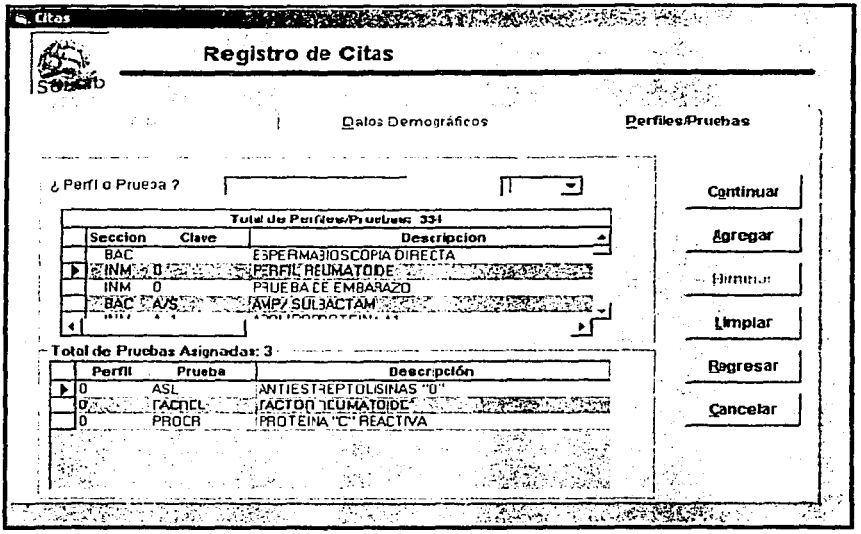


Figura 67.
Pantalla de Citas 3.

Dentro de la tercera parte de la pantalla de citas se asignan las pruebas y los perfiles que le serán practicados al paciente, esta pantalla les da la opción a los usuarios de que en caso de algún error sea posible borrar y volver a realizar el asignado de las pruebas y/o perfiles a los pacientes antes de ser registrados en la base de datos por medio de los procedimientos almacenados correspondientes de la forma.

```
    If KeyAscii = 13 Then
      Screen.MousePointer = vbHourglass
      If DE_SofiHJM.rsCmdS_PerPruBusca.State <> 0 Then
DE_SofiHJM.rsCmdS_PerPruBusca.Close
      DE_SofiHJM.CmdS_PerPruBusca "2", TxtBusPerPru.Text
      GridPerPru.ReOpen
      If DE_SofiHJM.rsCmdS_PerPruBusca.RecordCount = 0 Then
       MsgBox "¡ No Existen Perfiles/Pruebas Bajo ese Criterio ¡", vbInformation, "¡ AVISO ¡"
       GridPerPru.Visible = False
       CmdAgr.Enabled = False
       TxtBusPerPru.SelStart = 0
       TxtBusPerPru.SelLength = Len(TxtBusPerPru)
       TxtBusPerPru.SetFocus
      Else
       GridPerPru.Caption = "Total de Perfiles/Pruebas: " +
    Str(DE_SofiHJM.rsCmdS_PerPruBusca.RecordCount)
       GridPerPru.Visible = True
       CmdAgr.Enabled = True
      End If
      Screen.MousePointer = vbDefault
 End If
```

Figura 68.
Código que realiza la consulta y llena los datos en la tabla.

El fragmento de código presentado en la figura anterior es un ejemplo de cómo se realizan las búsquedas en el sistema, primero verifica que la conexión esté disponible y que el procedimiento almacenado esté cerrado, posteriormente, invoca al procedimiento almacenado de búsqueda y por medio del comando *Grid.ReOpen* realiza una actualización de los datos contenidos en la tabla, dentro de la mayor parte del sistema las búsquedas se realizan de manera similar.

4.3.2 Admisiones.

Dentro de esta pantalla se realiza la consulta de las citas y admisiones que se tienen programadas para una determinada fecha, así mismo es posible realizar una nueva cita, reprogramarla o eliminar el registro de las mismas. Esta pantalla está conformada por controles *TrueBD Grid*, *Sheridan Calendar Widget* y por diversos controles nativos de *Visual Basic* como *TextBox*, *ComboBox*, *OptionBottons*, *CommandBotons*, entre otros.

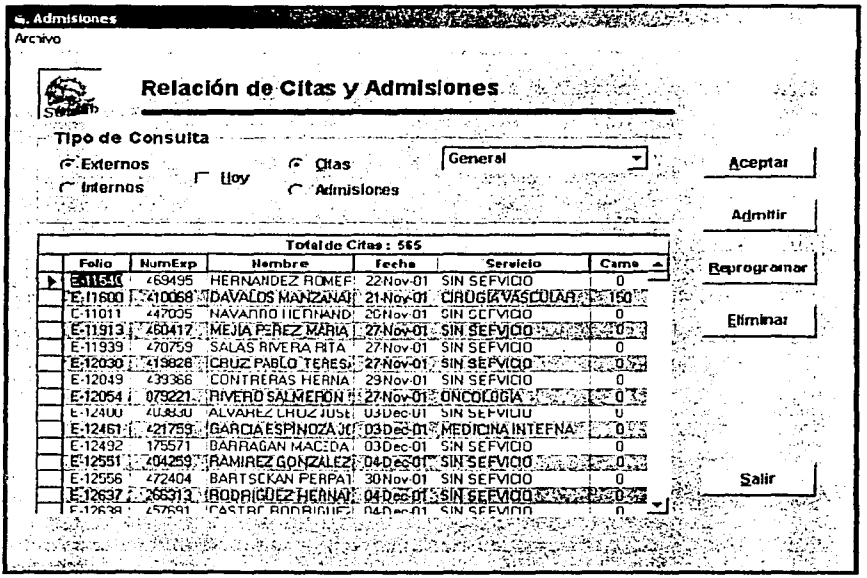


Figura 69.
Forma de Admisiones.

Parte de las funciones de esta pantalla son realizadas por los procedimientos almacenados descritos en la sección anterior, los cuales son agregados al proyecto de *Visual Basic* mediante un entorno de datos, a continuación presentamos una fracción del código en donde se hace la llamada de un procedimiento almacenado.

```
Screen.MousePointer = vbHourglass
  If DE_SofiHJM.rsCmdS_CitBusca.State <> 0 Then DE_SofiHJM.rsCmdS_CitBusca.Close
  If ChkHoy.Value = 1 Then 'Del dia de Hoy
   If OptCitas.Value Then
     GridCitas.Caption = "Total de Citas : "
  Select Case CmbOpciones.Text
       Case "General"
         DE_SofiHJM.CmdS_Citbusca "1", "C", "G", ""
       Case "Folio"
         DE_SofiHJM.CmdS_Citbusca "1", "C", "F", TxtBusca.Text
       Case "No. de Expediente"
         DE_SofiHJM.CmdS_Citbusca "1", "C", "A", Format(TxtBusca.Text, "000000")
       Case "Nombre"
         DE_SofiHJM.CmdS_Citbusca "1", "C", "N", TxtBusca.Text
      End Select
    End If
  End If
GridCitas.ReOpen
```

Figura 70.
Código que realiza la llamada de un procedimiento almacenado en el evento click del botón aceptar.

En este código la instrucción *DE_SofiHJM* hace referencia al entorno de datos del proyecto, el comando *DE_SofiHJM.CmdS_Citbusca* llama al procedimiento almacenado deseado y las instrucciones que se encuentran posteriormente entre comillas dobles son los parámetros necesarios para que el procedimiento almacenado funcione.

Dentro del módulo de admisiones, existen otras pantallas que complementan las funciones a realizar y de hecho facilitan al usuario su manejo, la siguiente pantalla se muestra al agregar una nueva cita o al editar la cita del usuario.

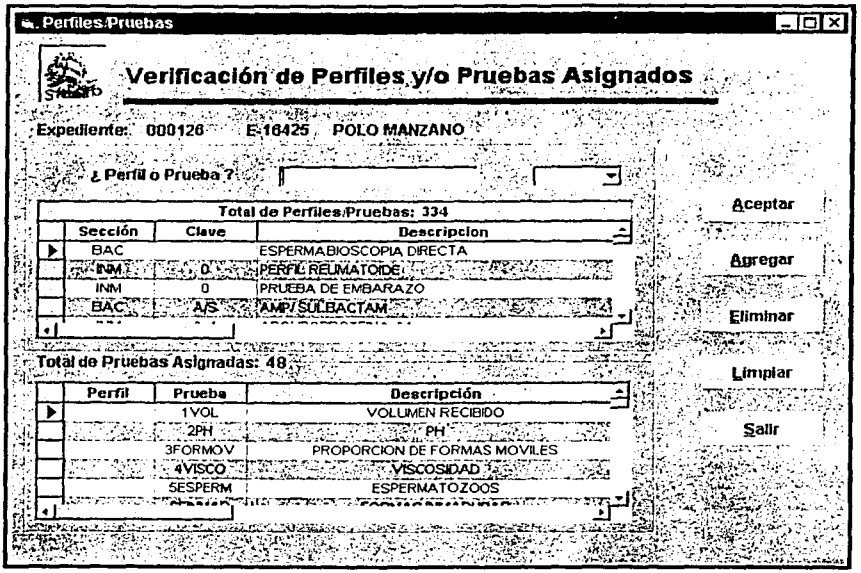


Figura 71.
Pantalla de perfiles y pruebas.

Dentro de esta pantalla los usuarios del laboratorio pueden verificar, asignar, eliminar o limpiar completamente las pruebas o perfiles que le serán practicadas al usuario, de igual forma el código es auxiliado por los procedimientos almacenados del sistema los cuales liberan parte del trabajo del mismo.

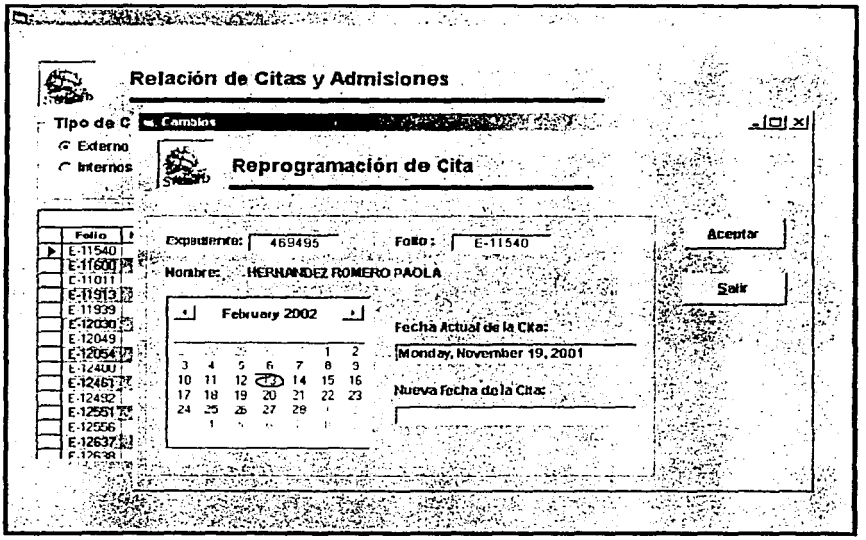


Figura 72.
Pantalla de reprogramación de citas.

Esta pantalla aparece al momento de activar el botón *reprogramar* correspondiente a admisiones, dentro de ella es posible reprogramar la cita asignada originalmente, una herramienta muy útil que facilita a los usuarios el no tener que teclear la fecha es el control *MonthView* de *Visual Basic*, en éste control es posible avanzar y retroceder de manera mensual y seleccionar el dia con un simple clic, al hacer esto la fecha se captura de manera automática en el control *TextBox* correspondiente a la nueva fecha de la cita, al activar el botón aceptar es llamado el procedimiento almacenado que se encarga de actualizar la tabla correspondiente a citas para almacenar el nuevo registro.

4.3.3 Catálogos.

El uso de los catálogos en el sistema es una parte importante ya que a través de ellos se realiza la actualización de los medios con los que cuenta el laboratorio. *El Sistema de Laboratorio Clínico Sofilab* cuenta con diferentes tipos de catálogos entre los que se encuentran Reactivos, Usuarios, Secciones, Servicios, Pruebas, Sistemas, Unidades, etc. Cada catálogo cuenta con información diferente, pero las pantallas con las que se manipula la información son similares en todos ellos.

Dentro de cada catálogo es posible insertar, borrar y editar la información que contiene, debido a que en algunos de ellos como en el caso del catálogo de usuarios la información es por su confidencialidad, de uso restringido a algunos de ellos, dependiendo de su clave y de las diversas medidas de seguridad con que cuenta el sistema y que discutiremos más adelante en éste mismo capítulo.

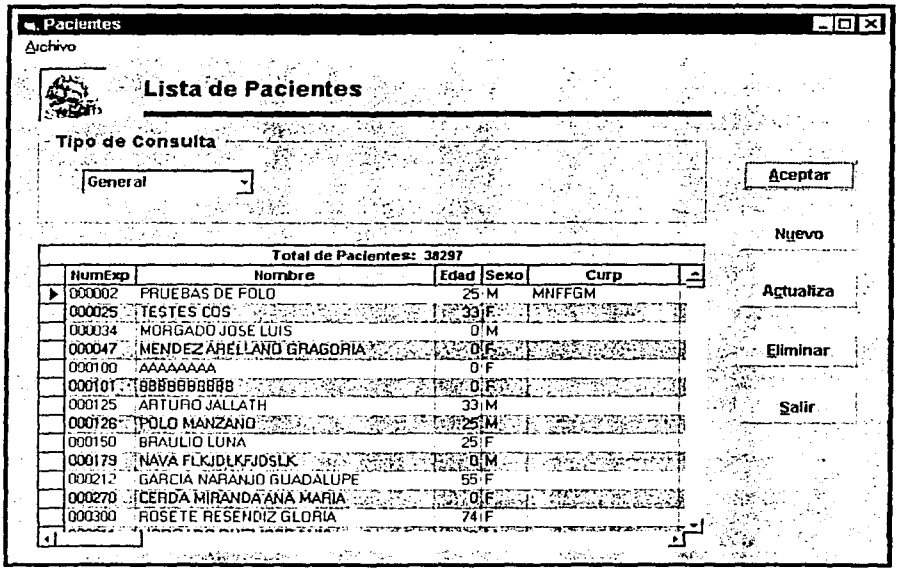


Figura 73.
Catálogo de pacientes.

Las pantallas principales de los catálogos del sistema están conformadas por controles nativos de *Visual Basic* así como de un control *True BDGrid* en el cual se despliega la información, dentro de éstas pantallas es posible realizar diversos tipos de búsquedas, como por ejemplo en el caso de los pacientes se pueden realizar búsquedas por expediente, CURP, nombre del paciente, etc. Esto se diseñó e implementó así para facilitar a los usuarios los procesos de bajas de material o de pacientes, al poder realizar búsquedas precisas su trabajo se facilita de manera significativa.

Las bajas en los registros se realizan inmediatamente después de la confirmación del usuario, las altas y cambios requieren del uso de la siguiente pantalla.

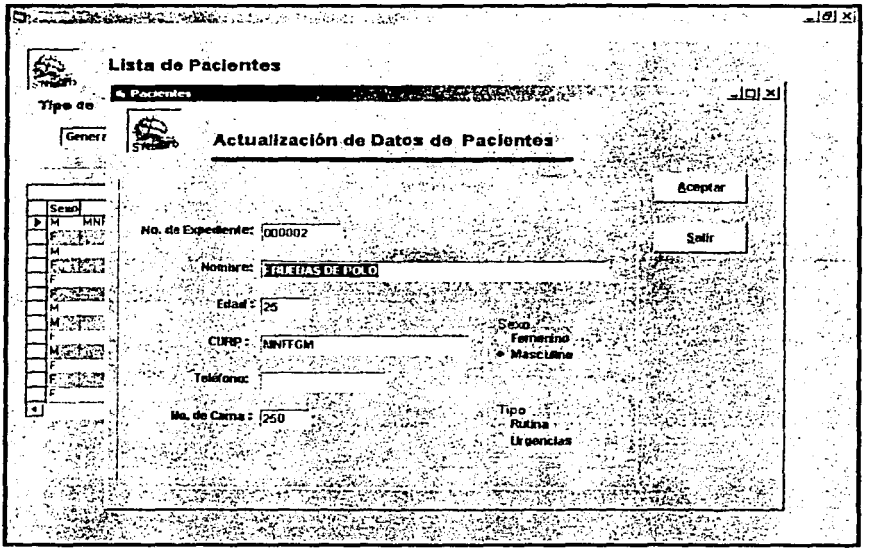


Figura 74.
Pantalla de varios correspondiente a pacientes.

Esta pantalla es la que se presenta en el momento de agregar un nuevo paciente o de editar un registro existente, en el caso de los reactivos y de los otros catálogos aparece un campo que trae el número de registro correspondiente al siguiente en orden si es un registro nuevo o el número del registro seleccionado, en el caso de pacientes nos trae el número de expediente correspondiente, los campos restantes que aparecen varían de acuerdo al catálogo y al tipo de datos. En varias ocasiones es posible utilizar la información de los demás catálogos para complementar la existente, por ejemplo en la figura anterior en el campo correspondiente a "*tipo de servicio*" se utiliza un control *SSOleDBCombo* de *Sheridan Data Widgets* que realiza una consulta y despliega los campos seleccionados del catálogo de servicios, evitando el recurrir a listas para verificar el tipo de servicio.

4.3.4 Interfaces.

El módulo de interfaces es un listado de los diversos aparatos de análisis clínico conectados al sistema, debido a que cada interfaz es un programa más complejo y específico para cada equipo de laboratorio al momento de ser seleccionado en el listado principal se ejecuta el comando *Shell* y llaman al programa correspondiente, en el punto 4.4 de la presente tesis se explica a detalle y ejemplifica el funcionamiento de una interfaz.

4.3.5 Estadísticas y monitoreo.

El módulo de estadísticas y monitoreo, como su nombre lo indica, sirve para que el administrador del sistema pueda tener una visión general o específica del número de accesos al sistema, número de operaciones (consultas, altas, bajas), etc, así mismo es posible observar cuantas citas se realizaron el día, cuantas no llegaron y cuáles secciones del laboratorio estuvieron más ocupadas, la siguiente pantalla muestra la pantalla de monitoreo del sistema.

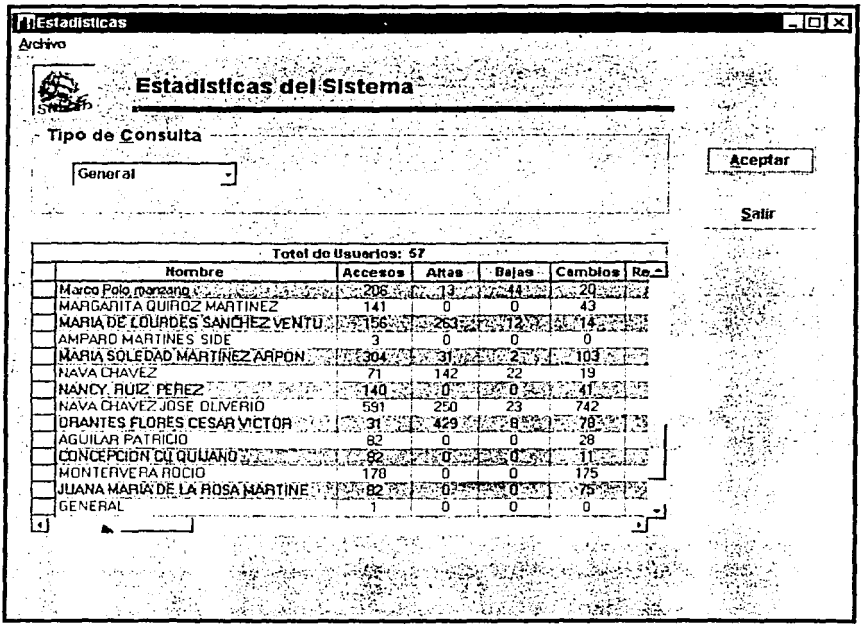


Figura 75.
Pantalla de monitoreo.

4.3.6 Mantenimiento.

El mantenimiento de la base de datos es una tarea muy importante para el adecuado funcionamiento de la base de datos, las tareas de respaldo de la base de datos, la limpieza del *Transaction Log*, la actualización de índices, etc. son vitales y deben de ser realizadas

de manera periódica, debido a que por razones de seguridad no todos los usuarios dentro del sistema pueden realizar estas tareas, se ha diseñado una pantalla a la que solamente los usuarios dados de alta como administradores del sistema pueden acceder y realizar las tareas antes mencionadas.

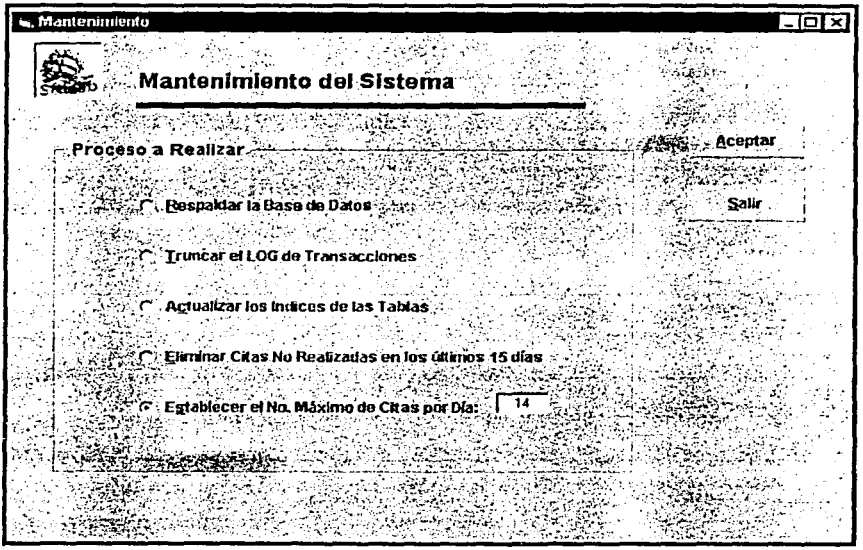


Figura 76.
Pantalla de mantenimiento.

El funcionamiento de esta pantalla es muy similar a las anteriores, al seleccionar una opción se ejecuta el procedimiento almacenado que trunca el log de la base de datos, que realiza el respaldo, etc. Todo esto con el fin de hacerlo de una manera amigable para el usuario y lo más seguro y confiable posible.

4.3.7 Reportes.

Los reportes son un elemento indispensable del sistema, sin duda alguna los usuarios no solamente buscan un sistema en donde sea posible ver que usuarios están dados de alta o la relación de los resultados de laboratorio, también quieren tener de manera impresa dicha información y es por ello que implementamos los reportes en el sistema.

Cada reporte está diseñado con el Software de *Seagate Crystal Reports* versión 8.0, esta herramienta nos permitió utilizar los mismos procedimientos almacenados desarrollados para las consultas del sistema en la construcción de los reportes del mismo, en algunos casos debido a que la información que se requería tener de manera impresa en el reporte era muy especial, se tuvo que desarrollar un procedimiento almacenado especial para la elaboración de dicho reporte, en el punto 4.2 de la presente tesis se mostró un ejemplo de este tipo de procedimientos almacenados.

Dentro de *Visual Basic* la llamada a impresión del reporte se realiza de manera similar a la forma en que se llama la ejecución de los procedimientos almacenados, se pasan los parámetros correspondientes y los modelos ya diseñados dentro de *Crystal Reports* se encargan de desplegarlo, el código siguiente muestra un ejemplo de un llamado de impresión de *Visual Basic* hacia *Crystal Reports*.

```
CRUni.ReportFileName = App.Path + "\RptUnidades.rpt"
CRUni.Connect = "DSN=" & G_Server & ";UID=sa;PWD=;DSQ=SofiHJM"
CRUni.WindowTitle = "Reporte De Unidades"
If DE_SofiHJM.rsCmdS_UniBusca.State <> 0 Then DE_SofiHJM.rsCmdS_UniBusca.Close
Select Case CmbOpciones.Text
 Case "General"
  CRUni.StoredProcParam(0) = "G"
  CRUni.StoredProcParam(1) = "N"
 Case "Clave"
  CRUni.StoredProcParam(0) = "C"
  CRUni.StoredProcParam(1) = TxtBusca
 Case "Descripción"
  CRUni.StoredProcParam(0) = "D"
  CRUni.StoredProcParam(1) = TxtBusca
 End Select
CRUni.PrintReport
```

Figura 77.
Código de llamada a impresión de reporte desde Visual Basic.

Dentro del sistema existen reportes que se pueden generar desde una pantalla en específico, como es el caso de los catálogos, y una pantalla en donde se mandan a imprimir reportes de carácter general como lo son la cantidad de citas que faltan por recibir, las citas recibidas, las no recibidas, las admisiones pendientes entre otras; a continuación se muestra la pantalla que administra la impresión de dichos reportes.

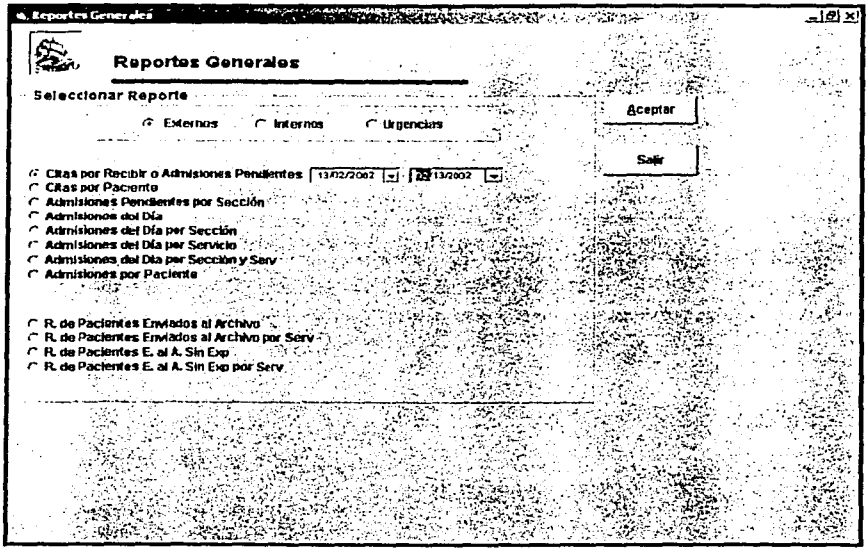


Figura 78.
Pantalla de reportes generales.

La forma en que trabaja *Crystal Reports* resultó amigable, práctica y eficiente; la manera en que se presentan los campos de la base de datos en el reporte nos da la oportunidad de manipular los espacios de la hoja y de agregarle detalles que le proporcionan al reporte una mayor presentación y una apariencia mucho más profesional, la siguiente imagen presenta de manera genérica el formato de página que ofrece *Crystal Reports* para el diseño del reporte.

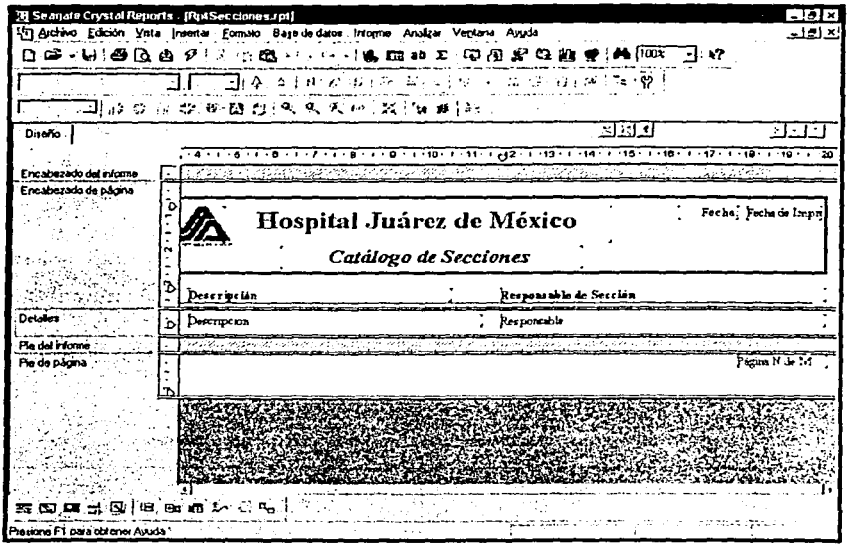


Figura 79.
Pantalla de diseño de Crystal Reports.

Podemos observar en figura, que la pantalla de diseño cuenta con una serie de secciones como la de *pie de página, detalles, encabezados de informe y de página*, etc. Cada sección puede ser organizada de acuerdo al criterio y necesidades del desarrollador, la sección de *detalles* es la sección que contiene los campos de la tabla o procedimiento almacenado que se desea mostrar, debido a que se presentan en una especie de caja de texto es posible moverlos, ajustarlos en tamaño o si no se quiere que algún campo en específico aparezca simplemente borrarlo. La sección de *encabezado* de página es generalmente la presentación de cada hoja, en está sección se pueden agregar imágenes (de hecho en todas las secciones) y diseñar un encabezado acorde con el proyecto, se le puede agregar fecha de impresión, hora de impresión, números de página, texto de manera general, etc.

Una vez ya realizado el diseño del reporte es posible ver una presentación preliminar lo que facilita las correcciones entre espacios y el ajuste de encabezados y de todos los objetos mostrados en el reporte, la siguiente página muestra un ejemplo de los reportes del sistema ya terminado.

HOSPITAL JUAREZ DE MEXICO

Estadísticas De Laboratorio por Perfil y Prueba

Sección:URI

Perfil:CUPROT

	PROTOTO			VOLUMEN			Total
	Externos	Internos	Total	Externos	Internos	Total	
02/17/2001 Pruebas	7	0	7	7	0	7	14
Pruebas Corridas	7	0	7	7	0	7	14
Pacientes	7	0	7	7	0	7	14
04/17/2001 Pruebas	6	0	6	6	0	6	12
Pruebas Corridas	6	0	6	6	0	6	12
Pacientes	6	0	6	6	0	6	12
05/17/2001 Pruebas	2	0	2	2	0	2	4
Pruebas Corridas	2	0	2	2	0	2	4
Pacientes	2	0	2	2	0	2	4
06/17/2001 Pruebas	4	0	4	4	0	4	8
Pruebas Corridas	4	0	4	4	0	4	8
Pacientes	4	0	4	4	0	4	8
07/17/2001 Pruebas	1	0	1	1	0	1	2
Pruebas Corridas	1	0	1	1	0	1	2
Pacientes	1	0	1	1	0	1	2
10/17/2001 Pruebas	7	0	7	6	0	6	13
Pruebas Corridas	7	0	7	6	0	6	13
Pacientes	7	0	7	6	0	6	13
11/17/2001 Pruebas	5	0	5	5	0	5	10
Pruebas Corridas	5	0	5	5	0	5	10
Pacientes	5	0	5	5	0	5	10
12/17/2001 Pruebas	4	0	4	4	0	4	8
Pruebas Corridas	4	0	4	4	0	4	8
Pacientes	4	0	4	4	0	4	8
17/17/2001 Pruebas	1	0	1	1	0	1	2
Pruebas Corridas	1	0	1	1	0	1	2
Pacientes	1	0	1	1	0	1	2
18/17/2001 Pruebas	6	0	6	6	0	6	12
Pruebas Corridas	6	0	6	6	0	6	12
Pacientes	6	0	6	6	0	6	12
19/17/2001 Pruebas	5	0	5	5	0	5	10
Pruebas Corridas	5	0	5	5	0	5	10
Pacientes	5	0	5	5	0	5	10

Figura 80.
Reporte de estadísticas.

4.3.8 Resultados.

Los resultados de los exámenes practicados a los pacientes son ingresados al sistema de manera manual por medio de los encargados del laboratorio y de forma automática por medio de las interfaces conectadas a los equipos de laboratorio, está última forma será explicada en el punto 4.4 del presente documento.

El sistema permite a los usuarios realizar la consulta de los exámenes realizados en determinadas fechas facilitando de esta manera la búsqueda de los resultados de los mismos, la siguiente imagen muestra la pantalla de consulta general de exámenes realizados.

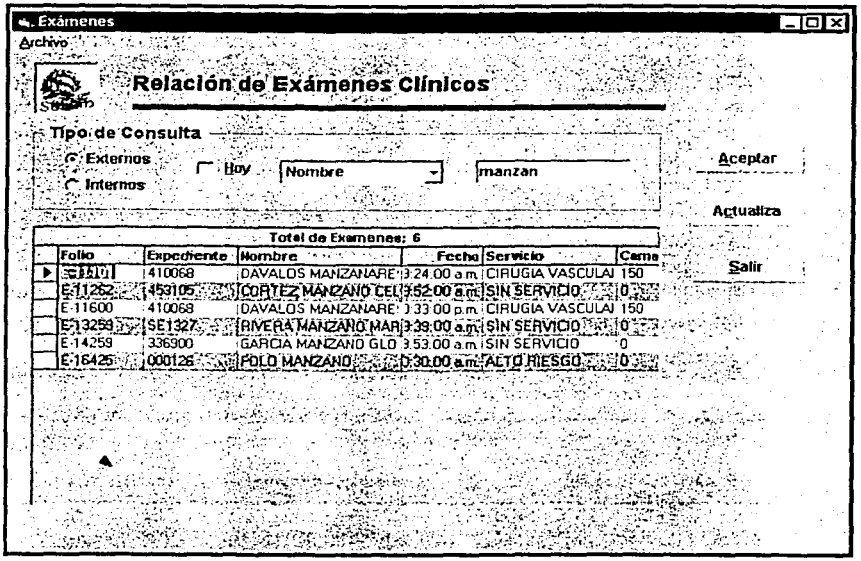


Figura 81.
Pantalla de consulta de exámenes.

Dentro de está pantalla el usuario puede buscar al paciente por medio de su número de folio, de expediente, por nombre o por fecha, además de permitirle al usuario ver los exámenes que se realizaron en la fecha actual o en fechas anteriores. Dentro de la opción Actualiza es posible hacer modificaciones a los resultados de manera manual, al seleccionar esta opción aparece la siguiente pantalla.

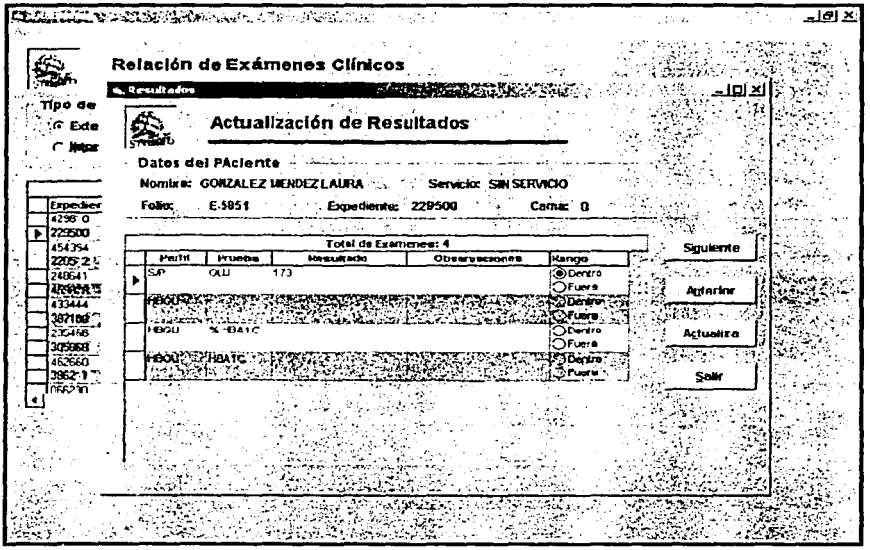


Figura 82.
Pantalla de actualización de resultados.

Los usuarios tienen la posibilidad de navegar sobre la tabla y modificar los datos de los resultados almacenados en la base de datos, así mismo es posible realizar la captura manual de los resultados en ésta misma pantalla, esta opción se muestra al presionar el botón *Nuevo* de la pantalla de consulta de exámenes.

4.4 Interfaces utilizadas en el sistema.

Las interfaces con los equipos de laboratorio conforman una gran parte del sistema, actualmente existen implementadas 7 interfaces en el laboratorio y se pretende desarrollar 3 más.

Lo primero que muestran las interfaces es una pantalla de consultas, en las que el encargado del laboratorio podrá revisar los folios que se encuentran pendientes para procesar, ésta es la pantalla.

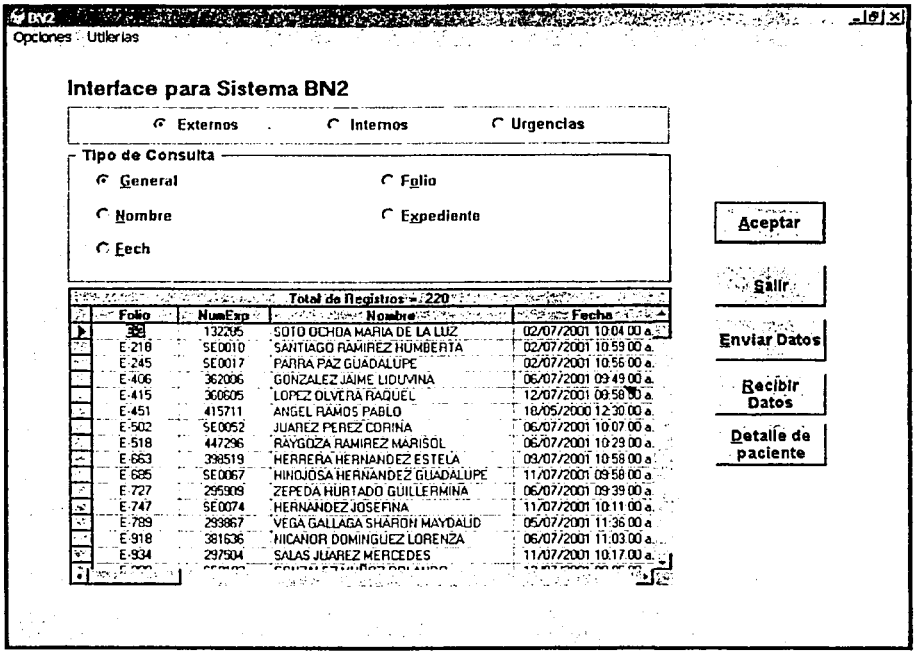


Figura 83.
Pantalla inicial de la interfaz.

La totalidad de los equipos de laboratorio ofrecen la facilidad de cambiar parámetros de comunicación, éstos varían desde la configuración de los puertos hasta la estructura de la trama de comunicaciones. En la interfaz, el menú de Opciones despliega una pantalla donde es posible cambiar las propiedades de comunicación del puerto.

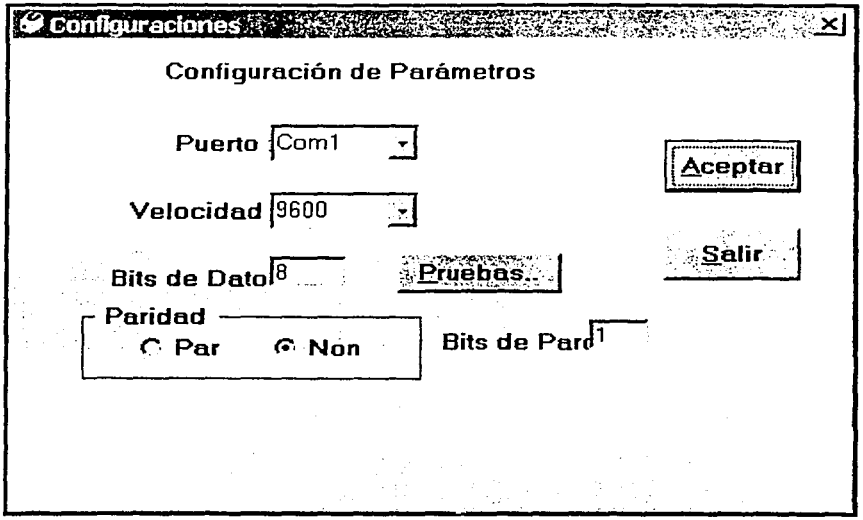


Figura 84.
Pantalla de configuración de puertos.

En esta pantalla se asigna el puerto de comunicaciones, la velocidad de transmisión, la paridad, bits de datos, bits de paro y una opción para relacionar las pruebas que maneja el aparato con las que maneja el sistema.

En el menú de utilerías existe la opción llamada leer archivo "*bin*". El archivo "*bin*" es un archivo que se genera durante la comunicación con el equipo de laboratorio, el propósito de esta opción es poder repetir la actualización de los datos sin necesidad de volver a transmitirlos al equipo de laboratorio, después de seleccionar el archivo "*bin*" la aplicación se encarga de actualizar los datos que provienen de dicho archivo.

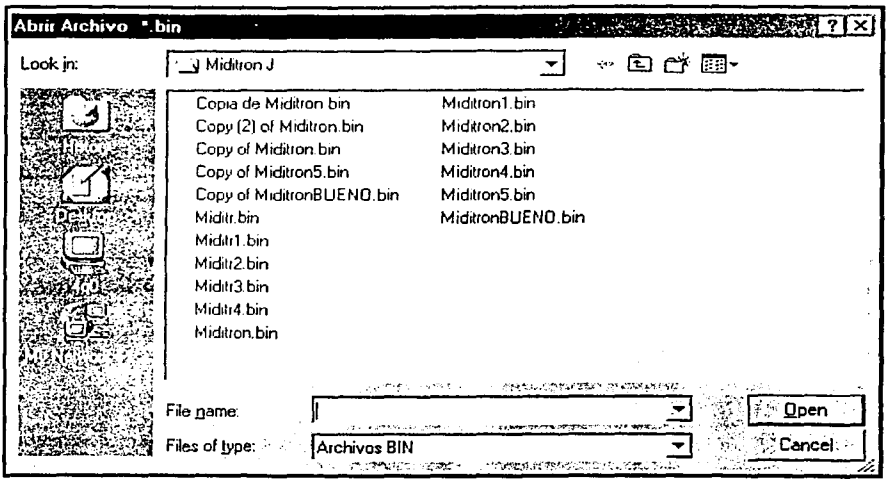


Figura 85.
Pantalla de selección de archivos "bin".

Si presiona el botón de enviar datos, la interfaz transmite los datos que se muestran en la tabla. Para ello es necesario establecer las peticiones correspondientes de comunicación y crear las *tramas* de envío a partir de la información dentro de la tabla de consulta.

Para el caso de la recepción de datos, se establece la comunicación y se comienza a recibir los resultados al mismo tiempo que se genera el archivo con extensión "*bin*", el cual es traducido y transmitido a la base de datos. A continuación describiremos un poco estos procesos así como la estructura de la comunicación entre la interfaz y el equipo de laboratorio conocidos normalmente como *Host* y *BNII* o Unidad Central (*CU*) respectivamente.

Antes de comenzar con la explicación del protocolo de comunicación es necesario aclarar los códigos *ASCII* especiales utilizados en las comunicaciones, estos suelen variar en cuanto a nomenclaturas y valores para cada equipo de laboratorio pero los siguientes son los más utilizados y los particulares para el equipo seleccionado como ejemplo.

Nombre	ASCII	HEX	DEC	Descripción
acknowledge	ACK	6	6	Reconocimiento Positivo
No acknowledge	NAK	15	21	Reconocimiento Negativo
Start of Text	STX	2	2	Principio de Texto o Trama

End of Text	ETX	3	3	Final de Texto o Trama
Carriage Return	CR	0D	13	Fin de linea
Block Check Character	BCC	*	*	Bloque de control

El *ACK* es la respuesta lógica cuando la trama es perfectamente comprendida por el receptor, de la misma manera el *NAK* indica que no se recibió adecuadamente el texto transmitido, *STX* y *ETX* denotan el principio y fin de una trama, El *CR* o "*Carriage Return*" (retomo de acarreo) es un separador de texto con el fin de tener linea por linea la información, el *BCC* es utilizado para verificar que los datos recibidos son los mismos que los transmitidos.

El protocolo de comunicación de esta interfaz tiene la siguiente estructura básica.

STX	Record identifier	Data Fields	BCC	CR	ETX
	< BCC	>			

El "*Record identifier*" es un caracter que determina el tipo de trama y el campo "*Data Fields*" contiene los datos particulares de la trama.

Como podemos observar El *BCC* es un caracter o caracteres que se calcula con los datos de los campos "*Record Identifier*" y "*Data Fields*". La forma de calcular el *BCC* varía en cada protocolo, para éste caso se suman los valores hexadecimales de cada caracter y se convierte en un caracter.

Para este ejemplo del equipo *Behring Nephelometer II (BN2)* es posible cambiar algunas opciones de la trama de comunicación, las cuales se describen a continuación.

- *STX ETX: Off/ON* es posible deshabilitar o habilitar el envío de este caracter.
- *ACK: A* establecer el caracter que representa un recepción aceptada
- NACK: E establecer el caracter que representa un recepción no aceptada.
- *End of Line*: *CR* establecer el caracter que representa un retorno de acarreo.
- *Checksum*: *Off/ON* es posible deshabilitar o habilitar el envío de este caracter.

Ahora se mostrará un ejemplo que detalla la estructura de cada trama en una petición de pruebas.

1. HOST => BNII

ASCII	STX	J	G	A	N	Z	B	A	F	F	1								
HEX	02	4A	47	41	4E	5A	42	41	46	46	31	20	20	20	20	20	20	20	20
posición	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19

ASCII														0	7	7	M	0	
HEX	20	20	20	20	20	20	20	20	20	20	20	20	20	30	37	37	4D	30	20
posición	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38

ASCII		0	1	0	2	0	3		0		0		0		0		0		0
HEX	20	30	31	30	32	30	33	20	30	20	30	20	30	20	30	20	30	20	30
posición	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57

ASCII		0		0		0		0		0		0		0		0		0	
HEX	20	30	20	30	20	30	20	30	20	30	20	30	20	30	20	30	20	30	20
posición	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76

ASCII	0		0	%	CR	ETX
HEX	30	20	30	35	0D	03
posición	77	78	79	80	81	82

N. Campo	Longitud	Posición	Contenido	Tipo	Formato	Comentarios
1	1	1	STX	Char		Inicio de Texto
2	1	2	J	Char		Tipo de trama
3	29	3-31	identificador de prueba	Char	Justificación izquierda	
4	1	32	Espacio	Char		
5	3	33-35	Edad	Int	J Derecha "047","_47"	
6	1	36	Sexo	Char		m.M= Masculino f.F=Femenino
7	1	37	Tipo de prueba	Char		0 = prueba normal
8	2	38-39	Observación	Int	J Derecha "01","_1"	00= sin comentarios
9	2	40-41	Prueba 01	Int	J Derecha "01","_1"	Clave de Prueba a realizar
10	2	42-43	Prueba 02	Int	J Derecha "01","_1"	"
11	2	44-45	Prueba 03	Int	J Derecha "01","_1"	"
12-27	2	46-77	Prueba 04 - Prueba 19	Int	J Derecha "01","_1"	"
28	2	78-79	Prueba 20	Int	J Derecha "01","_1"	"
29	1	80	BCC	Char		Block Check Character
30	1	81	CR	Char		Carriage Return
31	1	82	ETX	Char		Fin de Texto

2. BNII => HOST

ASCII	STX	A	?	CR	ETX
HEX	30	41	3F	0D	03
posición	1	2	3	4	5

N. Campo	Longitud	Posición	Contenido	Tipo	Formato	Comentarios
1	1	1	STX	Char		Inicio de Texto
2	1	2	A	Char		Tipo de trama
3	1	3	BCC	Char		Block Check Character
4	1	4	CR	Char		Carriage Return
5	1	5	ETX	Char	J Derecha "047","_47"	Fin de Texto

Como se observa en las tablas, cada trama se debe de armar de acuerdo con las especificaciones establecidas por el fabricante creador del protocolo de comunicación. La siguiente secuencia ejemplifica una transmisión donde la primera trama recibido no es válido.

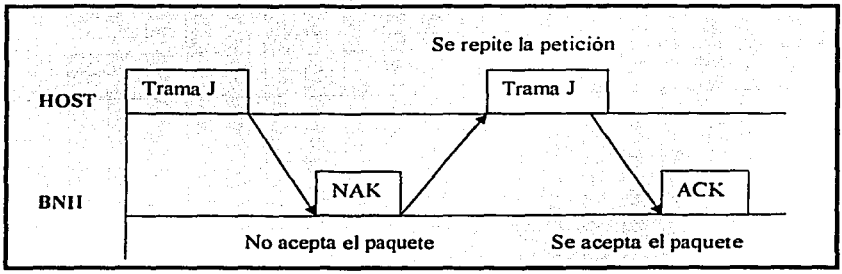


Figura 86.
Diagrama de transmisión de una trama no recibida.

Cuando la trama transmitida no es comprendida por el equipo receptor ya sea *HOST o BNII*, es posible reintentar el envío hasta que esta sea comprendida por el equipo.

De forma similar la recepción de datos requiere una confirmación de cada trama recibida y en este caso es necesario transmitir primero la trama D, especificada por el fabricante, el cual es entendido como petición de resultados y se obedece la siguiente secuencia.

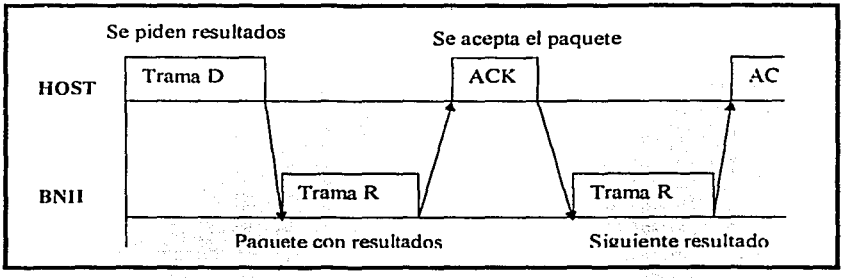


Figura 87.
Petición de resultados.

Cada trama R transmitido por el equipo contiene un resultado para una prueba especifica y para un paciente especifico. El siguiente ejemplo muestra la estructura de la recepción.

1. HOST => BNII

ASCII	STX	D	<	CR	ETX
HEX	30	41	3F	0D	03
posición	1	2	3	4	5

2. BNII => HOST

ASCII	STX	R	G	A	N	Z	B	A	F	F	I								
HEX	02	52	30	38	33	39	20	20	31	20	20	20	20	30	20	20	20	20	20
posición	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19

ASCII												1							
HEX	20	31	E2	34	37	30	30	45	2B	30	31	20	67	2F	6C	20	20	20	20
posición	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38

ASCII		2							1	.	1	4	0	0	E	+	0	1	1
HEX	02	20	20	20	20	20	20	20	31	2E	31	34	30	30	2F	2B	30	31	31
posición	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57

ASCII	G	/	|								#	CR	ETX
HEX	67	20	20	20							53	0D	03
posición	58	59	60	61	62	63	64	65	66	67	68	69	

3. HOST =>BNII

ASCII	STX	A	?	CR	ETX
HEX	30	41	3F	0D	03
posición	1	2	3	4	5

Cada trama de comunicación es guardada en un archivo, de esta manera es posible detectar en donde existen problemas y corregirlos. La traducción de este archivo varía en cada interfaz debido al protocolo de comunicación pero básicamente se realiza un barrido línea a línea del archivo y de acuerdo con lo mostrado en las tablas se extrae la información necesaria para actualizar el resultado en la base de datos.

Al igual que el sistema integral del laboratorio las interfaces utilizan procedimientos almacenados para todas sus actividades con la base de datos, los procedimientos almacenados *Sp_InterfBusca* y *Sp_InterfBuscaDist* son utilizados para generar la lista de trabajo de la sección (consulta de exámenes pendientes por sección, sistema y estatus) , *Sp_InterfRes* actualiza las pruebas provenientes del equipo de laboratorio.

4.5 Seguridad del sistema.

A continuación analizaremos los alcances del sistema en lo que a seguridad se refiere, para facilitar su comprensión se dividieron en tres puntos principales: sistema que describe la seguridad dentro de la aplicación en sí, base de datos referente a la protección que manejamos con el administrador de base de datos, y las restricciones a través del servidor de esta aplicación.

4.5.1 Sistema.

El Sistema Integral de Laboratorio Clínico Sofilab es el único medio de acceso a los datos del laboratorio en lo que a usuarios se refiere, es entonces necesario establecer el alcance que estos pueden tener, para ello es común utilizar claves y contraseñas de acceso, las cuales restringen las operaciones permitidas. Antes de asignar claves se deben establecer los tipos de usuarios.

El lenguaje de programación utilizado ofrece un tipo de protección en lo que a transferencia de datos se refiere, esta protección está basada en la encriptación de los datos transferidos y recibidos por el cliente.

Los puntos anteriores son detallados a continuación.

4.5.1.1 Criterios para niveles.

Para poder establecer los tipos de usuarios que soportaría nuestro sistema se solicitó la colaboración del personal administrativo del laboratorio, así como de la intervención de personal de la empresa *Sofilab*, de aquí se estableció la necesidad de separar principalmente a los usuarios del laboratorio con los usuarios del área de admisiones.

4.5.1.2 Niveles y claves.

Se estableció un máximo de seis caracteres para el uso de claves y contraseñas. Como en la gran mayoría de los sistemas la clave y la contraseña son solicitadas al inicio de la aplicación y a partir de ella se habilitan o se ocultan opciones del sistema.

Los siguientes son los tipos de usuarios que existen y las opciones a las que pueden acceder.

Tipo de Usuario	Iniciales	Opciones disponibles
Administrador	ADM	Todas las opciones
Monitoreo / Consulta Ejecutiva	MON	Citas, Admisiones, Pacientes, Reportes Generales, Resultados, Reportes Estadísticos, Monitoreo, Estadísticas de Monitoreo, Mantenimiento
Cita / Admisión	CYA	Citas, Admisiones, Pacientes, Reportes Generales y de Resultados

Pacientes	PAC	Altas de pacientes internos y Externos
Laboratorio	LAB	Interfaces, Registro de Resultados, Reportes de Resultados
Urgencias	URG	Altas de pacientes de urgencias, Reportes de Resultados.

4.5.2 Seguridad en la base de datos.

SQL Server impone la comprobación de los tipos de datos y también impone las reglas, disparadores y estructuras de índices. Los datos entrantes deben ajustarse a un complicado conjunto de criterios basados en los permisos concedidos al usuario o al grupo al que pertenece el usuario. Los datos podrán aceptarse o rechazarse en función de las incoherencias de la base de datos, la violación de reglas o disparadores, las comprobaciones de duplicidad de índices u otro tipo de problemas, como *SQL Server* se ejecuta dentro de un *Servidor Windows NT*, que tiene su propia idea de quien puede acceder al servidor, un determinado usuario podrá o no ver la consola de administración de *SQL Server*.

Adicionado a lo mencionado anteriormente, el sistema en general realiza la conexión al *Servidor de SQL* mediante el nombre de un usuario *"x"* con su respectiva clave válida y una contraseña, este usuario tiene únicamente permisos de ejecución de procedimientos almacenados y de poder operar sobre algunas tablas, si algún curioso logra desencriptar la cadena de conexión enviada por el sistema hacia el servidor no podrá realizar cambios significativos en la base de datos ya que el usuario no cuenta con tales permisos.

4.5.3 Servidor.

En cualquier implementación cliente/servidor, el servidor no es únicamente un terreno en el que se viertan los datos. También es responsable de la administración inteligente de los recursos, de la administración de la seguridad, de la administración de los datos etc. En esta sección en particular nos enfocaremos a las tareas que el servidor realiza para administrar la seguridad.

El servidor impide el acceso no autorizado a él mismo y a la base de datos, a la vez que permite un acceso protegido a los que tienen permiso válido de acceso. Esta tarea implica salvaguardar no sólo los propios datos, sino también las vistas, procedimientos y la administración de la base de datos, parte de esta tarea se deja a cargo del servidor de *Windows NT* y del control de dominios de *Windows NT*.

Toda transacción de negocios requiere que cada parte este segura de la identidad de la otra. La firma en el caso de una tarjeta bancaria, por ejemplo, es una forma de validación ya que es teóricamente imposible que otra persona haga la misma firma que uno.

Hasta hace relativamente poco tiempo la seguridad no era un tema considerado ya que las redes estaban aisladas del exterior y todo usuario que accesara a ellas debía estar físicamente en el mismo edificio. Así la autentificación de usuarios basados en su dirección IP y/o dominio eran más que suficiente.

Sin embargo, en la década de los 90, la red se hizo cada vez más virtual en esencia y la conexión con el mundo exterior, vía Internet o satélite, se hizo más la regla que la excepción. Asi mismo el par nombre/clave suele ser útil para identificar al usuario pero no al servidor.

Para resolver esta omisión, se creó el par clave pública/clave privada. Estas claves están relacionadas entre si, pero no hay manera de que se pueda extraer información de una a partir de la otra. Como su nombre lo indica la clave pública es la que el usuario envía a todos los destinatarios. Utilizándola, el que envia un mensaje puede encriptar la información. La única forma de desencriptarla es mediante la clave privada, que sólo se encuentra en poder del usuario. Las herramientas existentes para generar este tipo de claves son validadas por ciertas autoridades de certificación, esto es, una cierta empresa garantiza que una clave es auténtica mediante un certificado electrónico, *Windows NT* cuenta el "*Snap-In certificados*", que es en si una lista de las autoridades en las que confia Microsoft para la emisión de sus claves, éstos a su vez ayudan a la creación de diversos niveles de acceso, desde ser un administrador con acceso total al sistema hasta tener únicamente permisos de consulta.

En nuestro caso particular se cuenta con una clave de administrador que nos garantiza el acceso pleno a todos los servicios de *Windows NT*, para el caso de los usuarios solamente algunos de ellos tienen garantizado el acceso al servidor con sus respectivas restricciones y permisos, además de que la red es local.

CAPÍTULO V.
CONCLUSIONES.

5.1 Instalación.

Durante la fase de instalación el primer paso fue la implantación del hardware necesario para la puesta en marcha del sistema, recordemos que ya existía un sistema previo el cual ya contaba con una red local y estaciones de trabajo en las áreas del laboratorio, por lo que solamente fue necesario complementar la red con nuevos nodos y la instalación del servidor del sistema, este proceso tuvo una duración aproximada de 3 días, ligeramente inferior al tiempo estimado en la planeación de tiempos.

En lo referente a las interfaces, recordemos que ya existían tres de estas funcionando para el sistema anterior dentro de las áreas principales de laboratorio, las cuales son hematología *(Technicon H3)*, urianálisis *(Clinetec 500)* e inmunología *(Axym)*. Las interfaces correspondientes a hematología e inmunología fueron rediseñadas para el nuevo sistema y en cuanto al equipo de urianálisis se cambio por el *Miditron M*, por lo que su interfaz se construyó de manera completa, debemos mencionar que estas áreas del laboratorio son las que tienen una carga de trabajo más considerable con un promedio de 70 a 80 muestras por día sólo para los pacientes externos, por lo que la prioridad de desarrollo de estas interfaces fue mayor, en consecuencia las otras secciones de laboratorio quedaron pendientes para su automatización para los meses subsecuentes.

El siguiente paso fue la instalación del software requerido (base de datos y utilerías) y la importación de datos del sistema anterior al nuevo, cabe destacar que la duración de este último punto fue de más de dos días debido a que se requirió depurar la información, así como realizar varias revisiones de los datos para asegurar la integridad y legitimidad de los mismos.

Para facilitar la liberación del sistema en las terminales de laboratorio se generó previamente un programa de instalación el cual requirió ser reconstruido después de realizar los primeros intentos, debido principalmente a que algunas librerías pertenecientes a las herramientas de diseño del sistema no se agregaban debidamente en el proceso, por lo que fue necesario modificar el mismo agregando las librerías correspondientes, esto en consecuencia incremento el tiempo previsto para la instalación del sistema de 1 a 2 días.

Al igual que el sistema principal, todas las interfaces cuentan con un programa de instalación, el cual es relativamente sencillo, solo se debe tener especial cuidado en la configuración de los parámetros de comunicación y la asignación de pruebas, además de realizarse la capacitación del personal del laboratorio.

Posteriormente, se procedió a realizar las pruebas de funcionamiento del sistema, las cuales consistieron en una revisión completa de los módulos principales del mismo, a pesar de que en general funcionaban adecuadamente, se encontraron errores en el programa concernientes a detalles no considerados en cuanto a programación (*bugs*) y que surgen

durante su uso frecuente, esto no impidió la puesta en marcha del sistema, pero si se alargó el proceso ya que algunos de estos detalles eran recurrentes por la forma de manejo de los usuarios, por lo que tuvieron que ser corregidos de forma inmediata.

Por último, se requirió de la capacitación del personal de laboratorio en cuanto al uso y manejo del sistema, éste proceso se realizó de manera paralela a la puesta en marcha de éste proyecto, lo cual requirió de un constante monitoreo de nuestra parte, auxiliados por personal tanto del laboratorio como de la empresa *Sofilab*, aunado a esto fue realizado un manual de usuario que permite facilitar el proceso de capacitación de los nuevos usuarios.

5.2 Mantenimiento.

La fase de mantenimiento se centra en los cambios realizados en un sistema que van asociados a la corrección de errores a medida que evoluciona el entorno del software, y a los cambios derivados de las mejoras producidas por los requisitos variables del cliente. La fase de mantenimiento vuelve a aplicar los pasos de las fases de diseño y desarrollo, pero con el contexto del software ya existente.

Durante la fase de mantenimiento se encuentran cuatro tipos de cambios de los cuales se hablará a continuación.

Mantenimiento correctivo.

Incluso llevando a cabo las actividades de garantía de calidad y apegándose a las especificaciones de diseño, es muy probable que el usuario descubra defectos en el software. El mantenimiento correctivo tiene por función modificar la programación a fin de corregir los defectos que se presenten.

Mantenimiento preventivo.

El software de computadora se deteriora debido al cambio de su entorno, y por ello dicho mantenimiento también es llamado "reingeniería del software", se debe de orientar a la solución de los usuarios finales. En esencia, el mantenimiento preventivo realiza cambios en las aplicaciones a fin de que se pueda corregir, adaptar y mejorar el sistema en beneficio del usuario final.

En el caso del *Sistema Integral de Laboratorio Clínico Sofilab* este tipo de mantenimiento se presentó al realizar las modificaciones correspondientes en la base de datos, con el fin de no afectar el rendimiento general de la aplicación a medida que aumentaba la cantidad de información que maneja el sistema.

Mantenimiento adaptativo.

Al paso del tiempo, es posible que cambie el entorno para el que originalmente se desarrolló el software (la versión del sistema operativo, las políticas de la empresa, etc.) El

mantenimiento adaptativo produce modificaciones en el software a fin de ajustarlo a estos cambios de entorno.

Los cambios realizados al sistema solicitados por la empresa *Sofilab* para adaptar el sistema al Hospital Mérida es un claro ejemplo de este mantenimiento.

Mantenimiento perfectivo.

A medida que el software es utilizado por el usuario, este puede descubrir funciones adicionales que podrían ser de gran beneficio. El mantenimiento perfectivo lleva a las aplicaciones mas allá de sus requisitos funcionales originales.

Una ventaja que representa la arquitectura cliente/servidor es el hecho de que cuando cambian las reglas del negocio, solamente se tiene que modificar la programación realizada sobre los procedimientos almacenados ubicados en el servidor de la base de datos de tal manera que los cambios se aplican inmediatamente a todos los usuarios del sistema.

Aquí podemos destacar la inclusión del manejo de pacientes externos y de urgencias lo que derivó en una modificación significativa en el código del sistema para poder realizar dicha tarea.

5.3 Conclusiones generales.

Las tendencias de automatización y optimación de recursos en los laboratorios clínicos de diagnostico han hecho indispensable el uso de herramientas de informática para el manejo y procesamiento de datos, como consecuencia de ello la presente tesis representa una posible solución a las necesidades actuales de los laboratorios.

La implementación del *Sistema Integral de Laboratorio Clínico Sofilab* representa un avance significativo en comparación con el sistema anterior, solucionando problemas como son seguridad, velocidad, disponibilidad y replicas de información, los cuales fueron eliminado gracias a la reestructuración en el procesamiento de datos (actualización y depuración de los mismos), la disponibilidad y actualización de los datos es ahora inmediata gracias a que la base de datos se encuentra concentrada en un solo equipo (arquitectura cliente/servidor), evitando la espera de actualización de los datos al pasar una replica de un equipo a otro y manteniendo la integridad de los mismos. Como consecuencia de estos cambios estructurales y de la implementación de una tecnologia más avanzada se logró un incremento mayor al 100% en la velocidad del sistema, mayor seguridad y el manejo eficiente de grandes volúmenes de información.

El nuevo sistema ofrece al laboratorio un mejor control y seguimiento de las citas realizadas. La impresión directa desde el sistema redujo significativamente la impresión de múltiples reportes que se realizaban en las secciones del laboratorio, dichas impresiones se adecuaron a las necesidades del área administrativa, mostrando los datos generales del sistema de forma concisa , clara y con una presentación de calidad.

La implementación de interfaces en las diversas áreas del laboratorio, permitió una automatización completa, reduciendo la captura de resultados de manera manual de forma significativa eliminando de esta manera casi el total de los errores de captura y aumentando la velocidad de obtención de los mismos.

El Sistema Integral de Laboratorio Clínico Sofilab, ha representado un gran paso para una mejor organización y administración del laboratorio clínico del *Hospital Juárez de México*, dando como resultado un mejor servicio a los pacientes del mismo.

Gracias al incremento en la eficiencia que proporcionan los módulos de citas y admisiones los tiempos de espera de los pacientes se han visto reducidos significativamente, de igual forma, la rápida obtención de los resultados del laboratorio han permitido al laboratorio una entrega eficiente y oportuna de los mismos a los pacientes y los médicos según sea el caso.

La presentación, uso y manejo amigable del sistema facilita a los usuarios la realización de sus actividades cotidianas, esto además hace posible que el período de capacitación de nuevos usuarios sea considerablemente corto y con la posibilidad de ofrecer el sistema a otros laboratorios públicos y privados.

5.4 Comentarios finales.

El presente sistema fue posible gracias a la aplicación de los conocimientos adquiridos durante la carrera de Ingeniería en Computación, principalmente en las asignaturas de Redes, Sistemas Operativos, Comunicaciones Digitales, Ingeniería de Programación, Base de Datos entre otras y a las aportaciones del personal calificado en los procesos del laboratorio clínico. La puesta en marcha del sistema enriqueció nuestra experiencia profesional, como todo sistema, se encuentra en constante monitoreo y mantenimiento con el fin de perfeccionar su desempeño, debido al uso cotidiano del sistema, han surgido y se continúan realizado modificaciones al mismo, en este sentido el mantenimiento de tipo perfectivo ha sido el mas solicitado actualmente.

La colaboración en el desarrollo de un proyecto de tales características amplió nuestra visión de las posibilidades existentes en el campo de desarrollo de software, no solo nos dio una muestra de lo que significa desarrollar aplicaciones de manera profesional, sino que además pudimos comprender que es necesario un buen planteamiento inicial y la selección adecuada de las herramientas de desarrollo que serán empleadas, así mismo entendimos que debido al avance tecnológico tan acelerado en el que nos desarrollamos debemos enriquecer constantemente nuestro conocimiento sobre las nuevas herramientas de desarrollo, esto con el fin de que las aplicaciones que se desarrollen en un futuro se encuentren siempre a la vanguardia de la tecnología, lo cual se traduce directamente en beneficio para los usuarios.

BIBLIOGRAFÍA.

Visual Basic 6 Manual completo de programación, Aitken Peter, Editorial PARANINFO.

Programación de SQL Server 7.0 con Visual Basic 6.0, Vaughn William R., Editorial McGraw Hill.

Ingeniería del Software, Pressman Roger, Cuarta edición, Editorial McGraw Hill.

Apuntes de Bases de Datos. Semestre 97-2, Facultad de Ingeniería UNAM.

Interfacing Sensors to the IBM PC, Willis J. Tompkins, Prentice Hall.

BNII Data Interface Description, CO. DADE Behring Marburg GMBH

Microsoft Windows 2000, Goldberger Ricardo, Editorial MP.

Introducción a los sistemas operativos, Deitel Harvey, Editorial Addison – Wesley Iberoamericana.

Redes de computadores, protocolos, normas e interfaces, Black Uyless, Editorial Addison – Wesley Iberoamericana.

Tesis: Sistema de información de la administración universitaria, Jallath Arturo, 1999.

Diccionario Microsoft de informática e internet, Editorial McGarw Hill.

Páginas Web:

http://www.inei.gob.pe/cpi/bancopub/libfree/lib616/index.htm
http://www.freevbcode.com
http://www.allaboutnetwrks.net
http://www.map.es/csi/silice/Global71.html
http://sistemas.dgsca.unam.mx/tecnologia.htm