DESARROLLO E IMPLEMENTACIÓN DE UN SISTEMA DE INFORMACIÓN PARA EL LABORATORIO DE QUÍMICA Y MICROBIOLOGIA DEL CENTRO DE INVESTIGACIONES OCEANOGRÁFICAS E HIDROGRÁFICAS

MARIO IGNACIO DE LEON CÉSPEDES

JOSE LUIS PALLARES VARELA

UNIVERSIDAD TECNOLÓGICA DE BOLÍVAR INSTITUCIÓN UNIVERSITARIA

FACULTAD DE INGENIERÍA DE SISTEMAS

CARTAGENA

2002

DESARROLLO E IMPLEMENTACIÓN DE UN SISTEMA DE INFORMACIÓN PARA EL LABORATORIO DE QUÍMICA Y MICROBIOLOGIA DEL CENTRO DE INVESTIGACIONES OCEANOGRÁFICAS E HIDROGRÁFICAS

MARIO IGNACIO DE LEON CÉSPEDES

JOSE LUIS PALLARES VARELA

Proyecto de grado presentado como requisito para optar al título de ingeniero de sistema

Director
LUZ STELLA ROBLES
Ingeniera de Sistemas

UNIVERSIDAD TECNOLÓGICA DE BOLÍVAR INSTITUCIÓN UNIVERSITARIA

FACULTAD DE INGENIERÍA DE SISTEMAS

CARTAGENA

2002

Cartagena de Indias, 29 de abril del 2002

Señores
COMITÉ PROYECTO DE GRADO
Facultad de Ingeniería de Sistemas
La ciudad

Distinguidos señores.

Por medio de la presente me permito informales que el proyecto de grado titulado **"DESARROLLO E IMPLEMENTACIÓN DE UN SISTEMA DE INFORMACIÓN PARA EL LABORATORIO DE QUÍMICA Y MICROBIOLOGÍA DEL CENTRO DE INVESTIGACIONES OCEANOGRÁFICAS E HIDROGRÁFICAS"** ha sido desarrollado de acuerdo a los objetivos establecidos.

Como director del proyecto considero que el trabajo es satisfactorio y amerita ser presentado por sus autores.

Cordialmente,

LUZ STELLA ROBLES.

Cartagena de Indias, 29 de abril del 2002

Señores
COMITÉ PROYECTO DE GRADO
Facultad de Ingeniería de Sistemas
La ciudad

Distinguidos señores.

Por medio de la presente me permito informales que el proyecto de grado titulado **"DESARROLLO E IMPLEMENTACIÓN DE UN SISTEMA DE INFORMACIÓN PARA EL LABORATORIO DE QUÍMICA Y MICROBIOLOGÍA DEL CENTRO DE INVESTIGACIONES OCEANOGRÁFICAS E HIDROGRÁFICAS"** ha sido desarrollado de acuerdo a los objetivos establecidos.

Como asesor del proyecto considero que el trabajo es satisfactorio y amerita ser presentado por sus autores.

Cordialmente,

VICTOR HENRY BAEZ PINZON.

Cartagena de Indias, 29 de abril del 2002

Señores
COMITÉ PROYECTO DE GRADO
Facultad de Ingeniería de Sistemas
La ciudad

Distinguidos señores.

Por medio de la presente me permito informales que el proyecto de grado titulado **"DESARROLLO E IMPLEMENTACIÓN DE UN SISTEMA DE INFORMACIÓN PARA EL LABORATORIO DE QUÍMICA Y MICROBIOLOGÍA DEL CENTRO DE INVESTIGACIONES OCEANOGRÁFICAS E HIDROGRÁFICAS"** ha sido desarrollado de acuerdo a los objetivos establecidos.

Como autores del proyecto consideramos que el trabajo es satisfactorio y amerita ser presentado ante ustedes.

Cordialmente,

MARIO IGNACIO DE LEON CÉSPEDES **JOSE LUIS PALLARES VARELA**

Articulo 107.

La Universidad Tecnológica de Bolívar, se reserva el derecho de propiedad intelectual de todos los trabajos de grado aprobados y no pueden ser explotados comercialmente sin su autorización.

NOTA DE ACEPTACIÓN

_____

_____

_____

_____
Presidente del Jurado

_____
Jurado

_____
Jurado

Cartagena de Indias D. T. y C. 29 de abril de 2002

Doy Gracias a Dios por haberme dado sabiduría, tenacidad y fortaleza para culminar esta meta que emprendí hacen 5 años. Igualmente le pido me siga mostrando el camino como hasta ahora lo ha hecho, para emprender quizás la parte más difícil, como es mi ejercicio profesional.

A mis padres y familiares quienes con su ternura, amor y compresión me apoyaron en los momentos más difíciles de ésta ardua etapa de mi vida y a quienes debo gran parte de este éxito que hoy con gran esfuerzo y sacrifico finalizo.

Mario Ignacio De León Céspedes

A mi mamá, de todo corazón por ser la persona a la que le debo este gran paso en mi vida, la que siempre me ha apoyado en todo momento.

A mi abuela, quien siempre ha estado allí ayudándome, motivándome a seguir siempre a delante.

A mi tío, por su apoyo y sus sabios consejos.

A mis hermanos y a toda mi familia en general por su apoyo y comprensión a lo largo de mi formación profesional. Y sobre todo le agradezco a Dios por esta oportunidad que me ha dado.

José Luis Pallares Varela

AGRADECIMIENTOS

Los autores expresan sus más sinceros agradecimientos a:

LUZ STELLA ROBLES, directora del proyecto. Agradecemos su espíritu de colaboración, sus valiosos aportes y apoyo en todo momento.

VICTOR HENRY BAEZ, asesor del proyecto. Agradecemos sus valiosos aportes, orientaciones, paciencia y apoyo incondicional.

WILDERMAN CEREN PRENS, asesor externo. Agradecemos su importante colaboración a lo largo de la realización del proyecto; así como su deseo de apoyarnos incondicionalmente.

GUSTAVO TOUS, STELLA BETANCOUR Y JAVIER LLAMAS. Gracias por facilitarnos en todo momento la información necesaria para culminar satisfactoriamente este proyecto, así como sus orientaciones y ayuda incondicional.

CRISTINA OSORIO. Gracias por tu apoyo incondicional; por tu compañía y tus palabras de aliento que no nos dejaban desfallecer.

CONTENIDO

LISTA DE FIGURAS

Pág

LISTA DE TABLAS.

ANEXOS

Pág.

INTRODUCCIÓN

Este documento le permitirá obtener una orientación acerca de cómo implementar aplicaciones Cliente/Servidor teniendo en cuenta todos los aspectos imprescindibles para el desarrollo óptimo y eficiente en la cual entidades lógicas (Cliente/Servidor) independientes, trabajan en conjunto comunicándose a través de una red de computadoras, utilizando herramientas interactivas con el fin de accesar y manejar grandes volúmenes de información que se encuentra en diferentes sitios, plataformas computacionales y bases de datos.

En toda relación Cliente/Servidor se garantiza una total independencia entre el Cliente y el Servidor de manera que al Cliente le interesan los resultados que el Servidor le ofrece, más no el proceso con que los produce. La integración Cliente/Servidor es bastante relativa y depende no solo del proyecto a realizar, sino también e incluso en mayor medida de la plataforma de base de datos utilizada.

El Cliente es un proceso que se ejecuta en uno o varios nodos de una red y realiza peticiones de servicio a un Servidor, mientras que el Servidor es un proceso que se está ejecutando en un nodo de la red que controla el acceso a un determinado recurso generalmente una computadora que ejecuta acciones para otras computadoras; la red es el medio de transmisión de datos donde los mensajes son intercambiados por medio de protocolos

estándar de comunicación. No obstante se pueden dar relaciones Cliente/Servidor dentro de la misma máquina.

Cuando el cliente produce una petición, el Servidor despierta y atiende al Cliente; cuando el servicio culmina, el Servidor vuelve a un estado de espera.

De acuerdo con la forma de prestar el servicio, se pueden considerar dos tipos de Servidores: Servidores interactivos y Servidores concurrentes. De acuerdo a la naturaleza del servicio que el Servidor ofrece a sus Clientes, se puede considerar diferentes tipos de Servidores: Servidores de archivo, Servidores de bases de datos, Servidores de transacciones, Servidores de Groupware, Servidores de Objeto, Servidores Web entre muchos otros.

Las bases de datos son un conjunto de archivo interrelacionados creados y manejados por un sistema de gestión o de administración de bases de datos DBMS (Data Base Management System) el cual es un software que se encarga de controlar la organización, el almacenamiento, la recuperación, la seguridad y la integridad de los datos en una base de datos. Acepta solicitudes de la aplicación y genera ordenes al sistema operativo para que se transfieran los datos apropiados.

1. ANTECEDENTES, EL PROBLEMA Y LOS OBJETIVOS

1.1 PLANTEAMIENTO DEL PROBLEMA

1.1.1 Antecedentes del problema. El Centro de investigaciones Oceanográficas e Hidrográficas (CIOH) es un centro dedicado a realizar investigaciones básicas y aplicadas en las diferentes ramas de la oceanografía e hidrografía Colombiana, orientando sus esfuerzos al aprovechamiento de los recursos naturales.

El CIOH cuenta con un personal altamente calificado, quienes se comprometen de principio a fin, con la terminación de cada investigación. Su finalidad, entre otras cosas es la de brindar resultados confiables y acordes con las normas de calidad vigentes y reglas establecidas con el reglamento interno de la Institución.

La información que se generaba en los laboratorios de La División de Estudios Ambientales (DESAM) se archivaba y utilizaba de forma manual, utilizando la hoja de calculo excel en carpetas de forma impresa, representando así una labor ardua y demorada que no está exenta de errores. Lo anterior debido a que no se cuenta con un sistema computarizado organizado y especifico que permita el almacenamiento y procesamiento de la información generada.

El CIOH a través de la sección de contaminación (SECOM) ha querido ir acorde con la tecnología, y está en procura de sistematizar todos sus procesos. De lo cual ha surgido la necesidad de elaborar un sistema de información que permita calcular de manera eficiente, rápida y exacta todos los datos que en esta sección se requieren.

Con el presente proyecto se pretende proporcionar al CIOH un recurso que les permita cubrir con la necesidad de sistematización de la información en el área antes mencionada.

El procesamiento de información, implementado por los laboratorios de la Sección de Estudios de Contaminación del CIOH, ha variado en los últimos años, inicialmente los registros de los datos tomados en campo y generados en los laboratorios eran consignados en papel. Esta información fue organizada en periodos posteriores utilizando como herramienta la hoja de calculo Excel, sin embargo el tiempo invertido y la presentación de los formatos de resultados marcaron la necesidad de diseñar y crear un sistema de Información computarizado que recopilara toda la información generada por estudios e investigaciones desarrolladas en la Sección.

Desde la implantación de este sistema de información, los laboratorios de química, biología y microbiología han permitido organizar los resultados de una forma más estructurada e integral, los registros llevan anotaciones de campo de parámetros asociados a las variables ambientales a evaluar. El almacenamiento de la información en un solo sitio y la integridad de la misma, es fundamental para la generación de estudios y proyectos necesariamente sustentados por bases históricas de zonas o áreas de investigación. Es importante resaltar que además este sistema de información permite la participación más eficaz del CIOH con la

Red Nacional de Calidad Ambiental, integrada por Corporaciones Autónomas Regionales del Caribe y Pacífico e institutos de Investigación, todos ellos liderados por el INVEMAR (Instituto de Investigaciones Marinas y Costeras de Punta de Bentín).

Con el desarrollo e implementación de este nuevo sistema de información, en los laboratorios de la división de estudios ambientales se ha establecido un antes y después de la implantación de este nuevo sistema de información.

Antes

Los procedimientos para preparación de material y elementos necesarios para una salida de campo, no estaban documentados, planillas de registro y formatos de verificación no eran tomados en cuenta.

Después

En la actualidad todo está organizado en formatos que permiten llevar un registro de los procedimientos aplicados y así evitar que se olviden elementos, se atrasen las salidas o se dejen de tomar datos, situación que antes eran muy frecuente por la ausencia de formatos y planilla definidas. Ver anexo 2.

Antes

Para la actividad de la toma de datos en el campo se tienen formatos de registros de variables *in situ* (directamente registradas en el medio), tablas de control de toma de parámetros y mapas de estaciones con su ubicación geográfica. Este tipo de organización se ha implementado a partir del año 2000.

Después

Existen formatos definidos por lo que muchas variables y muestras de campo que antes se dejaban de tomar gracias a la ausencia de estos documentos; ahora es posible ya que se encuentra todo integrado en éste nuevo sistema de información.

Antes

La información recibida de campo era almacenada en formatos de registro diseñados para establecer un control de los datos y resultados. Este proceso antes era almacenado en papel y en formato Excel.

Después

La creación de este Sistema de Información que contiene las especificaciones requeridas por los laboratorios de la Sección de Estudios de Contaminación, ha permito que la implementación de este sistema tenga un grado de calidad óptimo y eficiente.

Antes

Cuando se llegaba a los laboratorios después de una salida de campo o recolección de muestras, y se comenzaba a recopilar toda la información pertinente a éstas, como por ejemplo el Sitio de muestreo donde fueron tomadas las muestras, las estaciones, los tipos de preservante que se añadieron, etc; todo el tiempo empleado para recopilar y almacenar esta información era de aproximadamente de treinta (30) minutos por muestra.

Después

Con la implementación de este nuevo sistema de información, el tiempo aproximado de recopilación y almacenamiento de muestras fue de 20 minutos, mientras los usuarios se familiarizaban y mecanizaban el ingreso de la información al nuevo sistema. Cuando se aprendió a manejar el software o el nuevo sistema, el tiempo bajó a 10 minutos aproximadamente por muestra; reduciendo así considerablemente el tiempo de ingreso de la información de éstas, en comparación a los tiempos empleados anteriormente cuando no existía este sistema.

1.1.2 Descripción del problema. Hoy en día la mayoría de las empresas se encuentran en procura de sistematizar sus procesos, puesto que brinda mayor rapidez y exactitud en la transformación de datos y entrega de resultados. Debido a esto, los laboratorios de química, biología y microbiología han decidido cambiar el proceso manual, actualmente aplicado en la Sección de Estudios de Contaminación (SECON) en lo concerniente a las etapas de recolección, admisión, procesamiento, consulta y entrega de información de los análisis que allí se realizan, por un sistema de información mas seguro, exacto, rápido y fácil de manejar por el usuario.

Para la realización del análisis e implementación de este sistema de información se deben tener en cuenta factores como el lenguaje de diseño y montaje de la base de datos, la plataforma computacional de trabajo y el lenguaje de enlace entre la base de datos y los formularios de captura e impresión de datos.

Actualmente se utilizan diferentes herramientas de Licencia Pública General (GPL) para el manejo de bases de datos y de interfaces entre las mismas, las cuales poseen propiedades que las convierten en una herramienta para el desarrollo de este proyecto.

Valiéndonos de esta propiedad y de los distintos métodos de diseño, elaboración e implementación de bases de datos, elaboración de interfaces entre estas bases de datos y un lenguaje de programación lo que se propone es integrar toda esta información a través de un software que permita agilizar el proceso de admisión, procesamiento y entrega de información de los diferentes análisis que se realizan en los laboratorios de esta entidad.

El proyecto básicamente será realizar una aplicación cliente para los laboratorios de Química y Microbiología, las cuales tendrán su Base de Datos desarrollada en MySQL y ubicada en un servidor remoto el cuál trabaja con Linux como plataforma computacional, la intercomunicación entre la aplicación cliente y la Base de Datos se realizará con MySQL ODBC drivers. La base de datos realizada podrá ser consultada sólo por usuarios autorizados desde la página WEB del CIOH utilizando como herramienta PHP (Hipertext Preprocesor).

1.1.3 Formulación del problema. ¿Es posible el diseño e implementación de un sistema de información integral para la admisión, procesamiento y entrega de resultados de la sección de contaminación más específicamente en el laboratorio de química, biología y microbiología?

¿Facilitaría el diseño de un sistema de información aplicado los procesos de recolección, admisión y posterior entrega de datos generados por los laboratorios en sus diversos análisis?

1.2 IMPACTO DE LA INVESTIGACIÓN

El diseño e implementación del sistema de información generará cambios favorables en el manejo de la información de los laboratorios de la sección de contaminación del CIOH, puesto que permitirá el procesamiento de los datos de una manera interactiva y amigable para los usuarios finales, proporcionándoles de esta manera agilidad en los cálculos, menor tiempo de procesamiento y la obtención de datos más confiables y exactos. El proyecto además brindará la posibilidad de consultar la base de datos e información pertinente a los laboratorios desde una página WEB que se unirá a la que actualmente tiene el CIOH, permitiendo así, que personas autorizadas puedan ver y consultar la información no solo a nivel interno sino incluso desde lugares diferentes al Centro.

1.3 JUSTIFICACIÓN

La escogencia del proyecto, se basa principalmente en la solicitud del Centro de Investigaciones Oceanográficas e Hidrográficas e inquietud de los autores del mismo, en diseñar e implementar un sistema de información que les permita almacenar los resultados de las muestras analizadas, procesarlos, consultarlos en una estación de trabajo o vía Internet,

eliminar registros y cumplir con las normas técnicas necesarias para lograr la acreditación de los laboratorios de la Sección de Estudios Ambientales y Contaminación. Esto se traduciría en notables beneficios a la hora de desarrollar los proyectos de investigación y control de las muestras que allí se realizan, permitiéndoles agilidad en los procesos de manejo de información, exactitud y sobre todo integridad de los datos obtenidos.

1.4 OBJETIVOS DE LA INVESTIGACIÓN

1.4.1 Objetivo general. Diseñar e implementar un sistema de información para el laboratorio de química, biología y microbiología del Centro de Investigaciones Oceanográficas e Hidrográficas; que almacene y procese los resultados de los análisis de las muestras microbiológicas, fisicoquímicas y biológicas analizadas, con el fin de apoyar todas las actividades de investigación que se llevan a cabo en dicho laboratorio, y que involucran el manejo de información sistematizada.

1.4.2 Objetivos específicos.

- Diseñar y realizar los formularios de captura e impresión de datos del sistema de información en el lenguaje de programación Visual FoxPro.

- Diseñar e implementar la base de datos del sistema de información en MySQL, teniendo en cuenta que se trabajará con una filosofía Cliente/Servidor y se utilizarán todos los conceptos relacionados con la misma.

- Gestionar y supervisar el ingreso de la información a la base de datos de los laboratorio de química, biología y microbiología para lograr de esta manera que los datos se encuentren integrados.

- Diseñar los mecanismos de acceso de usuarios autorizados para las consultas a la Base de Datos realizadas desde la Página Web del CIOH o desde las terminales donde se vaya a instalar el nuevo Sistema de Información.

- Establecer las interfaces entre la Base de Datos que se encontrará alojada en el servidor principal del Centro de Investigaciones Oceanográficas e Hidrográficas y la página WEB, utilizando como herramienta PHP.

- Entregar el software terminado junto con los manuales de usuario y manuales técnicos del mismo.

1.5 METODOLOGIA.

El método a utilizar será la investigación aplicada al desarrollo tecnológico, porque los laboratorios de química, biología y microbiología del Centro de Investigaciones Oceanográficas e Hidrográficas necesita estar a la Vanguardia de la tecnología, de esta forma se diseñará un nuevo sistema computarizado para el manejo de la información que colocará a éste Centro en un nivel moderno y eficaz al momento de prestar sus servicios.

2. ESTRATEGIA METODOLÓGICA DE LA INVESTIGACIÓN

2.1 MARCO TEÓRICO

Este proyecto se fundamenta en una serie de conceptos adquiridos a lo largo de la carrera, uniendo toda esa información con el fin de ponerla al servicio del Laboratorio de Química, Biología y microbiología del Centro de Investigaciones Oceanográficas e Hidrográficas, de tal forma que se mejore el proceso de recolección, admisión, procesamiento y posterior entrega de resultados en dicha entidad.

En nuestro proyecto lo que básicamente se plantea es diseñar, desarrollar e implementar un sistema de información basado en los requerimientos exigidos por el Centro de Investigaciones Oceanográficas e Hidrográficas y por algunas entidades como CARDIQUE, verificando que los cálculos y la información que procese el sistema a implementar cumpla las normas establecidas por las distintas entidades mencionadas anteriormente.

Las muestras constituyen la "materia prima" principal para todos los estudios de contaminación y microbiología que se realizan en los laboratorios pertinentes a ésta entidad. La toma de una muestra consiste en ir al campo; como por ejemplo a la Bahía de Cartagena, Ciénaga de la Virgen, Santa Marta y otros sectores de la Costa Caribe, y recolectar cada una

de las muestras de los diferentes sectores antes mencionados, clasificarlas, añadirles los preservativos necesarios para conservarla los cuales varían de una muestra a otra y almacenarla para luego llevarla al laboratorio. Para clasificar las muestras se tienen en cuenta diferentes factores como por ejemplo el sitio de donde se tomó, la profundidad de recolección de la misma, el tipo de muestra como por ejemplo sedimentos, cuerpo de agua, etc. Además factores ambientales como el viento, la nubosidad y la humedad entre otros.

Después de realizada la recolección de las muestras se llevan al laboratorio para hacer la respectiva admisión, la cual lleva el código de la muestra asignado en este mismo sitio, la lista de los diferentes parámetros que se desean evaluar, cabe destacar que a una misma muestra se le pueden realizar varios parámetros coincidentes entre una y otra. En la admisión también se incluyen otros factores indispensables para los estudios o parámetros que se le van a realizar a la misma; así como la persona responsable de la admisión.

Después de haber realizado la admisión, se clasifican nuevamente las muestras para realizar los respectivos estudios, dependiendo del periodo de vida de cada una de ellas. Luego de haberles analizado los parámetros y haber tomado los valores, se procede a realizar los cálculos y a diligenciar los respectivos formatos para almacenar esa información.

Al finalizar el diseño, desarrollo e implementación del sistema de información, éste permitirá tomar todas las especificaciones de las muestras y procesarlas.

2.2 ARQUITECTURA CLIENTE SERVIDOR

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

La filosofía de la arquitectura Cliente/Servidor es la combinación de tres tecnologías: sistemas manejadores de bases de datos relacionales (RDBMS), redes y una interfaz Cliente. Cada elemento contribuye a la totalidad de la plataforma con funciones bien especificas pero independientes entre sí.

Algunas definiciones acerca de la arquitectura Cliente/Servidor son:

- Cualquier combinación de sistemas que pueden colaborar entre sí para dar a los usuarios toda la información que ellos necesiten sin que tengan que saber donde está ubicada.

- Es una arquitectura de procesamiento cooperativo donde uno de los componentes pide servicio a otro.

- Es un procesamiento de datos de índole colaborativo entre dos o más computadoras conectadas a una red.

El término Cliente/Servidor es originalmente aplicado a la arquitectura de software que describe el procesamiento entre dos o más programas: Una aplicación y un servicio soporte.

Aunque existen diversas definiciones relacionadas con la arquitectura Cliente/Servidor es claro que todas expresan lo mismo con diferentes palabras. En definitiva podemos decir que la arquitectura Cliente/Servidor es una tecnología en la cual entidades lógicas (Cliente/Servidor) independientes, trabajan en conjunto comunicándose a través de una red de computadoras, utilizando herramientas interactivas con el fin de accesar y manejar grandes volúmenes de información que se encuentran en diferentes sitios, plataformas computacionales y bases de datos.

La gama de acciones que se realizan entre un Cliente y el Servidor es casi ilimitada. Lo más importante acerca de la relación Cliente/Servidor es que al Cliente no le importa cómo hace la tarea el servidor, solo le interesan los resultados.

Con la arquitectura Cliente/Servidor los usuarios no están limitados a un tipo de sistema o plataforma. Las estaciones de trabajo pueden ser PC's, Macs, estaciones de trabajo UNIX o cualquier combinación, así como poder correr múltiples sistemas operativos.

A continuación enunciaremos algunas de las ventajas y desventajas que presenta el uso de la tecnología Cliente/Servidor:

Ventajas

Mayor seguridad: Los datos son accedidos en forma indirecta.

Mejor rendimiento: Ya que por lo general el equipo servidor es más poderoso que el cliente y los PC's no leen los datos sino que la lectura y procesamiento de los datos se hacen en el servidor y él transmite los resultados a la PC para que se encargue de mostrarlos. La

efectividad en relación con el costo es mucho mayor. Los clientes solo necesitan el poder suficiente para ejecutar adecuadamente la aplicación frontal.

Desventajas

Complejidad: no son fáciles de configurar.

Requerimientos: Para dar servicio a muchos usuarios, el componente cliente frecuentemente necesita ejecutarse en un PC costoso. Las aplicaciones del servidor tienden a ser grandes y complejas y generalmente necesitan mucha memoria.

Costo: El rendimiento del servidor se reduce al aumentar el número de usuarios y peticiones de servicio, terminando el servidor por ser de uso exclusivo para el manejo de la base de datos o robusteciendo la maquina al doble para que cumpla las funciones de servidor de archivos y de base de datos.

2.2.2 Evolución de la arquitectura cliente servidor. Se puede observar que ha habido una clara evolución desde los inicios de los Sistemas Informáticos hasta llegar a los sistemas utilizados hoy en día. Esta evolución se ha dado como consecuencia de las nuevas necesidades que surgían a medida que evolucionaban los sistemas informáticos.

A continuación se menciona la forma como ha evolucionado la arquitectura Cliente/Servidor.

- **La era de la computadora central**

  Desde sus inicios el método de administración de datos a través de computadoras se basa en el uso de terminales remotas que se conectaban de manera directa a una

computadora central. Dicha computadora central se encargaba de prestar servicios solo a un grupo exclusivo de usuarios. El personal del área de sistemas se encargaba de integrar la información cuando las necesidades de los usuarios lo exigían.

- **La era de las computadoras dedicadas**

  Esta es la era en la que cada servicio empleaba su propia computadora que permitía que los usuarios de ese servicio se conectaran directamente. Esto debido a la aparición de computadoras pequeñas, de fácil uso, más económicas y más poderosas que las convencionales. Este modelo actualmente posee importantes limitantes. El primero que, conforme crece el número de usuarios que requieren acceso a los datos administrados por cada sistema, se presenta la necesidad de hacer uso de computadoras cada vez más poderosas en sus sistemas de entrada / salida. El segundo es que estas computadoras son incapaces de comunicarse entre si y por tanto la información compartida es nula.

- **La era de la conexión libre**

  La aparición de las computadoras de escritorio de manera masiva permitió que parte apreciable de la carga de trabajo de computo tanto en el ámbito de cálculo como en el ámbito de la presentación se lleven a cabo desde el escritorio del usuario. En muchos de los casos el usuario obtiene la información que necesita de alguna computadora de servicio.

- **La era del computo a través de redes**

  Esta es la era que se basa en el concepto de redes de computadoras, en la que la información reside en una o varias computadoras, los usuarios de ésta información hacen

uso de computadoras para trabajar y todas ellas se encuentran conectadas entre sí. Con esto todos los usuarios pueden accesar a la información de todas las computadoras y a la vez los diversos sistemas intercambian información.

- **La era de la arquitectura Cliente/Servidor**

  En esta arquitectura la computadora de cada uno de los usuarios, llamada Cliente, produce una demanda de información a cualquiera de las computadoras que proporcionan información, conocidas como Servidores; estos últimos responden a la demanda del Cliente que la produjo. Los Clientes y los Servidores pueden estar conectados a una red local o a una red amplia. Bajo este modelo cada usuario tiene la libertad de obtener la información que requiera en un momento dado proveniente de una o varias fuentes locales o distantes y de procesarla como según le convenga. Los servidores también pueden intercambiar información dentro de esta arquitectura.

Existen distintas manera de manipular la información, pero es claro que muchas de ellas no son las más eficientes, motivo por el cual se presentó la necesidad de crear un modelo que permitiera manejar grandes volúmenes de información de manera interactiva, rápida y segura otorgando acceso a varios usuarios al mismo tiempo y garantizando eficiencia en todas las operaciones realizadas.

Unos años atrás, el desarrollo de aplicaciones Cliente/Servidor era inevitable debido a que es más eficiente que el procesamiento centralizado. En este momento ya había Servidores eficientes y confiables. Existía tecnología para el desarrollo de la arquitectura Cliente/Servidor desde hacía ya bastantes años, la cual no era aprovechada al máximo. Se había establecido

un estándar para una Interface Cliente/Servidor; el ODBC SQL, adoptado por todos los fabricantes importantes de Servidores.

Los primeros trabajos conocidos para la arquitectura Cliente/Servidor fueron hechos por Sybase, a finales de los 80´s lanzaron un producto, para el segmento "Low –End", en conjunto con Microsoft, teniendo como soporte de la base de datos un Servidor OS/2 y como herramienta "Front End" básica el Dbase IV de Ashton Tate. El Dbase IV no se mostró como una herramienta adecuada.

En 1994, los principales fabricantes tradicionales (Informix, Oracle, Sybase) lanzaron al mercado poderosos servidores, IBM lanzo su producto DB2 para prácticamente todos los sistemas operativos importantes (además de sus clásicos MVS y VM, anunciaba AIX, OS/2, Windows NT, Hewlett Packard's UNÍX, Sun's UNÍX, etc) y Microsoft partió para su propio SQL Server para Windows NT.

Existía un conjunto de lenguajes "Front End" (Delphi, Foxpro, SQL, Visual Basic, etc). En aquel momento se creía que Visual Basic era el preferido en el mercado, cosa que está ocurriendo.

En estos dos últimos años, los servidores se han mostrado sólidos y eficientes, sus optimizadores probaron, en su mayoría, ser excelentes, debido a esto gran cantidad de empresas han adoptado el uso de aplicaciones Cliente/Servidor obteniendo resultados favorables.

Con una aplicación Cliente/Servidor, se gana solidez, rentabilidad, seguridad y consistencia en todo trabajo. Para aplicaciones medianas y pequeñas todos los Servidores son muy buenos, pero saber cual es el mejor Servidor depende de las necesidades y requerimientos en determinadas aplicaciones, y de las nuevas características que se vayan realizando como paralelismo, “read ahead”, etc. Cada nueva versión puede cambiar las posiciones. Para fortuna de los usuarios esta tecnología ha sido valorada y desarrollada por los proveedores que trabajan continuamente en forma paralela.

En la arquitectura Cliente/Servidor, es esencial el tránsito en la red. Un esquema Cliente/Servidor es optimo en la medida de que el tránsito en la red sea mínimo, porque utiliza la mínima potencia de máquina posible.

2.3 INTEGRACIÓN ENTRE CLIENTE Y SERVIDOR

En el ambiente Cliente/Servidor, el procesamiento se divide entre el sistema del Cliente y el Servidor. Estos programas están ligados por la red. El Cliente hace peticiones de los servicios que provee el Servidor.

La división de servicios efectuada por los Servidores de base de datos y las computadoras Cliente siguen una división natural de labores, tomando en consideración el potencial relativo de cada recurso. Como el Servidor de base de datos automáticamente maneja muchas de las

difíciles y complicadas tareas de transacciones, la responsabilidad de la aplicación es solo solicitar las consultas y manejar los datos resultantes.

Estas interacciones pueden parecer complejas, pero facilitan la obtención de información. Mientas más poderoso e inteligente sea el Servidor al que se esté conectando o mientras más poderoso e inteligente sea el Servidor al que su Servidor se convierte en Cliente, mayores probabilidades habrá de que se obtenga la respuesta con rapidez.

La integración Cliente/Servidor es bastante relativa y depende no solo del proyecto a realizar, sino también e incluso en mayor medida de la plataforma de base de datos utilizada. Según esto nos podemos encontrar los siguientes casos:

2.3.1 Presentación distribuida. En la Presentación Distribuida, 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). Desde el punto de vista del uso de los recursos, la arquitectura Cliente/Servidor es similar a la Arquitectura Centralizada. En síntesis la presentación distribuida cumple con las siguientes características:

- Se distribuye la interfaz entre el Cliente y la plataforma servidora.
- Las aplicaciones y los datos están ambos en el Servidor; similar a la arquitectura tradicional de un Host y Terminales.
- El PC se aprovecha solo para mejorar la interfaz gráfica del usuario.

Ventajas

- Revitaliza los sistemas antiguos.
- Bajo costo de desarrollo.
- No hay cambios en los sistemas existentes.

Desventajas

- El sistema sigue en el Host
- No se aprovecha la GUI y/ o LAN
- La interfaz del usuario se mantiene en muchas plataformas.

2.3.2 Presentación remota. La presentación remota cumple las siguientes características:

- La interfaz para el usuario esta completamente en el Cliente.
- La aplicación y los datos están en el Servidor.

Ventajas

- La interfaz del usuario aprovecha bien la GUI y la LAN.
- La aplicación aprovecha el Host.

Desventajas

- Las aplicaciones pueden ser complejas de desarrollar.
- Los programas de la aplicación siguen en el Host.

- El alto volumen de tráfico en la red puede hacer difícil la operación de aplicaciones muy pesadas.

2.3.3 Lógica distribuida. La lógica distribuida tiene las siguientes características:

- La interfaz está en el Cliente.
- La base de datos esta en el servidor.
- La lógica de la aplicación está distribuida entre el Cliente y el servidor.

Ventajas

- Arquitectura más corriente que permite manejar todo tipo de aplicaciones.
- Los programas del sistema pueden distribuirse al nodo más apropiado.
- Pueden utilizarse con sistemas existentes.

Desventajas

- Es difícil de diseñar.
- Difícil prueba y mantenimiento si los programas del Cliente y el Servidor están hechos en distintos lenguajes de programación.
- No son manejados por la GUI 4GL.

2.3.4 Administración de datos remotos. En la administración de datos remota, 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. La administración de datos remota cumple las siguientes características:

- En el Cliente reside tanto la interfaz como los procesos de la aplicación.
- Las bases de datos están en el servidor.

Todas las anteriores características es lo que comúnmente imaginamos como aplicación Cliente/Servidor.

Ventajas

- Configuración típica de la herramienta GUI 4GL
- Muy adecuada para las aplicaciones de apoyo a las decisiones del usuario final.
- Fácil de desarrollar, ya que los programas de aplicación no están distribuidos.
- Se descargan los programas del Host.

Desventajas

- No maneja aplicaciones pesadas eficientemente.
- La totalidad de los datos viajan por la red, ya que no hay procesamiento que realice el Host.

2.3.5 Bases de datos distribuidas. La base de datos distribuida tiene las siguientes características:

- La interfaz; los de la aplicación y parte de los datos de la base de datos están en el Cliente.

- El resto de los datos están en el Servidor.

Ventajas

- Configuración soportada por herramientas GUI 4GL.
- Adecuada para las aplicaciones de apoyo al usuario final.
- Apoya acceso a datos almacenados en ambientes heterogéneos.
- La ubicación de los datos es transparente para la aplicación.

Desventajas

- No maneja aplicaciones grandes eficientemente.
- El acceso a la base de datos distribuida es dependiente del proveedor del software administrador de bases de datos.

2.4 CARACTERISTICAS DE LA ARQUITECTURA CLIENTE/SERVIDOR

En general los sistemas Cliente/Servidor poseen las siguientes características distintivas:

Servicios. Cliente/Servidor se base en la presentación y consumo de servicios;
donde el Servidor es el proveedor y el Cliente es el consumidor de los mismos.
Recursos Compartidos. Un servidor esta en capacidad de manipular uno o más Clientes a la vez y controlar eficientemente el acceso a recursos compartidos.

Protocolos Asimétricos. La relación que se establece entre Cliente y Servidores es de "muchos a uno". El Servidor siempre está a la espera de una solicitud de servicio. La comunicación se inicia en el momento que un Cliente realice dicha solicitud.

Transparencia de Ubicación. Para el Cliente es transparente la ubicación del proceso servidor, éste puede residir ya sea en la misma máquina o en una máquina remota, en pocas palabras 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.

Mezcla e Igualdad. Las aplicaciones Cliente/Servidor son independientes del software y el hardware así como del sistema operativo; se puede mezclar e igualar las distintas plataformas de Clientes y servidores.

Intercambio Basado en Mensajes. Los clientes y servidores interactúan a través de un mecanismo de transmisión de mensajes. El mensaje es el mecanismo que utiliza esta arquitectura para enviar y recibir servicios.

Encapsulamiento de Servicios. El mensaje le indica al servidor qué servicios se solicitan, mientras la interfaz que se utilice para la publicación del mensaje sea la misma, es independiente para el Cliente qué Servidor responde su solicitud.

Facilidad de escalabilidad. Los sistemas Cliente/Servidor pueden ser escalados horizontal o verticalmente. La primera permite añadir o eliminar Clientes sin afectar significativamente el rendimiento y la segunda permite cambiar a otro Servidor o a servidores múltiples.

Integridad. La protección de la integridad de la información compartida es posible ya que el código y los datos del Servidor se conservan centralmente proporcionándole a los Clientes individualidad e independencia. Los cambios en el servidor implican pocos o ningún cambio en el cliente.

2.5 COMPONENTES FUNDAMENTALES

Como se ha venido diciendo, Cliente/Servidor es un modelo basado en la idea del servicio, en el que el cliente es un proceso consumidor de servicios y el servidor es un proceso proveedor de servicios. Además esta relación está establecida en función del intercambio de mensajes que es el único elemento de acoplamiento entre ambos. De aquí salen los tres componentes fundamentales sobre los cuales se desarrollan e implantan los sistemas Cliente/Servidor: el proceso Cliente que es quien inicia el diálogo, el proceso Servidor que pasivamente espera a que lleguen peticiones de servicio y la Red que permite la conectividad entre el Cliente y el Servidor para poder intercambiar mensajes.

Para entender en forma más ordenada y clara los conceptos y elementos involucrados en esta tecnología se puede aplicar una descomposición o arquitectura de niveles. Esta descomposición principalmente consiste en separar los elementos estructurales de esta tecnología en función de aspectos más funcionales de la misma:

Nivel de Presentación: Agrupa a todos los elementos asociados al componente Cliente.

Nivel de Aplicación: Agrupa a todos los elementos asociados al componente Servidor.

Nivel de comunicación: Agrupa a todos los elementos que hacen posible la comunicación entre los componentes Cliente y servidor.

Este modelo de descomposición en niveles, como se verá más adelante, permite introducir más claramente la discusión del desarrollo de aplicaciones en arquitecturas de hardware y software en planos.

2.5.1 Cliente .Es el proceso que permite al usuario formular los requerimientos y pasarlos al servidor, se le conoce con el término front-end. Este normalmente maneja todas las funciones relacionadas con la manipulación y despliegue de datos, por lo que están desarrollados sobre plataformas que permiten construir interfaces gráficas de usuario (GUI), además de acceder a los servicios distribuidos en cualquier parte de la red.

Las funciones que lleva a cabo el proceso cliente se resumen en los siguientes puntos:

- Administrar la interfaz de usuario.
- Interactuar con el usuario.
- Procesar la lógica de la aplicación y hacer validaciones locales.
- Generar requerimientos de bases de datos.
- Recibir resultados del servidor.
- Formatear resultados.

2.5.2 Servidor. Es el proceso encargado de atender a múltiples clientes que hacen peticiones de algún recurso administrado por él. Al proceso servidor se lo conoce con el término Back-end.

El servidor normalmente maneja todas las funciones relacionadas con la mayoría de las reglas del negocio y los recursos de datos.

Las funciones que lleva a cabo el proceso servidor se resumen en los siguientes puntos:

- Aceptar los requerimientos de bases de datos que hacen los clientes.
- Procesar requerimientos de bases de datos.
- Formatear datos para trasmitirlos a los clientes.
- Procesar la lógica de la aplicación y realizar validaciones a nivel de bases de datos.

Un Servidor de base de datos hace referencia al SABD (sistema administrador de base de datos). Este es independiente del hardware en que se instala, debe cumplir con todas las funciones que caracterizan a un SABD relacional, debe brindar facilidades que permitan la administración de los datos, y entre éstas se debe tener al menos las siguientes:

- Un diccionario de datos
- Memorias temporales y de caché.
- Mecanismos para satisfacer las solicitudes de los usuarios.
- Mecanismos de bloqueo y control de la concurrencia.
- Seguridad de datos.

- Mecanismos de recuperación.
- Mecanismos de configuración.

El Servidor esta continuamente esperando peticiones de servicio. Cuando se produce una petición, el servidor despierta y atiende al Cliente. Cuando el servicio culmina, el Servidor vuelve a un estado de espera.

De acuerdo con la forma de prestar el servicio, se pueden considerar dos tipos de servidores:

Servidores interactivos. El servidor además de recoger la petición de servicio, se encarga de atenderla. Si el servidor es muy lento en atender a los clientes y hay una demanda de servicio alta, los tiempos de espera que se van a originar serán grandes.

Servidores concurrentes. El Servidor recoge cada petición de servicio y crea otros procesos para que se encarguen de atenderlas. El Servidor puede recoger peticiones a muy alta velocidad.

De acuerdo a la naturaleza del servicio que el Servidor ofrece a sus Clientes, se pueden considerar los siguientes tipos de servidores:

Servidores de archivo. Servidor donde se almacenan archivos y aplicaciones de productividad como por ejemplo procesadores de texto, hojas de calculo, etc. Permite que el Cliente realice peticiones de registro de archivo al servidor a través de una red.

Estos realizan varias funciones, pero se consideran como las más importantes: compartir archivos por medio de una red y la contribución de ellos para la creación de depósitos compartidos de documentos, imágenes, planos de ingeniería y datos de gran dimensión.

Servidores de Base de Datos. Servidores donde se almacenan las bases de datos, tablas, índices. Es uno de los Servidores que más carga tiene. El Cliente envía solicitudes de mensajes (en SQL) al Servidor de base de datos. Las respuestas de cada instrucción SQL se retornan por medio de la red, además como las solicitudes SQL y los datos solicitados se encuentran en la misma máquina, es el Servidor quien se encarga de encontrar los datos solicitados, para enviárselos al Cliente. Es decir se aprovecha al máximo su potencial, teniendo como resultado un uso eficiente de la capacidad de procesamiento distribuida que posee. Estos son una gran ayuda en el funcionamiento de los sistemas de apoyo de proceso de toma de decisiones que precisan de consultas especificas y reportes flexibles.

Servidores de transacciones. Servidor que cumple o procesa todas las transacciones. Valida primero y recién genera un pedido al servidor de bases de datos. En este tipo de Servidores el Cliente invoca procedimientos remotos los cuales ejecutan en el servidor un grupo de instrucciones SQL. Estas instrucciones agrupadas de SQL se les llama TRANSACCIONES. Un servidor de transacciones permite crear aplicaciones Cliente/Servidor generando el código tanto para el cliente como para el Servidor. El componente Cliente se encarga de la Interfaz Gráfica de Usuario (GUI: Graphical User Interface). El componente del servidor maneja por lo general las transacciones de SQL contra una base de datos. A dichas aplicaciones se les llama Procesamiento de Transacciones en Línea (OLTP: OnLine Transaction Processing).

Servidores de Groupware. Servidor utilizado para el seguimiento de operaciones dentro de la red. Este sistema permite establecer una comunicación con contacto directo entre personas. El Groupware es quien se encarga de dirigir la administración de información como texto, imagen, correo, tableros de avisos y flujo de trabajo.

Servidores de Objeto. En este tipo de Servidores las aplicaciones Cliente/Servidor se presentan como un conjunto de objetos de comunicación. Los objetos tantos del cliente como del servidor se comunican por medio de un Corredor de Solicitudes de Objetos (ORB: Object Request Broker). Contienen objetos que deben estar fuera del Servidor de base de datos. Estos objetos pueden ser videos, imágenes, objetos multimedia en general. El Cliente pide un método de un objeto remoto, el ORB se encarga de localizar una instancia del mismo tipo de servicios solicitado, una vez la localiza pide el método que se pidió y envía los resultados al objeto del Cliente.

Servidores Web. Un Servidor Web envía documentos cuando los Clientes los solicitan por su nombre, Clientes y Servidores se comunican por medio de un protocolo similar a RPC denominado HTP. Se usan como una forma inteligente para comunicación entre empresas a través de Internet. Este servidor permite transacciones con el acondicionamiento de un browser específico.

2.5.3 Red. Es el medio de transmisión de datos, donde los mensajes se intercambian por medio de protocolos estándar de comunicación. En la red se deben considerar al menos los siguientes aspectos:

Tabla 1 : Características de la red.

Técnicas de transmisión	Tipo de banda
Tipo de cableado	cable coaxial, fibra óptica, etc.
Topología	bus, estrella, anillo.

2.6 ESQUEMA GENERICO DE UN SERVIDOR Y DE UN CLIENTE

En un ambiente distribuido las acciones que debe llevar a cabo el programa servidor son las siguientes:

- Abrir el canal de comunicación e informar a la red tanto de la dirección a la que responderá como de su disposición para aceptar peticiones de servicio.
- Esperar a que un Cliente pida servicio en la dirección que él tiene declarada.
- Cuando recibe una petición de servicio, si es un servidor interactivo, atenderá al Cliente. Si es un Servidor concurrente, creará un proceso para que de servicio al Cliente.
- Queda en espera de nuevas peticiones de servicio.

En un ambiente distribuido el programa Cliente debe llevar a cabo las siguientes acciones:

- Abrir el canal de comunicación y conectarse a la dirección de red atendida por el Servidor.

- Enviar al Servidor un mensaje de petición de servicio y esperar hasta recibir la respuesta.
- Cerrar el canal de comunicaciones y terminar la ejecución.

2.7 CLASIFICACIÓN DEL MODELO CLIENTE/SERVIDOR

Uno de los aspectos claves para entender la tecnología Cliente/Servidor, y por lo tanto contar con la capacidad de proponer, promocionar y llevar a cabo soluciones de este tipo, es llegar a conocer la arquitectura de este modelo y los conceptos o ideas asociados al mismo. Más allá de entender los componentes Cliente/Servidor/Red, es preciso analizar ciertas relaciones entre éstos, que pueden definir el tipo de solución que se ajusta de mejor forma a las estadísticas y restricciones acerca de los eventos y requerimientos de información que se obtuvieron en la etapa de análisis de un determinado proyecto. De hecho el analista o líder deberá conocer estos eventos/restricciones del negocio para, a partir de allí, hacer las consideraciones y estimaciones de la futura configuración, teniendo en cuenta aspectos como por ejemplo, la oportunidad de la información, tiempo de respuesta, tamaños de registros, tamaño de bases de datos, estimaciones del tráfico de red, distribución geográfica tanto de los procesos como de los datos, etc.

En tal sentido se presenta, en primer lugar, un esquema de clasificación basado en los conceptos de Fat Client/Thin Client, Fat Server/Thin Server, Two Tier, Three Tier, los cuales están bastante generalizados y sobrecargados de definiciones, pero que se consideran

necesarios y útiles para la aplicación del modelo Cliente/Servidor. Después se presentará un esquema de clasificación según otros aspectos.

2.7.1 Por tamaño de componentes. Este tipo de clasificación se basa en los grados de libertad que brinda el modelo Cliente/Servidor para balancear la carga de proceso entre los niveles de presentación, aplicación y base de datos. Dependiendo de que segmento de las capas de software tenga que soportar la mayor o menor carga de procesamiento, se habla de Fat Cliente (Thin Server) o Fat server (Thin Client). Consideraciones de este tipo son importantes al momento de decidir una plataforma de desarrollo/explotación, al punto que pueden definir la viabilidad o no de las mismas para enfrentar un cierto número de restricciones impuestas por una problemática a resolver.

2.7.1.1 Fat client (Thin Server). En este esquema de arquitectura el grueso de la aplicación es ejecutada en el cliente, es decir, el nivel de presentación y el nivel de aplicación corren en un único proceso cliente, y el servidor es relegado a realizar las funciones que provee un administrador de base de datos.

En general este tipo de arquitectura tiene mejor aplicación en sistemas de apoyo de decisiones (DSS: Decision Support System) y sistemas de información ejecutiva (EIS: Executive Information System) y como se concluirá más adelante, tiene pocas posibilidades de aplicarse en sistemas de misión crítica.

2.7.1.2 Fat server (Thin Client). Este es el caso opuesto al anterior, el proceso cliente es restringido a la presentación de la interfaz de usuario, mientras que el grueso de la aplicación corre por el lado del servidor de aplicación.

En general este tipo de arquitectura presenta una flexibilidad mayor como para desarrollar un gran espectro de aplicaciones, incluyendo los sistemas de misión crítica a través de servidores de transacciones.

2.7.2 Capas de un Sistema Cliente / Servidor. En la arquitectura Cliente/Servidor clásica existen dos capas (Two Tier Architecture), una que reside en la estación de trabajo Cliente y una segunda que reside en el servidor de base de datos. La parte residente en el Cliente resuelve tanto los problemas de diálogo con el usuario, como los de lógica (consistencia, cálculos, etc.) e incluso determina los accesos a la base de datos, mientras que en el Servidor se ejecutan, simplemente los comandos SQL determinados por la otra parte. Este esquema de aplicación Cliente/Servidor es el más utilizado, se considera como una arquitectura de dos niveles aún cuando el servidor deba acceder a otro servidor. El servidor se encuentra ubicado por lo general en otra máquina y suele ser un gestor de bases de datos. Normalmente el Cliente debe saber con que servidor debe conectarse.

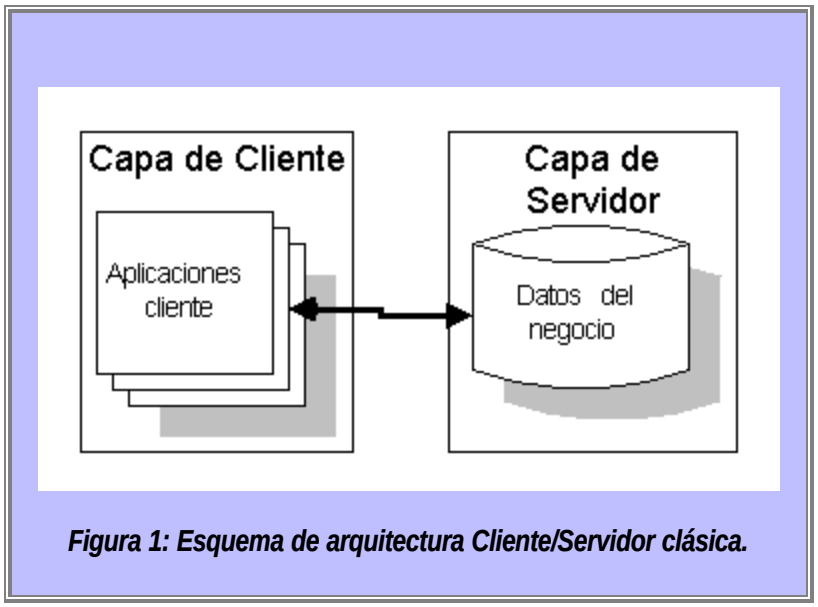

Figura 1: Esquema de arquitectura Cliente/Servidor clásica.

La arquitectura Cliente/Servidor de dos capas permite que se despreocupe de llevar a cabo la codificación de muchas validaciones en las aplicaciones de dichas arquitecturas.

Cualquier aplicación que acceda a la base de datos tendrá como beneficio la validación automática sin tener que añadir una línea de código. Esto asumiendo que la información se encuentra en un gestor de bases de datos potente.

Ventajas

Presenta una estructura de desarrollo bastante simple por cuanto el programador típicamente maneja un único ambiente de desarrollo (es más simple respecto de Cliente/Servidor en tres planos, puesto que reduce una capa de programación, como se verá más adelante).

Desventajas

La gran cantidad de información que viaja al cliente congestiona demasiado el tráfico de red, lo que se traduce en bajo rendimiento.

Por su bajo rendimiento esta estructura tiene un bajo espectro de aplicación, limitándose a la construcción de sistemas no críticos.

Existen estructuras de tres niveles (Three Tier Architecture) o más en las cuales el cliente se comunica con un "broker" o ruteador de servicios, quien se conecta con el servidor conveniente a los efectos de hacer la consulta.

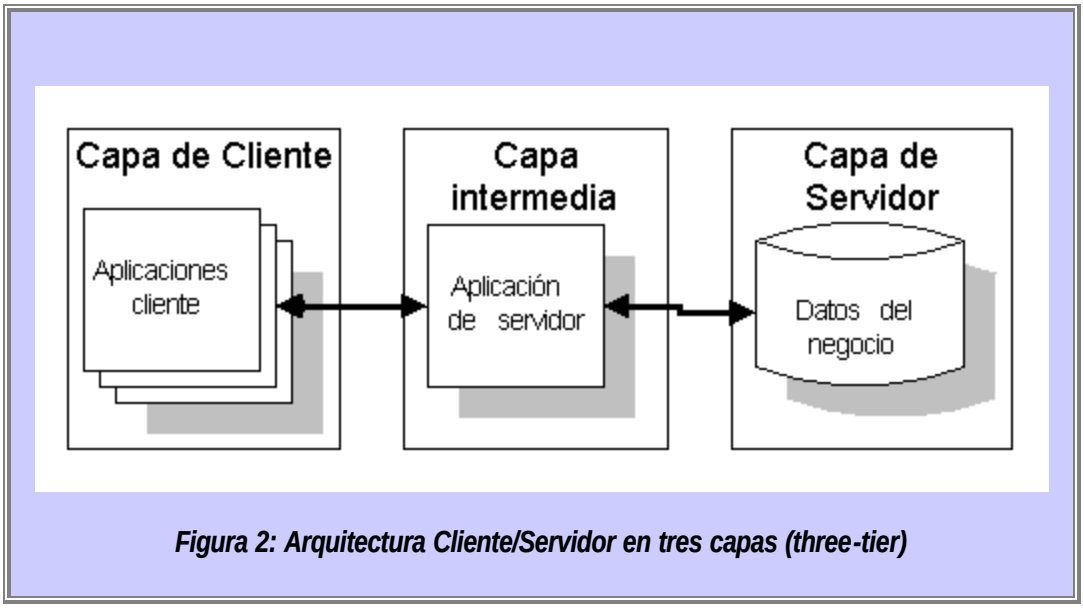

Figura 2: Arquitectura Cliente/Servidor en tres capas (three-tier)

Con la introducción de una capa intermedia es necesario un modo de comunicación que relacione aplicaciones Cliente con la aplicación que lleva a cabo las labores de la capa

intermedia. En este caso se tiene total libertad para escoger donde se coloca la lógica de la aplicación: en el Cliente, en el Servidor de base de datos o en otro(s) Servidores. También se tiene total libertad para elegir el lenguaje que se va a utilizar. Se utiliza un lenguaje general por lo que no existen restricciones de funcionalidad.

Ventajas

Reduce el tráfico de información en la red por lo que mejora el rendimiento de los sistemas (especialmente respecto de la estructura en dos planos).

Brinda una mayor flexibilidad de desarrollo y de elección de plataformas sobre la cual montar las aplicaciones.

Provee escalabilidad horizontal y vertical.

Se mantiene la independencia entre el código de la aplicación (reglas y conocimiento del negocio) y los datos, mejorando la portabilidad de las aplicaciones.

Los lenguajes sobre los cuales se desarrollan las aplicaciones son estándares lo que hace más exportables las aplicaciones entre plataformas.

Dado que mejora el rendimiento al optimizar el flujo de información entre componentes, permite construir sistemas críticos de alta confiabilidad.

El mismo hecho de localizar las reglas del negocio en su propio ambiente, en vez de distribuirlos en la capa de interfaz de usuario, permite reducir el impacto de hacer mantenimiento, cambios urgentes de última hora o mejoras al sistema.

Disminuye el número de usuarios (licencias) conectados a la base de datos.

Desventajas

Dependiendo de la elección de los lenguajes de desarrollo, puede presentar mayor complejidad en comparación con Cliente/Servidor de dos planos.

Existen pocos proveedores de herramientas integradas de desarrollo con relación al modelo Cliente/Servidor de dos planos, y normalmente son de alto costo.

Debido a estas desventajas es aquí la mayor importancia del Generador (aplicación creada en el presente trabajo).

2.8 VENTAJAS DE LA ARQUITECTURA CLIENTE / SERVIDOR.

Las ventajas de la arquitectura Cliente/Servidor son las siguientes:

Sistemas abiertos que permiten el intercambio con aplicaciones que no son propietarias, este concepto se aplica tanto a Hardware como Software.

Interoperatividad, componentes claves (Cliente, red, Servidor) trabajan juntos.

Escalabilidad, cualquiera de los componentes claves puede ser reemplazado cuando se necesite, ya sea para crecimiento o para reducir el procesamiento, sin tener mayor impacto en otros elementos.

Adaptabilidad, nueva tecnología (multimedia, redes de banda ancha, bases de datos distribuidas, pen cumputing) puede ser incorporada al sistema.

Costo accesible, el abaratamiento es asegurado por el uso de menos MIPs (millones de instrucciones por segundo) en cada plataforma.

Integridad de datos, mantenida por el servidor de bases de datos.

Accesibilidad, la forma de acceder a los datos es tan directa y fácil como se desee, los datos pueden ser accesados a través de WAN`s (Wide area Networks) con múltiples aplicaciones Clientes.

La distribución de tareas permite aumentar el desempeño con unos costos menores, el desempeño puede ser optimizado tanto por hardware como por procesos.

Seguridad, la seguridad de los datos es centralizada en el Servidor.

2.9 BASE DE DATOS RELACIONALES

Aunque existen diferentes tipos de bases de datos, la realidad es que el sistema de más uso es el relacional (Oracle, DB2, SQL Server, Informix, etc). Las bases de datos relacionales, fueron propuestas a finales de los años 60 por Codd con su teoría de las relaciones y fijaba sus aspectos fundamentales. Se tardo casi diez años en llevar a la práctica los modelos teóricos de Codd y la primera base de dato relacional que salió al mercado fue Oracle. A partir de aquí el modelo ha sido perfeccionado y actualizado hasta nuestros días.

El Modelo Relacional le permite despreocuparse completamente del complejo almacenamiento de los datos, pues solo los presenta en forma de sencillas tablas, en las cuales solo es necesario especificar las columnas, filas y nombres de tablas para obtener los datos deseados mediante instrucciones SQL. Es decir el Modelo Relacional nos muestra una clara diferencia entre el aspecto físico de los datos y su representación lógica.

Entre las principales funciones de un sistema de administración de bases de datos relacional están:

- Representar los datos como un conjunto de tablas.
- Facilitar el diseño, la visión y la modificación de los datos.
- Usar los valores de los datos para establecer relaciones, brindando máxima flexibilidad.
- Proveer operaciones relacionales como selección, proyección y unión.
- Proveer el acceso más rápido posible a los datos deseados.
- Proteger los datos de usuarios no autorizados con un sofisticado control de seguridad.

- Asegurar la consistencia de los datos con control de transacciones atómicas.
- Asegurar los datos contra fallas de hardware y software con herramientas de recuperación basadas en bitácoras. (Journaling)
- Brinda lenguajes para definición de datos (DDL- Data Definitions Languaje) y para manipulación de datos (DML – Data Manipulation Languaje). Ambos lenguajes utilizan sintaxis similar por lo que se puede considerar uno solo.
- Integración con lenguajes de programación, usando sentencias de manipulación de datos incluidas directamente en programas fuentes.
- Trabajar con diccionarios de datos.
- Proveer fácil acceso a bases de datos remotas.
- Proveer un entorno interactivo para escribir y probar consultas a la base de datos.
- Trasladar las definiciones de datos en los programas y los ubica en la base de datos.
- Proveer funciones para buscar, ordenar y realizar análisis estadísticos.
- Proveer herramientas de diseño y administración.
- Permite definición de datos en forma dinámica; crear tablas, almacenar datos, consultar, deshacer y reintentar.
- Permite cambios dinámicos a las definiciones de la base de datos; agregar, cambiar o eliminar campos, tablas e índices.

2.9.2 Características de las bases de datos relaciónales. Las características más significativas de las bases de datos relacionales son:

- Los datos se encuentran interrelacionados, es decir, existen relaciones lógicas entre ellos.
- Independencia de los datos respecto de los procesos que los explotan.

- Los datos se almacenan sin redundancias: no se guarda información repetida ni redundante.
- Proporcionan mecanismos de seguridad e integridad.

2.9.3 Historia de MySQL. IBM empezó a comercializar en 1.981 el SQL y desde entonces este producto ha tenido un papel importante en el desarrollo de la bases de datos relacionales. IBM propuso y fue aceptada , una versión de SQL al Instituto de Estándares Nacional Americano(ANSI) y desde entonces es utilizado de forma generalizada en las bases de datos relacionales. En 1.983 nació DB2 la más popular(por lo menos en los grandes ordenadores) de las bases de datos de este tipo hasta estos mismos momentos.

En el mundo GNU, una de las bases de datos que se reseña en cualquier referencia de aplicaciones de éste tipo bajo LINUX, es MySQL aunque no está incluida en ninguna distribución ya que no tiene licencia GNU como tal, para comercializarla a ella o a cualquier software que la utilice o se sirva de ésta habrá que adquirir una licencia.

Alrededor de la década del 90, Michael Windenis (monty@analytikerna.se) comenzó a usar mSQL para conectar tablas usando sus propias rutinas de bajo nivel (ISAM). Sin embargo, después de algunos testeos llego a la conclusión que mSQL no era lo suficientemente rápido ni flexible para sus necesidades. De todo esto surgió en una nueva interfaz SQL (claro que con código mas portable) con algunas apariencias similares en la API de C y en los nombres y

funciones de muchos de sus programas. Esto había sido hecho para lograr con relativa facilidad portar aplicaciones y utilidades de MiniSQL a MySQL.

El Origen del nombre MySQL no esta perfectamente claro. Algunos lo atribuyen al hecho de que un gran numero de nuestras librerias y herramientas le asignamos el prefijo “My” por costumbre. Sin embargo la hija de ‘Monty’ es ademas llamada My. Asi que cual de las dos razones da el nombre a MySQL es aun un misterio.

2.9.4 Qué es MySQL. MySQL es un sistema manejador de Bases de Datos relacionales; utilizado para la creación y manipulación de las mismas.

MySQL es un gestor de bases de datos SQL (Structured Query Language). Es una implementación Cliente-Servidor que consta de un servidor y diferentes clientes (programas/librerías) a los que podemos agregar, acceder y procesar datos grabados en una base de datos. Actualmente el gestor de base de datos juega un rol central en la informática, como única utilidad o como parte de otra aplicación.

Es un Sistema de Gestión de Base de Datos Relacional. El modelo relacional se caracteriza a muy grandes rasgos por disponer que toda la información debe estar contenida en tablas, y las relaciones entre datos deben ser representadas explícitamente en esos mismos datos. Esto añade velocidad y flexibilidad.

MySQL es un software de código abierto esto quiere decir que es accesible para cualquiera, para usarlo o modificarlo. Podemos descargar MySQL desde Internet y usarlo sin pagar nada, de esta manera cualquiera puede inclinarse a estudiar el código fuente y cambiarlo para adecuarlo a sus necesidades. MySQL usa el GPL (GNU Licencia Publica General) para definir que podemos y no podemos hacer con el software en diferentes situaciones. Entre otras cuestiones esta licencia aclara que no cuesta dinero a menos que lo incluyamos en un software comercial y tenemos el código fuente.

La base sobre la que está construido MySQL es un conjunto de rutinas que han sido muy demandadas en los ambientes de producción por muchos años. A pesar de seguir en desarrollo MySQL ofrece una alta funcionalidad así como robustez y flexibilidad.

MySQL es muy conocido en el mundo Linux y es una de las Bases de Datos más usadas. Las críticas que recibe suelen venir de las carencias que el producto tiene, algunas de estas son:

1.- Inexistencia de Transacciones

2.- Imposibilidad de hacer subconsultas (Subquerys)

3.- Inexistencia de Procedimientos almacenados

4.- Carencia de Tiggers en las Claves externas

5.- Sin soporte para integridad referencial.

Existen ciertos estándares SQL que también son manejados por MySQL; estas normas vienen dadas por el Instituto Nacional Norteamericano de Normalización (ANSI), que se refieren a la semántica y sintaxis del mismo. Estos estándares están compuestos por un conjunto de sentencias que según su función se dividen en:

LDD (Lenguaje de Definición de Datos, DDL): Creación y modificación de la estructura o esquema de la base de datos.

*CREATE**:*** Crear Objetos de la BD

*ALTER**:*** Modificar Objetos de la BD.

*DROP**:*** Eliminar Objetos de la BD.

LMD (Lenguaje de manipulación de Datos, DML): Consulta y mantenimiento de los datos.

SELECT. Consulta de los datos

INSERT. Inserción de datos.

UPDATE. Modificación de datos.

DELETE. Borrado de datos.

LCD (Lenguaje de Control de Datos, DCL): Seguridad e integridad de los datos.

GRANT. Dar acceso a los Objetos de la BD.

REVOKE. Quitar acceso a los Objetos de la BD.

COMMIT. Confirmación de Transacción.

ROLLBACK. Vuelta atrás de Transacción.

2.9.5 Principales características de MySQL. MySQL es muy rápido, confiable, robusto y fácil de usar tanto para volúmenes de datos grandes como pequeños. Además tiene un conjunto muy practico de características desarrolladas en cooperación muy cercana con los usuarios. Sin embargo bajo constante desarrollo, MySQL hoy en día ofrece un rico y muy útil conjunto de funciones. La conectividad, velocidad y seguridad hace de MySQL altamente conveniente para acceder a bases de datos en Internet. El siguiente es un listado de las principales características de MySQL.

- El principal objetivo de MySQL es velocidad y robustez.
- Escrito en C y C++, testado con GCC 2.7.2.1. Usa GNU autoconf para potabilidad.
- Clientes C, C++, JAVA, Perl, TCL.
- Multiproceso, es decir puede usar varias CPU si éstas están disponibles.
- Puede trabajar en distintas plataformas y Sistemas Operativos distintos.
- Sistema de contraseñas y privilegios muy flexible y segura.
- Todas la palabras de paso viajan encriptadas en la red.
- Registros de longitud fija y variable.
- 16 índices por tabla, cada índice puede estar compuesto de 1 a 15 columnas o partes de ellas con una longitud máxima de 127 bytes.
- Todas las columnas pueden tener valores por defecto.
- Utilidad (Isamchk) para chequear, optimizar y reparar tablas.
- Todos los datos están grabados en formato ISO8859_1.
- Los clientes usan TCP o UNIX Socket para conectarse al servidor.
- El servidor soporta mensajes de error en distintas lenguas.

- Todos los comandos tienen -help o -? Para las ayudas.
- Diversos tipos de columnas como enteros de 1, 2, 3, 4, y 8 bytes, coma flotante, doble precisión, carácter, fechas, enumerados, etc.
- ODBC para Windows 95 (con fuentes), se puede utilizar ACCESS para conectar con el servidor.

Tables

2.9.6 Administración de MySQL (Seguridad). El sistema de seguridad de MySQL garantiza que cada usuario pueda hacer las cosas que le están permitidas.

El sistema decide los diferentes privilegios dependiendo de "QUÉ USUARIO" conectado a "QUÉ BASE DE DATOS" desde "QUÉ HOST". El sistema de privilegios está basado, en el contenido de 5 tablas, **host, user, db, tables_priv, colums_priv** de la base de datos "mysql".

La tabla **user** contiene información sobre los usuarios, desde qué máquinas pueden acceder al servidor MySQL, su clave y sus diferentes permisos. La tabla **host** informa sobre que máquinas podrán acceder al sistema, así como a las bases de datos que tendrán acceso y sus diferentes permisos. Finalmente, las tablas **db, tables_priv, columns_priv** nos proveen de un control individual de las bases de datos, tablas y columnas (campos).

Tabla 2 : MySql

Database: mysql

Db

Host
User
Tables_priv
Colums_priv

Las columnas de las primeras 3 tablas son:

Tabla 3: db

Field	Type	Null	Key	Default	Extra
Host	Char(60)		PRI		
Db	Char(32)		PRI		
User	Char(16)		PRI		
Select_priv	Char(1)			N	
Insert_priv	Char(1)			N	
Update_priv	Char(1)			N	
Delete_priv	Char(1)			N	
Create_priv	Char(1)			N	
Drop_priv	Char(1)			N	

Tabla 4 : Host.

Field	Type	Null	Key	Default	Extra
Host	Char(60)		PRI		
Db	Char(32)		PRI		
Select_priv	Char(1)			N	
Insert_priv	Char(1)			N	
Update_priv	Char(1)			N	
Delete_priv	Char(1)			N	
Create_priv	Char(1)			N	
Drop_priv	Char(1)			N	

Tabla 5 : User.

Field	Type	Null	Key	Default	Extra
Host	Char(60)		PRI		
User	Char(16)		PRI		
Password	Char(16)				
Select_priv	Char(1)			N	
Insert_priv	Char(1)			N	
Update_priv	Char(1)			N	
Delete_priv	Char(1)			N	
Create_priv	Char(1)			N	
Drop_priv	Char(1)			N	

Reload_priv	Char(1)	N
Shutdown_priv	Char(1)	N
Process_priv	Char(1)	N
File_priv	Char(1)	N

He aquí una breve descripción de los diferentes permisos:

Select_priv: Permite utilizar la sentencia SELECT

Insert_priv: Permite utilizar la sentencia INSERT

Update_priv: Permite utilizar la sentencia UPDATE

Delete_priv: Permite utilizar la sentencia DELETE

Create_priv: Permite utilizar la sentencia CREATE o crear bases de datos

Drop_priv: Permite utilizar la sentencia DROP o eliminar bases de datos

Reload_priv: Permite recargar el sistema mediante *mysqladmin reload*

Shutdown_priv: Permite parar el servidor mediante *mysqladmin shutdown*

Process_priv: Permite manejar procesos del servidor

File_priv: Permite leer y escribir ficheros usando comando como SELECT INTO OUTFILE y LOAD DATA INFILE

Grant_priv: Permite otorgar permisos a otros usuarios

Index_priv: Permite crear o borrar índices

Alter_priv: Permite utilizar la sentencia ALTER TABLE

Si se deja en blanco los campos **user**, **host** o **db**, se hace referencia a cualquier usario, servidor o base de datos. Se consigue el mismo efecto poniendo el símbolo **%** en el campo.

Se puede siempre comprobar los privilegios con el script "mysqlaccess".

Un HOST debe ser un "host local", un numero IP, o una expresión SQL . Si en la tabla "db" la columna host está vacía significa "cualquier host" en la tabla de "host". Si en la tabla "host" o "user" la columna host está vacía significa que cualquier HOST puede crear una conexión TCP con el servidor.

Una columna "USER" vacía significa cualquier nombre de usuario.
Los privilegios predefinidos en Windows les dan privilegios full a todos los usuarios locales a todas las bases de datos; ya que en la instalación se crea el usuario **root** sin password.

Se deben crear distintos usuarios con distintos permisos. Entre ellos, el usuario administrador de **MySQL**, con todos los permisos, y como recomendación de seguridad, el usuario **nobody** sólo con el permiso de ver (SELECT).

Se debe colocar una contraseña para todos los usuarios y quitar la fila en la tabla de mysql.user que tiene Host = ' localhost ' y User = ' ' .

También se debe agregar una contraseña para el usuario root.

(El siguiente ejemplo empieza quitando al usuario anónimo que permite a cualquiera para acceder a la base de datos 'Test'.):

C:\mysql\bin\mysql mysql

mysql> DELETE FROM user WHERE Host='localhost' AND User='';

mysql> QUIT

C:\mysql\bin\mysqladmin reload

C:\mysql\bin\mysqladmin -u root password your_password

Después que se digitado la contraseña, si se desea bajar el servidor de mysqld, se puede hacer usando esta orden así:

mysqladmin -u root -p your_password shutdown

2.9.7 Estructura de MySQL. Una vez instalado en Linux el gestor de bases de datos tendremos los siguientes ficheros y directorios:

En Linux

Directorios:

- bin
- data
- include
- lib
- mysql-test
- scripts

- share
- sql-bench
- suport-files
- tests

Ficheros:

- COPVING
- COPVING.LIB
- ChangeLog
- INSTALL-BINARY
- README
- Configure
- manual.html
- manual_toc.html

En los ficheros README, INSTALL-BINARY, manual.html, etc., viene información sobre la instalación del servidor de bases de datos, manual de funcionamiento, etc. que es bastante completo y efectivo.

En Windows:

Carpetas:

- bench
- bin

- data
- Docs
- examples
- include
- lib
- scripts
- share

Archivos:

- infolist
- my-example
- mysqlbug
- Readme
- Uninst.isu

Dentro del directorio **/data** encontraremos como subdirectorios de éste, cada una de las bases de datos que se van creando. En el momento de la instalación se define por defecto los archivos en los que se apoya el sistema de seguridad, esta base de datos es "mysql". En esta base de datos es donde se guardarán todos los permisos y restricciones a los datos de nuestras bases de datos.

En el directorio **/bench** encontraremos ejemplos de SQL.

El directorio **/share** contiene los mensajes de error del servidor en cada uno de los idiomas que está disponible.

Los directorios **/include** y **/lib** se encuentran los ficheros *.h y las librerías necesarias.

El directorio **/bin** están todos los ficheros ejecutables, entre los más importantes destacaremos:

'mysql'

Una Shell de SQL (con readline de GNU). Se puede usar tanto interactivamente como no.

'mysqladmin'

Utilidades de administración. Crear/borrar base de datos. Información sobre procesos y versiones.

'mysqld'

El SQL "deamon" (“demonio”). Debe estar siempre ejecutándose.

'mysqlshow'

Visualiza información sobre base de datos, tablas y campos.

'safe_mysqld'

Arranca "mysqld".

'mysqlaccess'

Script para chequear los privilegios de una combinación: Host, Usuario y base de datos.

'mysqlbug'

Se utiliza para enviar los posibles errores (bug) que se encuentren en el gestor.

'mysql_install_db'

Crear grandes tablas con privilegios por defecto, se ejecuta cuando se instala por primera vez en un sistema nuevo.

'isamchk'

Chequea, optimiza y repara tablas.

La principal herramienta de MySQL es **mysqladmin**, la cuál como parece indicar su nombre es la encargada de la administración.

MySQL crea por defecto al usuario **root** con todos los permisos posibles habilitados, se puede utilizar este usuario como administrador o crear otro, por ejemplo **mysqladmin**. Como el usuario **root** lo crea sin clave de acceso, lo primero que se debe realizar es asignarle una:

mysqladmin -u root -p password "miclave"

A partir de ahora cualquier operación que se realice como root se deberá especificar la clave. Hay que destacar que entre el modificador -p y la clave no debe haber espacios.

mysqladmin -u root –pmiclave

Pues bien, ya se ha preparado para crear una base de datos

mysqladmin -u root -pmiclave create mibasededatos

Para borrarla:

mysqladmin -u root -pmiclave drop mibasededatos

Para cada base de datos que nosotros se creen, MySQL crea un directorio con el nombre que se le ha asignado a la base de datos. Dentro de este directorio, por cada tabla que se define MySQL va ha crear tres archivos: **mitabla.ISD, mitabla.ISM, mitabla.frm**

El archivo con extensión **ISD**, es el que contiene los datos de la tabla, el **ISM** contiene información acerca de las claves y otros datos que MySQL utiliza para buscar datos en el fichero **ISD**. Y el archivo **frm** contiene la estructura de la propia tabla.

Dado que las bases de datos de MySQL son simples ficheros de un directorio, para realizar copias de seguridad, se pueden utilizar las herramientas de compresión que habitualmente usamos en nuestro sistema y luego copiarlo a otro lugar, o simplemente esto último.

2.10 ODBC (OPEN DATA BASE CONNECTIVITY)

Para el envío de solicitudes y respuestas entre Clientes y Servidores, se ha establecido un estándar que permite comunicarse con cualquier base de datos a través de una interfaz

común llamado ODBC; ODBC tiene como misión proporcionar acceso a bases de datos de datos relacionales utilizando como sostén el lenguaje SQL; este manejador (ODBC) posee un código que contiene los aspectos específicos de una base de datos en particular y proporciona acceso a ella mediante un conjunto estándar de llamadas API.

Microsoft introdujo la especificación del ODBC con el fin de que los programas fueran totalmente independiente del servidor de base de datos y que luego en tiempo de ejecución, una DLL genérica resolviera automáticamente todas las conversiones necesarias.

Los Sistemas de Gerencia de Base de Datos soportados por los ODBC actualmente en el mercado son muy diferentes unos de otros: sistemas relacionales, archivos *.DBF, archivos propietarios del Btrieve, etc. Generando con esto algunos problemas de compatibilidad y soluciones ineficientes ya que cada Sistema de Gerencia de Base de datos tiene sus particularidades que deben ser tenidas en cuenta e incluso, explotadas para obtener las soluciones realmente eficientes.

Una solución dada podría ser: utilizar el ODBC SQL como un producto de uso general para comunicar los programas generados para el Cliente con el Servidor correspondiente de la manera que sea más adecuada en cada caso, generando para ello el código especifico necesario.

Por medio de la interfaz de programa de aplicación (API. Application Program Interface) se realiza el envío de solicitudes y respuestas entre los Clientes y el servidor. A continuación se mencionarán dos tipos de APIs: APIs incorporadas: las instrucciones de tipo SQL se

incorporan en el código de la aplicación. APIs a nivel de llamados: el programador puede emitir instrucciones de SQL utilizando el lenguaje huésped.

El APIs de ODBC esta conformado por una serie de librerías dinámicas (DLL) que reciben el nombre de drivers ODBC los cuales permiten:

La gestión de fuentes de datos. Entendiéndose por fuente de datos, una conexión a una base de datos a la que se le da un nombre, es necesario tener un drivers ODBC para cada base de datos a la que se quiera conectar una determinada aplicación.

Gestión de la comunicación SQL. El manejador ODBC también es el encargado de gestionar el envío de las consultas SQL, así como los resultados entre el servidor y las aplicaciones.

Se puede decir entonces que el API de ODBC utiliza un conjunto de funciones estándar las cuales permiten realizar distintas operaciones, como consultas SQL, al lenguaje del servidor particular con el que se esta trabajando.

Antes de la aparición del estándar ODBC el sistema que se utilizaba era el SQL embebido: este permitía colocar sentencias SQL en programas creados en otros lenguajes de programación. El desarrollador implica una interfaz (aplicación) con una serie de comandos SQL precompilados e invariables, pero no puede disponer de los datos del servidor desde otras aplicaciones.

La necesidad de ampliar el número de Clientes susceptibles de utilizar las aplicaciones sobre diversas plataformas utilizando diversas herramientas es en gran parte lo que hace necesario

el planteamiento del ODBC, como también la posibilidad de vincular dinámica y directamente datos con hojas de calculo, procesos de texto, etc. Independientes de su fabricante.

Microsoft fue quien introdujo la especificación del ODBC como se mencionó anteriormente, en ese momento ya existían sistemas de acceso a bases de datos relacionales, donde dicho acceso estaba comprometido directamente con los fabricantes de hardware y software. Con ODBC se logro que los programas resultaran independientes del Servidor de base de datos y que luego en tiempo de ejecución, una DLL genérica resolviera automáticamente todas las conversiones necesarias.

2.10.1 Qué se debe tener en cuenta

- A cuantos bits esta trabajando el equipo donde se desea instalar.
- Bajo qué plataforma se está trabajando (Sistema Operativo).
- Espacio suficiente (55Megas).
- A cuántos bits trabaja la herramienta que se quiere comunicar con la base de datos.
- La versión de la base de datos a la cual se quiere conectar.
- La versión SQLNET que se está utilizando.

2.10.2 Usos de ODBC. continuación se mencionarán los diferentes usos que se le pueden dar a ODBC.

- ODBC es útil cuando deseamos crear aplicaciones en las cuales se utilizan bases de datos independientes basadas en archivos para el desarrollo que después se unirán a un modelo Cliente/Servidor, evitando así realizar grandes cambios a nivel de código.

- Permite la transportación de aplicaciones ya existentes de un lenguaje a otro.

- Cuando se va a crear una aplicación que necesite ser compatible con datos ya existentes en alguna base de datos.

- El sistema empleado para acceder modularmente a diversos dispositivos viene siendo el driver. Microsoft plantea un tipo de driver que se comunica con una base de datos en vez de hacerlo con un dispositivo periférico de hardware.

- Para realizar un enlace, se define un conjunto de funciones comunes (API de ODBC) que cada sistema debe implementar con sus instrucciones propias para compatibilizarse con el estándar.

2.10.3 Arquitectura de ODBC. as capas que intervienen en la realización de la conexión son las siguientes:

Aplicación

Se encarga de realizar las llamadas a la API de ODBC para enviar las instrucciones SQL necesarias y así obtener los resultados esperados.

Administrador de drivers ODBC

Realiza dos tareas que corresponden al modo de funcionamiento oculto que realiza la carga de los drivers necesarios por petición de cada aplicación y el modo de funcionamiento mediante interfaz que se encarga del mantenimiento de drivers y fuentes de datos.

Drivers

Recibe las llamadas de la aplicación hacia funciones de la API de ODBC y las traduce al lenguaje nativo del Servidor especifico.

Fuentes de datos

Contiene la información asociada en el servidor de la base de datos destino, se debe tener en cuenta los siguientes aspectos, el *Sistema Operativo* en el que se encuentra el Servidor, el *driver* que se utiliza, el *tipo de sistemas gestor y la plataforma de red* a la que se necesita acceder para conectarse a él.

Servidor

Es el Sistema Gestor de la Base de Datos Relacional a la que se desea conectar.

2.10.4 Niveles de conformidad de API ODBC. La API ODBC define tres niveles de conformidad, es decir, tres tipos de drivers en función de los servicios que ofrecen a las aplicaciones.

Core

Ofrece conexión a la base de datos, preparación y ejecución de sentencias SQL, recepción de conjuntos de resultados, gestión de transacciones y obtención de información sobre errores.

Leve 1

Además del nivel anterior, ofrece conectividad a fuentes de datos utilizando cuadros de diálogo diseñados específicamente por el driver ODBC, como también capacidades para obtener información sobre catálogos y fuentes de datos.

Leve 2

Es similar al nivel anterior, pero además proporciona capacidades de acceso a las fuentes de datos disponibles para una aplicación Cliente, obtención de conjuntos de resultados en formato array, uso de cursores más sofisticados y obtención de información adicional de catálogos como privilegios, claves y lista de procesamiento almacenados.

2.10.5 Fuentes de datos ODBC. ara que una aplicación pueda acceder a los datos almacenados en un Servidor a través de ODBC, se debe crear una fuente de datos ODBC. Se distinguen tres tipos de fuentes de datos ODBC en función del origen de datos y de su accesibilidad:

Fuentes de datos de usuario

Este tipo de fuentes de datos solo son accesibles a la cuenta de usuario de red de Windows NT o Windows 95 que las creó. No son muy apropiadas para ser utilizadas por aplicaciones que se ejecuten como un servicio que no inicie como una cuenta de usuario sino de sistema.

Fuentes de datos de sistemas

Este tipo es visible por cualquier cuenta de usuario en la máquina en la que se han creado.

Fuentes de datos de fichero

Este tipo de fuentes se almacenan en ficheros en el Cliente, más no en el registro del Sistema Operativo.

La configuración de estas fuentes de datos se realiza a partir del proceso de creación de un DSN el cual contiene parámetros de la cadena de conexión e información adicional proporcionada por el desarrollador cuando se instala. Una vez registrado, el DSN incluye el nombre del origen de datos, el nombre del Servidor, el tipo de controlador y otros datos sobre la interfaz de red y la configuración de la seguridad. Es posible configurar los DSN de diversas formas, entre las que se incluye la aplicación ODBC del panel de control, la utilización por programas de llamadas de la API y la simple copia de archivos. Algunas técnicas de conexión no utilizan los DSN en absoluto.

3. MARCO CONCEPTUAL

C.I.O.H

CENTRO DE INVESTIGACIÓN OCEANOGRAFICAS E HIDROGRAFICAS

SOFTWARE

Software es el conjunto de instrucciones que las computadoras emplean para manipular datos (Los Programas) Sin el software, la computadora sería un conjunto de medios sin utilizar. Al cargar los programas en una computadora, la máquina actuará como si recibieran a una educación instantánea; de pronto "sabe" cómo pensar y cómo operar.

Software es un conjunto de programas, documentos, procedimientos, y rutinas asociados con la operación de un sistema de computo.
Distinguiéndose de los componentes físicos llamados hardware.

Comúnmente a los programas de computación se les llama software; el software asegura que el programa o sistema cumpla por completo con sus objetivos, opera con eficiencia, esta adecuadamente documentado, y es suficientemente sencillo de operar.

Es simplemente el conjunto de instrucciones individuales que se le proporciona al microprocesador para que pueda procesar los datos y generar los resultados esperados.

ORDENADOR

Es el dispositivo electrónico capaz de recibir un conjunto de instrucciones y ejecutarlas realizado cálculos sobre los datos numéricos, o bien compilando y correlacionado otros tipos de información.

SERVIDOR

Un servidor es un ordenador de gran potencia que se encarga de “prestar un servicio” a otros ordenadores (por ejemplo, el suyo) que se conectan a él. Varios servidores juntos, es decir conectados entre sí forman una red IRC.

Estas máquinas se encargan de hacer que los mensajes que usted escriba lleguen a su destino y también de hacer llegar hasta usted, los mensajes del resto de usuarios. En definitiva, se ocupan de gestionar todo el tráfico de información que se produce en la red IRC.

CLIENTE

Un cliente es un consumidor de los servicios del servidor; en otras palabras, un cliente utiliza espacio en disco, impresora o MODEM proporcionando por el servidor.

BASES DE DATOS

Una base de datos no es más que un conjunto de archivos con unas características propias que los hacen especiales. Entre estas características se puede destacar su facilidad para almacenar datos de diversos formatos en un mismo archivo y su ordenación si se desea.

PROCESAMIENTO

Sometimiento de algo a un proceso de elaboración, transformación, de tal manera que se obtengan al final los resultados esperados.

PHP

PHP es el acrónimo de Hipertext Preprocesor. Es un lenguaje de programación del lado del servidor gratuito e independiente de plataforma, rápido, con una gran librería de funciones y mucha documentación.

Un lenguaje del lado del servidor es aquel que se ejecuta en el servidor web, justo antes de que se envíe la página a través de Internet al cliente. Las páginas que se ejecutan en el servidor pueden realizar accesos a bases de datos, conexiones en red, y otras tareas para crear la página final que verá el cliente. El cliente solamente recibe una página con el código HTML resultante de la ejecución de la PHP. Como la página resultante contiene únicamente código HTML, es compatible con todos los navegadores.

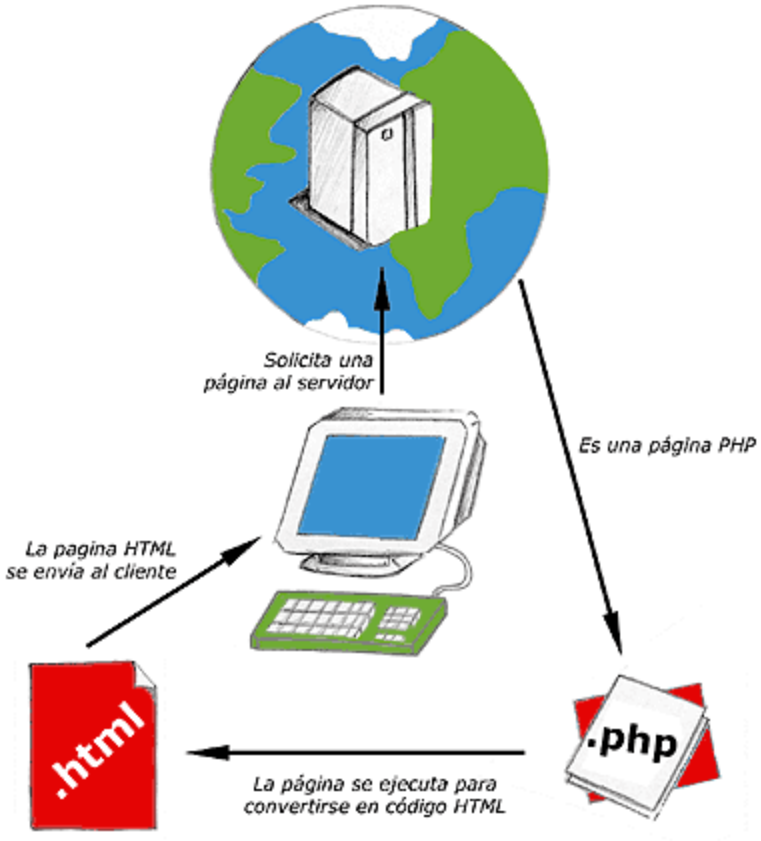

Figura 3: Esquema del funcionamiento de las páginas PHP

4. APLICACIONES CLIENTE SERVIDOR EN VISUALFOX PRO

VISUAL FOXPRO.

Una aplicación Cliente/Servidor de Visual FoxPro combina la eficiencia, la velocidad, la interfaz gráfica de usuario y las sofisticadas funciones de consulta, informes y proceso de Visual FoxPro con el acceso multiusuario, almacenamiento masivo de datos, seguridad incorporada, robusto proceso de transacciones, inicio de sesiones y la sintaxis nativa del Servidor de un origen de datos o Servidor ODBC.

En la actualidad es mucho más sencillo realizar conexiones a bases de datos remotas por medio de ODBC, pues Visual FoxPro permite no sólo realizar llamadas explícitas a funciones

de ODBC, sino crear vistas locales sobre tablas remotas, lo que supone que se puede trabajar con tablas de MySQL prácticamente como si fueran tablas de Visual FoxPro.

Mediante el uso de ODBC, es posible realizar consultas desde Aplicaciones Visual FoxPro usando el lenguaje SQL nativo del origen de datos al cual se está conectando, como también llamar procedimientos almacenados que pueden contener parámetros tanto de entrada como de salida, así como valores de retorno.

Visual FoxPro utiliza toda la potencia de ODBC integrando en su versión estándar todas las funcionalidades para el acceso a estos drivers, contiene un conjunto de drivers ODBC para conectarse a Servidores SQL Server, MySQL entre otros.

4.1 CONSULTAS DE PASO A TRAVÉS O SQL

Se puede mejorar una aplicación mediante la tecnología de paso a través de SQL para crear objetos en el Servidor, ejecutar procedimientos almacenados de Servidor y ejecutar comandos con la sintaxis nativa del Servidor.

La tecnología de paso a través de SQL proporciona acceso directo a un Servidor remoto con las funciones de paso a través de SQL de Visual FoxPro. Estas funciones facilitan acceso y control adicional del Servidor. El paso a través de SQL es la mejor herramienta para crear conjuntos de resultados de solo lectura y para utilizar cualquier otra sintaxis nativa de SQL.

Se pueden crear cursores mediante la tecnología de paso a través de SQL. Los cursores creados solo existen durante la sesión actual.

Mediante consultas de paso a través de SQL se pueden ejecutar consultas utilizando ODBC, de tal manera que éstas no sean procesadas localmente, dejándole ésta tarea al motor de base de datos con cual se esta conectando, como se muestra en la figura 4.

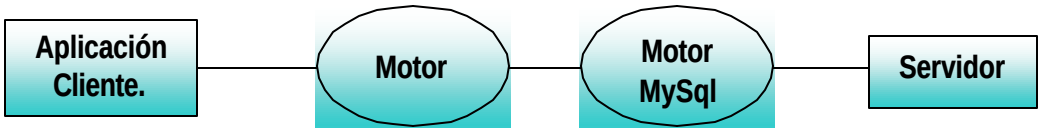

Figura 4: Consultas de paso a través de SQL.

4.1.1 Funciones SQL de Visual FoxPro. A continuación se muestran algunas de las funciones SQL de Visual FoxPro que admiten el trabajo con orígenes de datos remotos, agrupados según la tarea.

Tabla 6 : Funciones SQL de Visual FoxPro.

Tarea	Función	Descripción
Administración de conexiones	SQLCONNECT ( )	Realiza una conexión con el origen de datos para operaciones de paso a través de SQL
	SQLDISCONNECT ( )	Rompe una conexión a un origen de datos ODBC.

Control y ejecución de instrucciones SQL.	SQLEXEC ( )	Ejecutar una instrucción de paso a través de SQL en una conexión activa.
	SQLCOMMIT ( )	Confirma los cambios realizados.
	SQLROLLBACK ( )	Deshace los cambios realizados.
Control Vario	SQLGETPROP ( )	Obtiene una propiedad de conexión de una conexión activa.
	SQLSETPROP ( )	Establece una propiedad de una conexión activa, es decir, cambia valores de la configuración.

Para conectarse a un origen de datos ODBC remoto mediante el paso a través de SQL se debe llamar la función SQLCONNECT ( ) de Visual FoxPro para crear una conexión; una vez realizado ésto, se usan las funciones de paso a través de SQL para enviar comandos al origen de datos remotos para su ejecución.

Para usar funciones de paso a través de SQL de visual FoxPro se debe tener en cuenta lo siguiente:

- Confirmar que el sistema puede conectar el equipo al origen de datos.
- Establecer una conexión con su origen de datos con las funciones SQLCONNECT ( ) o SQLSTRINGCONNECT ( ).

- Usar funciones de paso a través de SQL de visual FoxPro para obtener datos en cursores de Visual FoxPro y procesar la obtención de datos comandos y funciones estándar de Visual FoxPro.
- Desconectarse del origen de datos con la función SQLDISCONNECT ( ).

4.1.1.1 SQLCONNECT ( ). La función SQLCONNECT establece una conexión entre un Cliente y un Servidor (BASE de Datos Remota), mediante un origen de datos (DSN). La sintaxis que utiliza esta función es la siguiente:

SQLCONNECT ([NombreOrigenDatos, cIdUsuario, Contraseña I NombreConexión])

Donde NombreOrigenDatos hace referencia al nombre del DSN creado para realizar la conexión; cIdUsuario especifica un identificador de usuario para iniciar la sesión en el origen de datos; contraseña, especifica la contraseña para el origen de datos; y cNombreConexión, especifica el nombre de una conexión creado con CREATE CONNECTION.

SQLCONNECT ( ) devuelve un controlador numérico positivo distinto de cero si consigue conectar con el origen de datos. Este valor numérico se almacena en una variable de memoria que recibe el nombre de gnConn la cual se utiliza en posteriores llamadas de función en las que se necesite un controlador de conexión para verificar si se realizó la conexión o no. SQLCONNECT ( ) devuelve valores menores o iguales que 0 si no puede establecerse la conexión.

Si se ejecuta SQLCCONNECT ( ) sin ningún de sus argumentos adicionales, se mostrará el cuadro de diálogo *Seleccionar conexión u Origen de datos*, el cual permite elegir un origen de datos.

Ejemplo:

```
STORE SQLCONNECT (“DSNMySql”, “rootlq”, “Laboratorio”) TO gnConn
IF gnConn <= 0
  =MESSAGEBOX(“No se puede realizar la conexión”, 16, “Error de Conexión SQL”)
ELSE
 =MESSAGEBOX(“Conexión realizada”, 48, “Mensaje de conexión SQL”)
SQLDISCONNECT(gnConn)
ENDIF
```

4.1.1.2 Sldisconnect ( ). Termina una conexión con un origen de datos. La sintaxis que se utiliza para esta función es la siguiente:

```
SQLDISCONNECT ( nControladorConexión)
```

Donde nControladorConexión especifica el controlador de conexión con el origen de datos que devuelve SQLCONNECT ( ). Si se desea terminar todas las conexiones activas, se especifica 0 para nControladorConexión.

SQLCONNECT devuelve un controlador de datos numérico 1 si se termina la conexión, -1 si hay un error de nivel de conexión y –2 si hay un error de nivel de entorno.

Si se ejecuta SQLDISCONNECT ( ) en una secuencia de función asíncrona o durante una transacción, SQLDISCONNECT ( ) generará un error.

Ejemplo:

```
STORE SQLCONNECT (“DSNMySql”, “rootlq”, “Laboratorio”) TO gnConn
IF gnConn <= 0
  =MESSAGEBOX(“No se puede realizar la conexión”, 16, “Error de Conexión SQL”)
ELSE
 =MESSAGEBOX(“Conexión realizada”, 48, “Mensaje de conexión SQL”)
SQLDISCONNECT(gnConn)
ENDIF
```

4.1.1.3 SQLEXEC ( )

La función SQLEXEC ( ) permite enviar una instrucción SQL al origen de datos, donde se procesa la instrucción. En el caso más sencillo, cualquier cadena que escriba en el segundo parámetro de la función SQLEXEC ( ) se pasa al origen de datos sin interpretación. Esto le permite ejecutar cualquier instrucción mediante el SQL nativo del origen de datos. La sintaxis que se utiliza para esta función es la siguiente:

SQLEXEC (nControladorConexión, [cComandoSQL, [NombreCursor]])

Donde nControladorConexión especifica el controlador de conexión con el origen de datos que devuelve SQLCONNECT ( ); cComandoSQL especifica la instrucción SQL transferida al

origen de datos; NombreCursor especifica el nombre del cursor de Visual FoxPro al que se envía el conjunto de resultados. Si no incluye un nombre de cursor, Visual FoxPro utiliza el nombre predeterminado SQLRESULT.

También puede usar la función SQLEXEC ( ) para crear una consulta parametrizada o pasar extensiones SQL al origen de datos. La instrucción SQL puede contener una cláusula parametrizada WHERE que crea una vista parametrizada. Todos los parámetros de la cláusula WHERE se deben definir antes de ejecutar SQLEXEC ( ). Por ejemplo si los parámetros son variables de memoria, éstas deberán crearse e inicializarse antes de ejecutar SQLEXEC ( ).

Para múltiples conjuntos de resultados, se obtienen nuevos nombres de cursor al agregar un número al nombre del primer cursor.

SQLEXEC ( ) devuelve el número de conjuntos de resultado si hay más de uno; devuelve 0 si sigue ejecutándose y devuelve 1 cuando ha acabado de ejecutarse. Devuelve –1 si se produce un error de nivel de conexión.

Si se utiliza SQLEXEC ( ) para ejecutarse una instrucción SQL preparada con SQLPREPARE ( ), solo se requerirá el argumento nControladorConexión del controlador de conexión.

Si la instrucción SQL genera un conjunto de resultados, SQLEXEC ( ) almacena el conjunto de resultados en el cursor especificado de Visual FoxPro. Si la instrucción SQL genera dos o más conjuntos de resultados y se establece SQLSETPROP ( ) como 1 (modo por lotes),

podrá asignar un nombre a cada conjunto de resultados al establecer la opción SQLSETPROP ( ) BatchMode como 0 y cambiar el nombre del cursor cada vez que llame a SQLMORERESULTS ( ).

SQLEXEC ( ) es una de las cuatros funciones que puede ejecutarse en modo síncrono o asíncrono. La configuración asíncrona de SQLSETPROP ( ) determina si estas funciones se ejecutan en modo sincrónico o asíncrono. En modo asíncrono, deberá llamar a SQLEXEC( ) repetidamente mientras devuelve un valor distinto de 0 (sigue ejecutándose).

Ejemplo:

```
STORE SQLCONNECT (“DSNMySql”, “rootlq”, “Laboratorio”) TO gnConn
IF gnConn <= 0
  =MESSAGEBOX(“No se puede realizar la conexión”, 16, “Error de Conexión SQL”)
ELSE
        =MESSAGEBOX(“Conexión realizada”, 48, “Mensaje de conexión SQL”)
       SQLEXEC(gnConn, “SELECT * FROM análisis”,”cAnalisis”)
       SQLDISCONNECT(gnConn)
ENDIF
```

4.1.2 Crear una consulta parametrizada. Para cear una consulta parametrizada con paso a través de SQL, se colbca un signo de cierre de interrogación (?) antes de un parámetro de Visual FoxPro y a continuación, se incluye el parámetro en una cadena SQL que se envía con SQLEXEC( ).

El parámetro que se suministra se evalúa como expresión de Visual FoxPro y el valor se envía como parte de la instrucción SQL de la vista. Si la evaluación falla, Visual FoxPro le pide el valor del parámetro.

Si el parámetro es una expresión, se escribe la expresión del parámetro entre paréntesis. Esto asegura que toda la expresión se evalúa como parte del parámetro.

Los orígenes de datos ODBC no aceptan parámetros en las siguientes ubicaciones:

- En una lista de tablas o campos SELECT
- Como las dos expresiones de un predicado de comparación.
- Como los dos operandos de un operador binario.

4.1.3 Devolver valores de parámetros. Los parámetros de entrada / salida solo están disponibles después de que se ha buscado el último conjunto de resultados de una instrucción. Es decir los valores de entrada / salida se devuelven a Visual FoxPro solo después de que:

SQLEXEC ( ) devuelva (1) en modo de proceso por lotes

SQLMORERESULTS ( ) devuelva (2) en modo de no-proceso por lotes.

Si la instrucción SQLEXEC ( ) solicita múltiples conjuntos de resultado, los parámetros de salida sólo estarán disponible con seguridad después de que se haya buscado el último conjunto de resultados en el origen de datos.

4.1.4 Usar extensiones ODBC de SQL. Puede usar SQLEXEC ( ) para ejecutar extensiones ODBC de SQL si se encierra la instrucción SQL con sintaxis de escape estándar o extendida de grupo de acceso SQL.

Se puede usar paso a través de SQL para realizar combinaciones externas en datos remotos con la sintaxis de escape ODBC, si el servidor admite combinaciones externas. Una combinación externa combina información de una o más tablas independientemente de si se han encontrado filas coincidentes.

4.1.5 Administrar conexiones con paso a través de SQL. Cuando se crea una vista remota se debe elegir un nombre de origen de datos ODBC o un nombre de conexión que se utilizará como canalización para el Servidor remoto, una vez activada la vista. Para tener acceso a datos remotos directamente como paso a través de SQL, es necesario disponer del controlador para una conexión activa. Un controlador es un valor que hace referencia a un objeto; en este caso el controlador hace referencia a una conexión de origen de datos. Para obtener un controlador se debe solicitar una conexión con el origen de datos utilizando la función SQLCONNECT ( ). Si la conexión se realiza correctamente, la aplicación recibirá un controlador para su uso en llamadas posteriores de Visual FoxPro.

La aplicación puede solicitar múltiples conexiones para un origen de datos. También puede trabajar con múltiples orígenes de datos ODBC solicitando una conexión con cada origen de datos al que se desea tener acceso. Si desea reducir el número de conexiones utilizadas,

puede configurar vistas remotas para compartir la misma conexión. Puede desconectarse de un origen de datos mediante la función SQLDISCONNECT ( ).

Visual FoxPro se basa en la definición del origen de datos ODBC almacenada en el archivo Odbc.ini de Windows o en el registro de Windows NT para conectarse a un origen de datos. Si se cambian el nombre de la información de inicio de sesión para un origen de datos, se debe tener presente que estos cambios pueden influir en que una aplicación que utilice ese origen de datos pueda conectarse a no al servidor remoto deseado.

4.2 DEFINICIÓN DE CONEXIONES

4.2.1 Create connection. Visual FoxPro posee la posibilidad de utilizar los servicios de ODBC a través del entorno y no sólo por medio de comandos y funciones. Ejemplo de ésto es la posibilidad de crear conexiones predefinidas por medio del Diseñador de Conexiones. Para ejecutarlo se debe tener abierta una base de datos (DBC), luego se escribe CREATE CONNECTION.

En el Diseñador de Conexiones se puede definir todas las características de una conexión a base de datos por medio de ODBC de forma sencilla y sin necesidad de utilizar las funciones

SQLSETPROP y SQLGETPROP para su configuración. Esta conexión se almacena dentro de la base de datos y está disponible siempre que se abra la base de datos en cuestión.

4.2.2 Controlar propiedades de entorno y conexión. El entorno Cliente/Servidor se establece cada vez que se abre Visual FoxPro. El entorno existe para esa sesión y desaparece al cerrar Visual FoxPro. El entorno Cliente/Servidor contiene:

- Propiedades globales que actúan como prototipo para las nuevas conexiones.
- Valores de error para los errores que ocurren fuera de la conexión especificada.

La función SQLSETPROP ( ) sirve para controlar el valor de la propiedad predeterminada en el entorno de conexión y de las propiedades dentro de conexiones individuales. Los métodos empleados para introducir valores SQLSETPRO ( ) son coherentes para las conexiones de entorno e individuales.

Las propiedades especificas con uno de los dos valores posibles pueden emplear un valor lógico (.F. o .T.) para eExpresión.

Los nombres de propiedades se pueden abreviar hasta la forma más breve que no dé lugar a ambigüedades. Por ejemplo, puede utilizar “Asynchronous”, “Asynch” o “A” para especificar la propiedad Asynchronous. Los nombres de propiedades no distinguen entre mayúsculas y minúsculas. Cuando se inicia una conexión, ésta hereda valores predeterminados para las propiedades de la conexión. Puede usar SQLSETPROP ( ) para modificar estos valores.

4.2.3 Establecer propiedades de conexión. Para ver el valor actual de propiedades para una conexión, se utiliza SQLGETPROP ( ) con el controlador de conexión adecuado.

La siguiente tabla muestra las propiedades de conexión a las que se puede tener acceso con SQLGETPROP ( ).

Tabla 7 : Propiedades de conexión de SQLGETPROP ( )

PARA	USE	OBJETIVO
Mostrar la informaci Utilizada para crear la Conexión activa.	ConnectString	La cadena de conexión de inicio de sesión.
	DataSource	El nombre del origen de datos, según lo ha definido ODBC.
	Password	La contraseña de conexión
	UserID	La identificación del usuario.
Trabaja con conexiones Compartida	ConnectBusy	Verdadero (T)si una conexión compartida está ocupada; falso si no, lo esta (F)
Controlar la apariencia de la interfaz	DispLogin	Controla cuándo se muestra el cuadro de diálogo Inicio de sesión de ODBC.
	DispWarnings	Controla cuándo se muestra o no los mensajes de error.
Controlar los intervalos de tiempo	ConnectTimeout	Especifica el tiempo (seg.) que hay que esperar antes de devolver un error de fin de tiempo de espera para la conexión.

	IdleTimeout	Especifica el intervalo de tiempo de espera inactivo (seg). Las conexiones activas en proceso de cualificación se desactivan tras el intervalo de tiempo especificado.
	WaitTime	Controla la cantidad de tiempo en milisegundos que transcurre antes de que Visual FoxPro compruebe si la instrucción SQL ha terminado de ejecutarse.
	Query Timeout	Controla el tiempo (seg). Que hay que esperar antes de devolver un error general de fin de tiempo de espera.
Administrar transacciones.	Transactions	Determina la forma en que la conexión administra las transacciones en la tabla remota.
Controlar la recopilación de conjuntos de resultados en los cursores de presentación	Asynchoronous	Especifica si los conjuntos de resultados se devuelven de forma síncrona (valor predeterminado) o asíncrona.
	BathMode	Especifica si SQLEXEC ( ) devuelve todos los conjuntos de resultados a la vez (valor predeterminado) o individual con SQLMORERESULTS ( ).
	PacketSize	Especifica el tamaño del paquete de red

		utilizado por la conexión.
Mostrar controladores ODBC internos.	ODBChdbc[2]	El controlador interno de conexión ODBC que pueden utilizar los archivos de biblioteca Externas (archivos .fll) para llamar a las funciones API de ODBC.
	ODBChstmt[2]	El controlador interno de instrucciones ODBC que pueden utilizar los archivos de bibliotecas externas (archivos .fll) para llamar a las funciones API de ODBC.

4.3 ACTUALIZAR DATOS REMOTOS CON PASO A TRAVÉS DE SQL.

Al utilizar funciones de paso a través de SQL para actualizar datos en un Servidor remoto, se controla si se desea actualizar los datos, así como detalles específicos acerca de las actualizaciones, estableciendo propiedades en el cursor del conjunto de resultados. Visual FoxPro comprueba estas propiedades cuando se solicita una actualización antes de realizarla.

Para actualizar datos remotos es necesario definir cinco propiedades: Tables, keyFieldlist, UpdatenameList, UpdatableFieldList y SendUpdates. Se puede especificar propiedades

adicionales, tales como Buffering, UpdateType y WhereType para adecuarse mejor a los requisitos de la aplicación.

Para permitir actualizar en un cursor de vista activo se utiliza la función CURSORSETPROP( ) para especificar las propiedades de actualización del cursor de vista: Tables, keyFieldlist, UpdatenameList, UpdatableFieldList y SendUpdates.

Los cursores de vista de paso a través de SQL no admiten actualizaciones hasta que no especifique propiedades de actualización para el cursor de vista. Si se desea almacenar los valores de las propiedades de actualización de forma definitiva, se debe crear una definición de vista. Visual FoxPro suministra valores predeterminados que preparan la vista para que sea actualizable cuando se crea mediante el Diseñador de vistas o el lenguaje. Se puede utilizar la función CURSORSETPROP ( ) con el fin de agregar información adicional para personalizar o aumentar la cantidad de valores predeterminados.

Las propiedades de actualización que se establecen en el cursor de vista activo tienen nombres ligeramente distintos que sus correspondientes en DBSETPROP ( ).

4.3.1 Detectar cambios realizados por otros usuarios. En aplicaciones multiusuarios, los conflictos con las actualizaciones de otros usuarios se detectan mediante la consulta SQL Update, que se genera cuando se intenta realizar una escritura localmente. El nivel de detección depende del valor de la propiedad WhereType.

4.3.2 Imponer actualizaciones. Se puede utilizar la función TABLEUPDATE ( ) para controlar si los cambios realizados en una tabla o cursor por otro usuario de la red se sobrescriben cuando usted envía sus propias actualizaciones. Si se establece el parámetro Force de TABLEUPDATE ( ) como verdadero (T) y la propiedad CURSORSETPROP ( ) updatetype está establecida al valor predeterminado 2, los datos anteriores se actualizarán con los nuevos datos enviados, siempre y cuando el valor del campo clave de la tabla remota no se haya modificado. Si el valor del campo clave de la tabla remota ha cambiado o si la propiedad UPDATETYPE está establecida a 1, Visual FoxPro enviará una instrucción DELETE y luego una instrucción INSERT a la tabla remota.

4.3.3 Modo de procesamiento síncrono y asíncrono de visual foxpro. A continuación se explica el comportamiento de cada tipo de procesamiento:

4.3.3.1 Procesamiento síncrono. En el modo síncrono, el control no vuelve a la aplicación hasta que no se haya completado la ejecución de la función.

Modo Síncrono por lotes. Cuando se ejecuta una instrucción de paso a través de SQL de forma síncrona en el modo por lotes, el control no vuelve hasta que no se hayan recuperado todos los conjuntos de resultados. El nombre del primer cursor se especifica mediante el parámetro cNombreCursor en la función original. Si el cursor especificado ya existe, el conjunto de resultados sobrescribirá el cursor en el modo síncrono por lotes, visual FoxPro

crea los nombres de cursores adicionales indexando de forma exclusiva al nombre del primer cursor.

Modo síncrono sin lotes. Cuando se ejecuta una instrucción de paso a través de SQL de manera síncrona en el modo sin lotes, la primera instrucción recupera el primer conjunto de resultados y devuelve 1. Luego se debe llamar a la función SQLMORERESULTS ( ) repetidas veces; si lo desea, especificar un nuevo nombre distinto para los conjuntos de resultados indexados de forma única al nombre básico. Cuando SQLMORERESULTS ( ) devuelve un valor 2, no hay más resultados disponibles.

4.3.3.2 Procesamiento asíncrono. En el modo asíncrono, la aplicación debe continuar llamando a la misma función de paso a través de SQL hasta que devuelva un valor distinto de 0 (en ejecución). El nombre predeterminado para el conjunto de resultados, Sqlresult, se puede cambiar explícitamente con el parámetro cNombreCursor la primera vez que llama a la función. Si el nombre especificado para un conjunto de resultados ya se ha utilizado, el nuevo conjunto de resultados sobrescribirá la información del cursor existente.

Modo asíncrono por lotes. Cuando se ejecuta el modo por lotes de forma asíncrona, cada llamada repetida de la función original devuelve 0 (en ejecución) hasta que se devuelven todos los conjuntos de resultados a los cursores especificados. Una vez recuperados todos los resultados, el valor de retorno es el número de cursores o bien un número negativo que indica un error.

Modo asíncrono sin lotes. Cuando se procesa en modo asíncrono sin lotes, SQLEXEC() devuelve un valor 1 al completar la recuperación de cada conjunto de resultados. A

continuación, la aplicación debe llamar a SQLMORERESULTS ( ) repetidas veces hasta que devuelva un valor de 2, indicando que no hay más resultados disponibles.

Los conjuntos de resultados remotos se recuperan en dos etapas: primero se prepara el conjunto de resultados en el Servidor; luego el conjunto de resultados se recopila en un cursor local de Visual FoxPro.

4.4 IMPLEMENTACIÓN DE UNA APLICACIÓN FOXPRO / MySQL.

A continuación se explicará en una forma general algunos aspectos de la aplicación Cliente desarrollada para los laboratorios de química, biología y microbiología del CIOH.

Si se desea analizar de una forma más detallada la aplicación Cliente desarrollada dirigirse al manual técnico (Anexo 4).

Para desarrollar una aplicación Cliente/Servidor en Visual FoxPro, se deben llevar a cabo los siguientes pasos:

- Realizar al menos una conexión con el origen de datos.
- Ejecutar una o varias sentencias SQL.
- Procesar resultados.
- Gestionar los errores que se puedan producir durante la ejecución de la aplicación.
- Cierre de la conexión.

Una vez estudiadas las características, propiedades, eventos y en general la metodología necesaria para realizar una aplicación Cliente/Servidor bajo Visual FoxPro y MySQL, se mostrará cada uno de los pasos a seguir para el desarrollo de dicha aplicación.

Una vez elaborada la base de datos “BDLABQUIM” se crea el Origen de Datos ODBC (DSN) utilizando cualquiera de las formas descritas.

4.4.1 Instalación de MyODBC. A continuación se describirá el proceso de instalación del driver MyODBC ODBC 32 bits para Windows 9x. Gracias a éste driver podremos acceder desde un cliente Windows, con Visual FoxPro a nuestro Servidor remoto MySQL.

Lo primero que debemos hacer es conseguir el driver MyODBC más reciente.

Una vez que tenemos el programa en nuestro ordenador (cliente Windows), ejecutamos el programa de instalación, Setup.

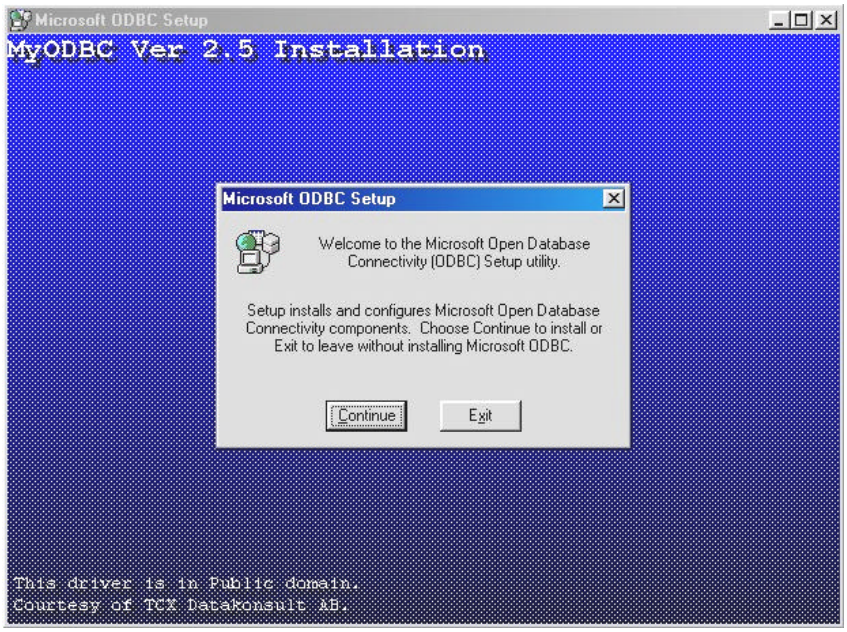

Figura 5: Ventana instalación MyODBC

La caja de diálogos de la instalación (Install Drivers) permite escoger diversos drivers ODBC para instalar, pero MyODBC solo da como opción el driver para MySQL, el cuál seleccionaremos.

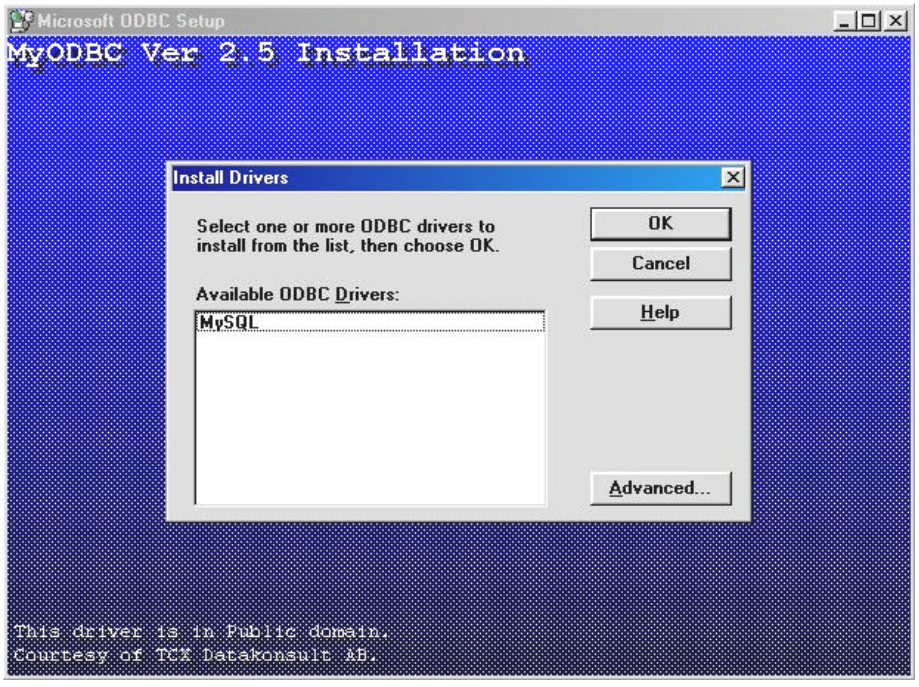

Figura 6 : Driver MySql

Una vez que el driver ODBC para MySQL ha sido instalado, la siguiente caja de diálogos nos pide configurar el origen de la base de datos a conectar. Dentro de la caja de diálogos Data Source, elegimos el botón Add, para añadir un nuevo origen de datos.

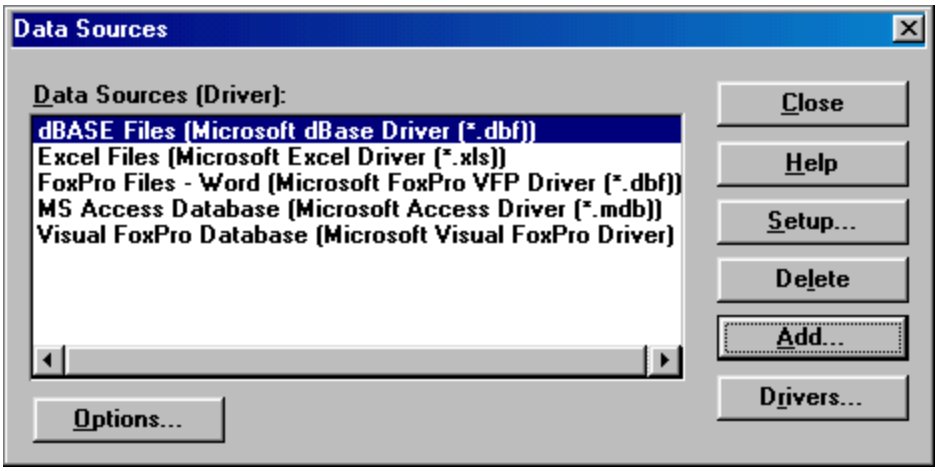

Figura 7 : Orígenes de datos

A continuación seleccionamos el driver ODBC correspondiente al de la base de datos a la cuál nos vamos conectar, en nuestro caso MySQL.

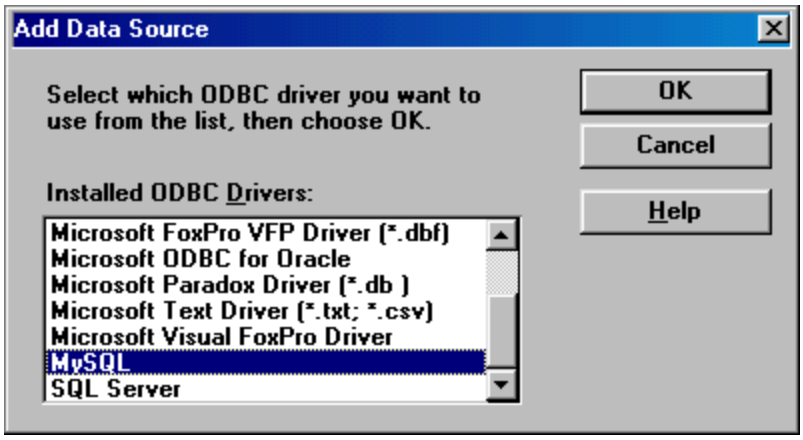

Figura 8 : ODBC de la Base de datos.

Y es este momento cuando nos aparece el cuadro de configuración de nuestra conexión.

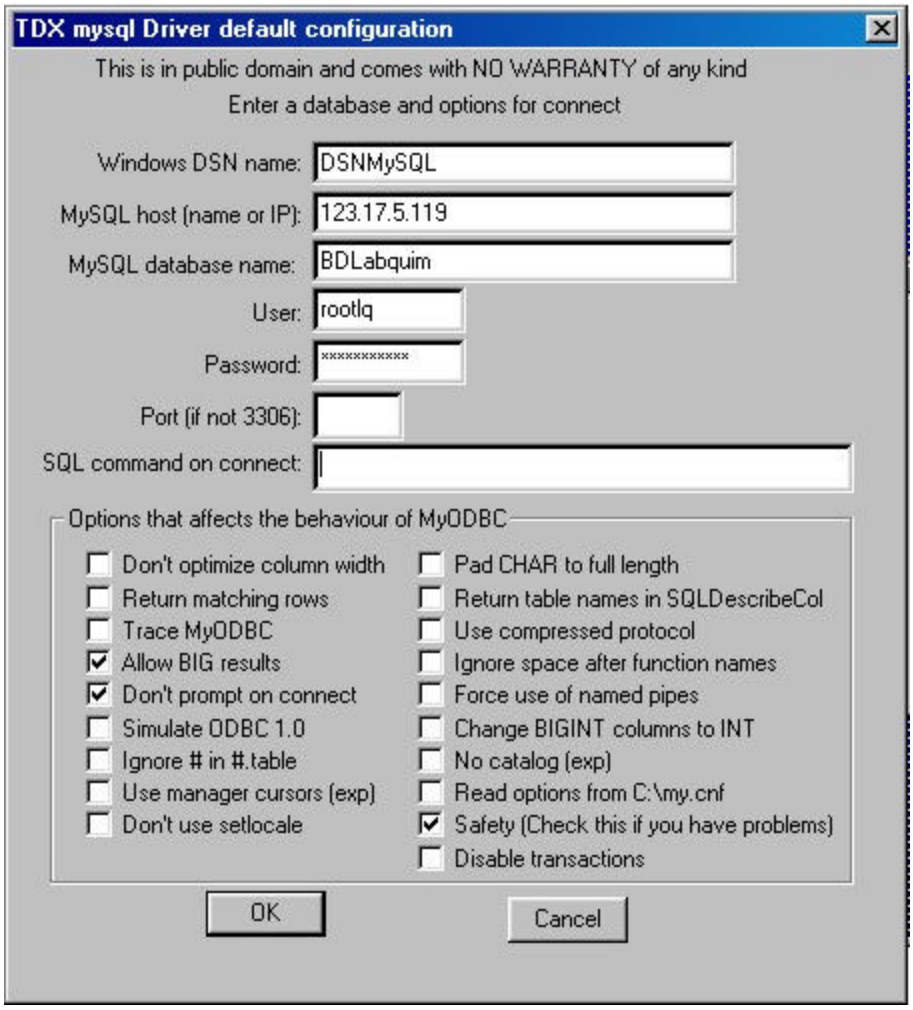

Figura 9 : Ventana de configuración de MySql.

3.4.1.1 Configuración del cliente windows. Con el controlador de MySQL instalado. Podemos hacer lo siguiente con el objetivo de configurar la parte Cliente.

Hacemos clic en Inicio, Configuración, Panel de control en el icono ODBC Data Sources (32 bit) luego vamos a la pestaña Use DNS hacemos clic en el botón ADD. Desde allí seleccionamos el Driver de MySQL.

Después aparece la pantalla de configuración del Driver de MySQL que se muestra en la anterior gráfica, en la que debemos ingresar una serie de datos para que podamos entrar en el servidor donde tenemos nuestro MySQL.

Windows DNS Name: Ingresamos el nombre con el que se va a identificar en Windows la Base de Datos MySQL a la que vamos a acceder. (ejemplo: DSNMySql)

MySQL Host (Name or IP): Aquí le indicamos la IP del Servidor o su nombre. (ejemplo: 192.168.0.1)

MySQL Database Name: El Nombre de la Base de datos que vamos a acceder. (ejemplo: BDLABQUIM)

User: El usuario con privilegios para acceder a la Base de Datos. (ejemplo: rootlq)

Password: El password para acceder de ese usuario

Port : Es el puerto por donde se conecta a la Base de datos, por defecto usara el 3306

3.4.1.2 Configuración del servidor linux . Para poder acceder al servidor Linux y acceder a MySQl no basta con lo anterior, sino que tenemos que dar privilegios a los usuarios para que puedan acceder a las Base de Datos y Tablas que contiene MySQL

Para hacer ésto necesitaremos acceder a la consola del servidor y acceder a MySQL

Para empezar accederemos como root a la Tabla mysql, donde se dan todos los privilegios de acceso a todos los usuarios

shell> mysql –user = root mysql

Ahora indicamos que desde la IP 192.168.0.2 el usuario: rootlq, con el password: *****, va a poder acceder a la Base de datos MySQL

```
mysql> INSERT INTO user (Host,User,Password)
VALUES('192.168.0.2','rootlqi',PASSWORD('****'));
```

Ahora indicamos que desde la IP 192.168.0.2 el usuario: rootlq, puede acceder a la tabla: NombreTabla con los privilegios de: Seleccionar, Insertar, Modificar, Borrar, Crear, y Borrar

```
mysql> INSERT INTO db
      (Host, Db,User, Select_priv, Insert_priv, Update_priv, Delete_priv,
       Create_priv, Drop_priv)
      VALUES
      ('192.168.0.2','NombreTabla','rootlqi','Y','Y','Y','Y','Y','Y');
```

Por último le indicamos que empiece a aplicar estos privilegios en este momento

```
mysql> FLUSH PRIVILEGES;
```

Tener en cuenta que las instrucciones siempre terminan con punto y coma (;)

Para salir del modo consola de Mysql basta digitar:

```
mysql> \q
```

Ya creado el DSN se procederá a crear un procedimiento de Visual FoxPro en el cual se debe digitar el código para establecer la conexión con la base de datos “DBLABQUIM” de la siguiente manera.

Figura 10 : Contraseña.

El siguiente código se encuentra asociado al botón Entrar.

```
* Procedimiento de arranque

STORE SQLCONNECT("DSNMySQL", "rootlq", "Laboratorio") TO gnconn
if gnconn<=0
        =MESSAGEBOX('No se puede realizar la conexión', 16, 'Error de Conexión')
ENDIF


IF gnconn>0
    Do while contador<=3
        sMySql='select Login, Passwords from passwords where
        Login=?ThisForm.TxtLogin.value and Passwords=?ThisForm.TxtPassword.value'
        Sqlexec(gnconn,sMySql,'cPasswords')
        IF not EOF()
```

```
            thisform.release
            do form frmEntrada.scx
            exit
        ELSE
            =MESSAGEBOX('Nombre de usuario y/o contraseña Incorrecta', 16, 'Error...')
            contador=contador+1
            ThisForm.TxtLogin.value=''
            ThisForm.TxtPassword.value=''
            exit
        ENDIF
enddo
ENDIF
```

La declaración de la variable que representan el entorno de ODBC y la conexión, es pública debido a que la aplicación se conecta a la base de datos en el momento en que inicia y la mantiene abierta durante todo el tiempo de ejecución de la aplicación. Estas variables solo se liberan cuando la aplicación termina. El proceso de conexión se lleva a cabo en un procedimiento, este procedimiento permite indicar el punto de carga de una aplicación.

Luego de haber desarrollado el código correspondiente a la conexión, se pasa a construir las diferentes opciones de la aplicación. En esta se podrá manipular información perteneciente a las tablas de la base de datos BDLABQUIM. Para cada formulario se visualizará y se detallará su código más adelante.

- **Menú Principal**

Figura 11 : Menú principal.

Es el primer formulario que se llama desde el formulario de contraseña de entrada, mediante la instrucción DO FORM frmentrada.scx. consta de una barra de menú con las siguientes opciones: Análisis y recepción de muestra, Búsqueda de muestras, Edición y Actualización de muestras, Formatos de análisis, Ayuda y por último Salir.

- **Opción Recepción y análisis de muestras**

Sitio de muestreo.

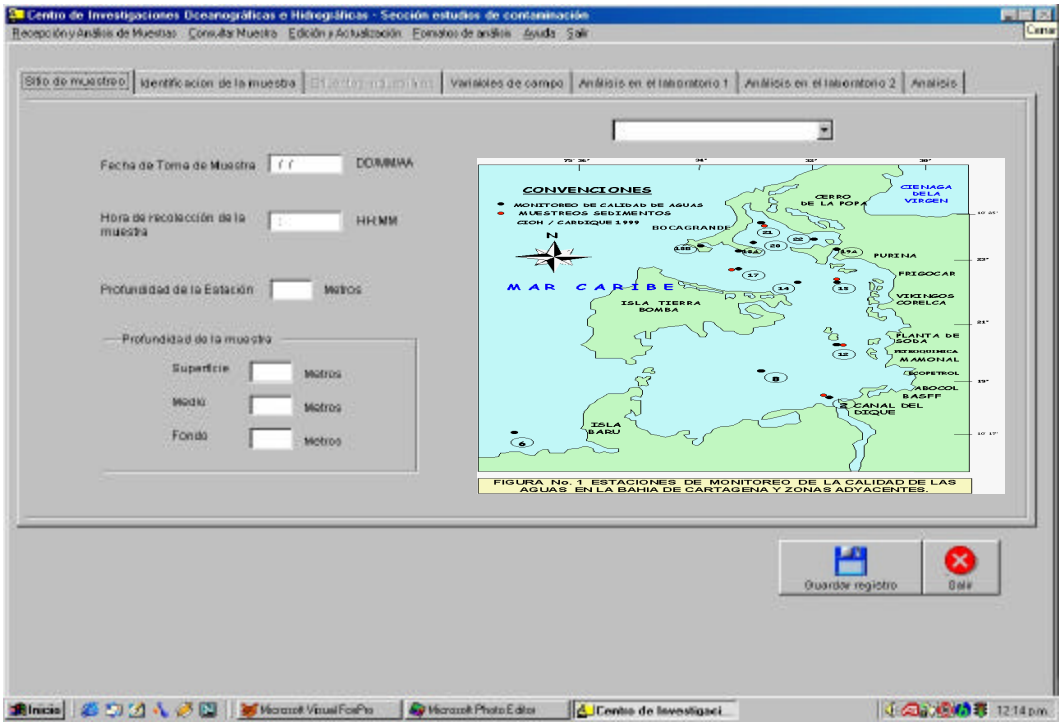

Figura 12 : Recepción y Análisis de muestras.

Este formulario fue diseñado utilizando el objeto *pageframe*. Todas las *page* que conforman el *pageframe* fueron diseñadas siguiendo una secuencia lógica de la información a ingresar. La primera de estas opciones es el *Sitio de Muestreo*, en ella se almacena toda la información relacionada al sitio de muestreo donde fue tomada la muestra; además se puede visualizar los diferentes mapas de los sitios de muestreo.

Las instrucciones necesarias para almacenar la información ingresada se encuentran en la el botón *Guardar Registro*. Todos los datos ingresados en este formulario son almacenados en la tabla muestra mediante la siguiente instrucción:

```
sMySql="insert into muestra(codfecha, fechatoma, horatoma, horarecep, nreplicas, fechaelimina, codprograma, profundidad1, profundidad2, profundidad3, codigo, tipo) values(?ThisForm.Pageframe1.PageIM.txtCodFecha.value,";
+"?ThisForm.Pageframe1.PageSM.txtFechaToma.value,";
+"?ThisForm.Pageframe1.PageSM.txtHoraToma.value,";
+"?ThisForm.Pageframe1.PageSM.txtProfundidad1.value,";
+"?ThisForm.Pageframe1.PageSM.TxtProfundidad2.value,";
+"?ThisForm.Pageframe1.PageSM.TxtProfundidad3.value)"
sqlexec(gnconn, sMySql)
```

Este código muestra la función SQLEXEC() en la cual se envía una instrucción SQL al origen de datos, donde se procesa la instrucción. Con esta instrucción se envía a la tabla *muestra* los datos del *Sitio de Muestreo*.

- **Opción Identificación de la muestra**

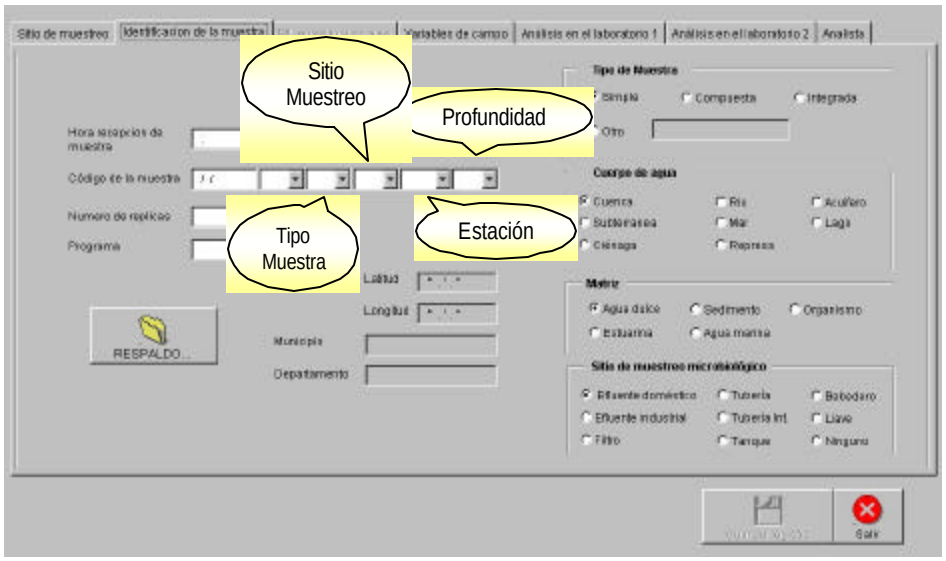

Figura 13 : Identificación de la muestra.

En este formulario se ingresan otros datos de la muestra relacionados a su Identificación.

Cuando el formulario *frmprincipal* inicia su proceso de carga, se ejecuta el código correspondiente al procedimiento *INIT*, dicho código muestra la función SQLEXEC( ) en la cual se envía una instrucción SQL al origen de datos, donde se procesa la instrucción. Con esta instrucción se solicitan algunos de los datos que conforman el código de la muestra algunos de estos son el *origen de la muestra, tipo de muestra, sitio de muestreo, estación y profundidad de la muestra* con el fin de que puedan ser visualizados inmediatamente se carga el formulario.

```
*muestra en formulario el origen de la muestra (I/E)
sMySql='select * from codorigen'
SQLEXEC(gnconn, sMySql, 'codorigen')
```

*muestra en formulario el tipo de la muestra

```
sMySql='select * from codtipomuestra'
SQLEXEC(gnconn, sMySql, 'codtipomuestra')

*muestra en formulario el sitio de muestreo
sMySql="select * from codsitiomuestreo"
SQLEXEC(gnconn, sMySql, 'codsitiomuestreo')

*muestra en formulario la estación
sMySql='select * from codestacion'
SQLEXEC(gnconn, sMySql, 'codestacion')

*muestra en formulario la profundidad de la muestra
sMySql='select * from codprofmuestra'
SQLEXEC(gnconn, sMySql, 'codprofmuestra')
```

Al escoger el *sitio de muestreo* aparece inmediatamente el Municipio y Departamento donde se encuentra ese sitio de muestreo en particular. Al seleccionar algún sitio de muestreo se ejecuta el código correspondiente al procedimiento *Click* del objeto *txtcodsitiomuestreo*.

```
sCodigo = alltrim(This.value)
sMySql="SELECT * FROM codsitiomuestreo WHERE sitio = ?sCodigo"
SQLEXEC(gnconn,sMySql,'cTemporal')
go top  && Va al inicio de la tabla temporal
Thisform.PageFrame1.PageIM.txtDepartamento.value=cTemporal.Departamento
Thisform.PageFrame1.PageIM.txtMunicipio.value=cTemporal.Municipio
```

Igual sucede con el objeto *txtcodestacion*. Al seleccionar alguna estación de muestreo se muestra inmediatamente las coordenadas de su Latitud y Longitud al igual que la ejecución del código correspondiente al procedimiento *Click* del objeto *txtcodestacion.*

```
FOR nCnt = 1 TO This.ListCount
IF This.Selected(nCnt)  && ¿Está seleccionado?
        sMySql='select * from codestacion where This.Selected(nCnt) in Estacion'
    SQLEXEC(gnconn, sMySql, 'codestacion')
     Thisform.PageFrame1.PageIM.TxtLatitud.value=codestacion.Latitud
     Thisform.PageFrame1.PageIM.TxtLongitud.value=codestacion.Longitud
ENDIF
ENDFOR
```

Cuando se escoge la opción *Efluente Industrial (EI)* en alguno de los objetos *txtcodorigen, txtcodtipomuestra, txtcodsitiomuestreo, txtcodestacion, txtcodprofmuestra*. Se ejecuta el código correspondiente al procedimiento *Click* el cual permite que se active o desactive la opción *Efluentes Industriales.*

```
sCodestacion = alltrim(this.value)
if sCodestacion=='EI'
 if sTmuestra<>'EI' .OR. sTmuestra==''
   if sCSitiomuestreo<>'EI' .OR. sCSitiomuestreo==''
     if sProfmuestra<>'EI' .OR. sProfmuestra==''
         Thisform.Pageframe1.Page3.enabled=.T.
     endif
```

```
    endif
 endif
else
 if sTmuestra<>'EI' .OR. sTmuestra==''
    if sCSitiomuestreo<>'EI' .OR. sCSitiomuestreo==''
       if sProfmuestra<>'EI' .OR. sProfmuestra==''
          Thisform.Pageframe1.Page3.enabled=.F.
       endif
    endif
 endif
endif
```

- **Opción Respaldo**

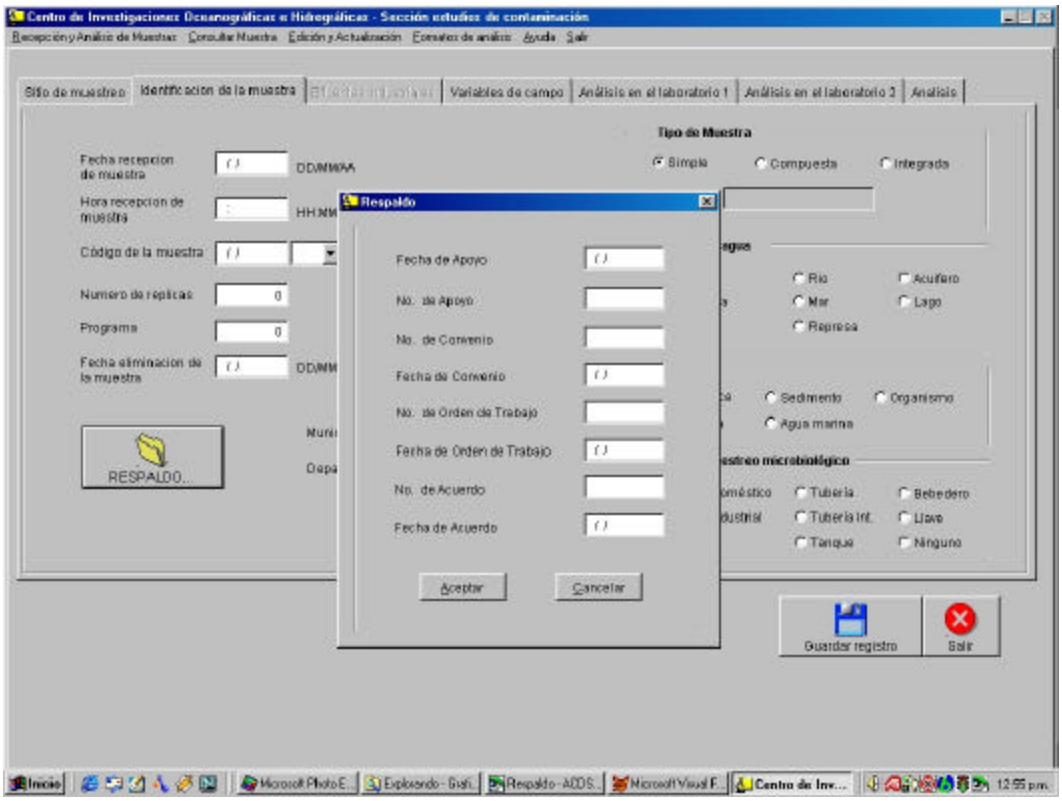

Figura 14 : Respaldo

El botón Aceptar de este formulario permite adicionar un Nuevo registro con los datos que respaldan una muestra, estos datos son almacenados en la tabla *respaldo*; teniendo en cuenta que no todas las muestras tienen respaldo. El código necesario para lograr este fin se muestra a continuación.

```
sMySql="insert into respaldo (FechaApoyo, NumeroApoyo, NumeroConvenio, FechaConvenio,
NumeroOtrabajo, FechaOtrabajo, NumeroAcuerdo, FechaAcuerdo)";
+"values(?ThisForm.txtFechaApoyo.value,?ThisForm.txtNumeroApoyo.value,";
+"?ThisForm.txtNumeroConvenio.value,?ThisForm.txtFechaConvenio.value,";
+"?ThisForm.txtNumeroOtrabajo.value, ?ThisForm.txtFechaOtrabajo.value, ";
+"?ThisForm.txtNumeroAcuerdo.value, ?ThisForm.txtFechaAcuerdo.value)"
```

sqlexec(gnconn, sMySql)

- **Opción Efluentes Industriales**

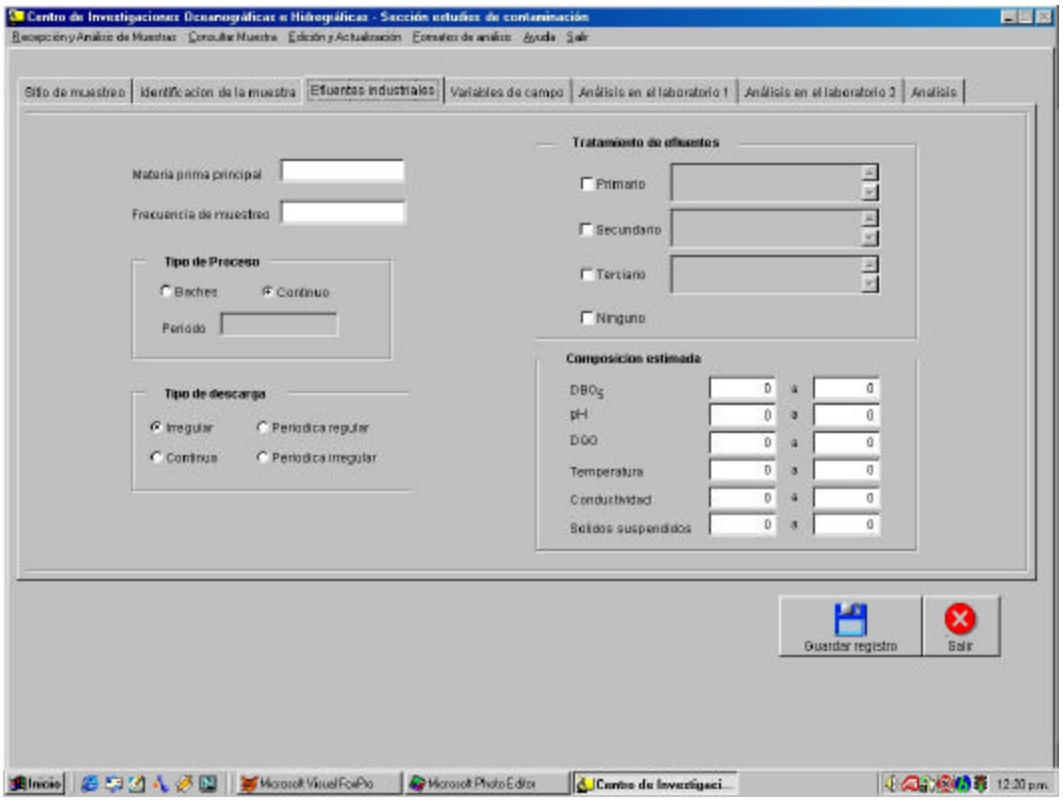

Figura 15 : Efluentes Industriales.

En esta opción se ingresan todos los datos de las muestras que son tomadas en Efluentes Industriales, es decir, en empresas o industrias. Este es el único formulario disponible para tal fin. La información ingresada en este formulario es almacenada en la tabla *efluentes,* cuando se presiona sobre el botón *Guardar registro;* el código necesario para lograr este fin se encuentra asociado al procedimiento *click* de este mismo botón y se muestra a continuación.

if ThisForm.Pageframe1.Page3.enabled=.T.

```
sMySql="insert into efluentes(codigo, materiappal, tipoproceso, periodo, frecuenciamuestreo,
tipodescarga, ";
+"descripcionTefluentesP, descripcionTefluentesS, descripcionTefluentesT,"; +"DBO5Ini,
DBO5Fin, pHIni, pHFin, DQOIni, DQOFin, TempIni, TempFin,"; +"ConductividadIni,
ConductividadFin, SSTIni, SSTFin)";
+"values(?nue_cod,?ThisForm.Pageframe1.Page3.EfluenIndMateriaPriPal.value,";+"?tipopro,
?ThisForm.Pageframe1.Page3.TipoPPeriodo.value,";
+"?ThisForm.Pageframe1.Page3.EfluenIndFrecueMues.value,?tipodesc,"; +"?trataeflueP,
?trataeflueS, ?trataeflueT,";
+"?ThisForm.Pageframe1.Page3.DBO5Ini.value,";
+"?ThisForm.Pageframe1.Page3.DBO5Fin.value,";
+"?ThisForm.Pageframe1.Page3.PHIni.value,";
+"?ThisForm.Pageframe1.Page3.PHfin.value,";
+"?ThisForm.Pageframe1.Page3.DQOIni.value,";
+"?ThisForm.Pageframe1.Page3.DQOFin.value,";
+"?ThisForm.Pageframe1.Page3.TempIni.value,";
+"?ThisForm.Pageframe1.Page3.TempFin.value,";
+"?ThisForm.Pageframe1.Page3.ConducIni.value,";
+"?ThisForm.Pageframe1.Page3.ConducFin.value,";
+"?ThisForm.Pageframe1.Page3.SSTIni.value,";
+"?ThisForm.Pageframe1.Page3.SSTFin.value)"
        sqlexec(gnconn, sMySql)
endif
```

- **Opción Variables de campo**

En esta opción se encuentra disponible para ingresar toda la información relacionada a las variables de campo donde fueron tomadas las muestras, estas variables de campo también son llamadas Parámetros InSitu, todos estos datos son almacenados en la tabla *variablesdecampo*. En el procedimiento *Click* del botón guardar registro, se encuentra el siguiente código necesario para lograr tal fin.

```
sMySql="insert into variablesdecampo(codigo, caudal,ph, conductividad, od, salinidad,
velocidadviento, direccionviento, turbidez,aspecto, color, olor, transparencia,
tempsuperficialagua, residuo, nubosidad, observaciones)";
+"values(?nue_cod, ?ThisForm.Pageframe1.Page4.Txtcaudal.value,";
+"?ThisForm.Pageframe1.Page4.Txtph.value, ";
+"?ThisForm.Pageframe1.Page4.Txtconductividad.value,";
+"?ThisForm.Pageframe1.Page4.Txtod.value, ";
+"?ThisForm.Pageframe1.Page4.Txtsalinidad.value,";
+"?ThisForm.Pageframe1.Page4.Txtvviento.value, ";
+"?ThisForm.Pageframe1.Page4.CmbDirViento.value,";
+"?ThisForm.Pageframe1.Page4.Txtturbidez.value, ";
+"?ThisForm.Pageframe1.Page4.Cmbaspecto.value,";
+"?ThisForm.Pageframe1.Page4.Txtcolor.value, ";
+"?ThisForm.Pageframe1.Page4.Txtolor.value,";
+"?ThisForm.Pageframe1.Page4.Txttranparencia.value, ";
+"?ThisForm.Pageframe1.Page4.Txttemperatura.value,";
+"?ThisForm.Pageframe1.Page4.Cmbresiduo.value, ?nubosidad, ";
```

+"?ThisForm.Pageframe1.Page4.EdtObservaciones.value)"

sqlexec(gnconn, sMySql)

Cuando el formulario *frmprincipal* inicia su proceso de carga, se ejecuta el código correspondiente al procedimiento *INIT,* en donde se envía una instrucción SQL al origen de datos para ser procesada. Con esta instrucción se solicitan los datos concernientes a *aspecto* y *residuo* con el fin que puedan ser visualizados inmediatamente se carga el formulario. El código para lograr este fin es el siguiente.

*Muestra el aspecto de la muestra

```
sMySql='select nombre from aspecto'
SQLEXEC(gnconn, sMySql, 'aspecto')
```

*Muestra el residuo de la muestra

```
sMySql="select nombre from residuo"
SQLEXEC(gnconn, sMySql, 'residuo')
```

- **Opción Análisis en el Laboratorio 1 y Análisis en el Laboratorio 2**

En este formulario se seleccionan los análisis que se le van a efectuar a las muestras, a la vez que se ingresan algunos datos relacionados al análisis que se ha seleccionado. Para una muestra se puede seleccionar más de un parámetro. Los parámetros seleccionados así como los datos ingresados de los mismos, son almacenados en la tabla *parametros*; esta tabla es dinámica, es decir que va creciendo o cambiando su estructura a medida que se van ingresando nuevos registros. Cuando se ingresa un nuevo registro a la tabla *parametros* se

crea un nuevo campo que tiene como nombre el código sin la fecha (IAPCVPC01F1) de la muestra que se esté ingresando en ese momento, en este campo se va a almacenar el código de la muestra asociado al parámetro que se ha elegido realizar a la misma. El código necesario para almacenar los datos en la respectiva tabla, se encuentra asociado al procedimiento *Click* del botón guardar registro.

```
&&Devuelve un número de caracteres específico de una expresión de caracteres.
&&Se elimina la fecha al código de la muestra.
CodTemp=SUBSTR(ALLTRIM(nue_cod),9,15)

&&Se crea un nuevo campo a la tabla parametros
sMySql="ALTER TABLE parametros ADD "+ CodTemp +" VARCHAR(24)"
sqlexec(gnconn,sMySql)

if ThisForm.Pageframe1.Page5.Chkamonio.value==1
sMySql="UPDATE parametros SET " + CodTemp + " = ?nue_cod WHERE nombre =
?ThisForm.Pageframe1.Page5.Chkamonio.caption"
        sqlexec(gnconn,sMySql)
endif

if ThisForm.Pageframe1.Page5.Chkaerobmesofilos.value==1
sMySql="UPDATE parametros SET " + CodTemp + " = ?nue_cod WHERE nombre =
?ThisForm.Pageframe1.Page5.Chkaerobmesofilos.caption"
        sqlexec(gnconn,sMySql)
```

```
        endif

if ThisForm.Pageframe1.Page5.Chkbiofito.value==1
sMySql="UPDATE parametros SET " + CodTemp + " = ?nue_cod WHERE nombre =
?ThisForm.Pageframe1.Page5.Chkbiofito.caption"
            sqlexec(gnconn,sMySql)
endif

if ThisForm.Pageframe1.Page5.Chkbiozoo.value==1
sMySql="UPDATE parametros SET " + CodTemp + " = ?nue_cod WHERE nombre =
?ThisForm.Pageframe1.Page5.Chkbiozoo.caption"
            sqlexec(gnconn,sMySql)
endif

if ThisForm.Pageframe1.Page5.ChkCompOrgano.value==1
sMySql="UPDATE parametros SET " + CodTemp + " = ?nue_cod WHERE nombre =
'Compuestos organoclorados'"
            sqlexec(gnconn,sMySql)
endif

if ThisForm.Pageframe1.Page5.ChkCTFilMem.value==1
sMySql="UPDATE parametros SET " + CodTemp + " = ?nue_cod WHERE nombre =
'Coliformes totales por filtración de membrana'"
            sqlexec(gnconn,sMySql)
```

```
endif

if ThisForm.Pageframe1.Page5.Chkcoliftotal.value==1
sMySql="UPDATE parametros SET " + CodTemp + " = ?nue_cod WHERE nombre =
'Coliformes totales por NMP'"
        sqlexec(gnconn,sMySql)
endif

if ThisForm.Pageframe1.Page5.Chkcolifecal.value==1
sMySql="UPDATE parametros SET " + CodTemp + " = ?nue_cod WHERE nombre =
'Coliformes fecales por NMP'"
        sqlexec(gnconn,sMySql)
endif

if ThisForm.Pageframe1.Page5.ChkCFFilMem.value==1
sMySql="UPDATE parametros SET " + CodTemp + " = ?nue_cod WHERE nombre =
'Coliformes fecales por filtración de membrana'"
        sqlexec(gnconn,sMySql)
endif

if ThisForm.Pageframe1.Page5.ChkClorofila.value==1
sMySql="UPDATE parametros SET " + CodTemp + " = ?nue_cod WHERE nombre =
?ThisForm.Pageframe1.Page5.ChkClorofila.caption"
        sqlexec(gnconn,sMySql)
```

```
endif

if ThisForm.Pageframe1.Page5.Chkdqo.value==1
sMySql="UPDATE parametros SET " + CodTemp + " = ?nue_cod WHERE nombre =
?ThisForm.Pageframe1.Page5.Chkdqo.caption"
        sqlexec(gnconn,sMySql)
endif

if ThisForm.Pageframe1.Page5.Chkdbo5.value==1
sMySql="UPDATE parametros SET " + CodTemp + " = ?nue_cod WHERE nombre = 'DBO5'"
        sqlexec(gnconn,sMySql)
endif

if ThisForm.Pageframe1.Page5.Chkenterococos.value==1
sMySql="UPDATE parametros SET " + CodTemp + " = ?nue_cod WHERE nombre =
?ThisForm.Pageframe1.Page5.Chkenterococos.caption"
        sqlexec(gnconn,sMySql)
endif

if ThisForm.Pageframe1.Page6.Chkfosforototal.value==1
sMySql="UPDATE parametros SET " + CodTemp + " = ?nue_cod WHERE nombre =
?ThisForm.Pageframe1.Page6.Chkfosforototal.caption"
        sqlexec(gnconn,sMySql)
endif
```

```
if ThisForm.Pageframe1.Page6.Chkayg.value==1
sMySql="UPDATE parametros SET " + CodTemp + " = ?nue_cod WHERE nombre =
?ThisForm.Pageframe1.Page6.Chkayg.caption"
        sqlexec(gnconn,sMySql)
endif

If ThisForm.Pageframe1.Page6.Chkhapn.value==1
sMySql="UPDATE parametros SET " + CodTemp + " = ?nue_cod WHERE nombre =
?ThisForm.Pageframe1.Page6.Chkhapn.caption"
        sqlexec(gnconn,sMySql)
endif

if ThisForm.Pageframe1.Page6.ChkMOFO.value==1
sMySql="UPDATE parametros SET " + CodTemp + " = ?nue_cod WHERE nombre =
?ThisForm.Pageframe1.Page6.ChkMOFO.caption"
        sqlexec(gnconn,sMySql)
endif

if ThisForm.Pageframe1.Page6.ChkNKT.value==1
sMySql="UPDATE parametros SET " + CodTemp + " = ?nue_cod WHERE nombre =
?ThisForm.Pageframe1.Page6.ChkNKT.caption"
        sqlexec(gnconn,sMySql)
endif
```

```
if ThisForm.Pageframe1.Page6.ChkNitratos.value==1
sMySql="UPDATE parametros SET " + CodTemp + " = ?nue_cod WHERE nombre =
?ThisForm.Pageframe1.Page6.ChkNitratos.caption"
        sqlexec(gnconn,sMySql)
endif

if ThisForm.Pageframe1.Page6.ChkNitritos.value==1
sMySql="UPDATE parametros SET " + CodTemp + " = ?nue_cod WHERE nombre =
?ThisForm.Pageframe1.Page6.ChkNitritos.caption"
        sqlexec(gnconn,sMySql)
endif

if ThisForm.Pageframe1.Page6.ChkOD.value==1
sMySql="UPDATE parametros SET " + CodTemp + " = ?nue_cod WHERE nombre =
?ThisForm.Pageframe1.Page6.ChkOD.caption"
        sqlexec(gnconn,sMySql)
endif

if ThisForm.Pageframe1.Page6.ChkOrtoFosfato.value==1
sMySql="UPDATE parametros SET " + CodTemp + " = ?nue_cod WHERE nombre =
?ThisForm.Pageframe1.Page6.ChkOrtoFosfato.caption"
        sqlexec(gnconn,sMySql)
endif
```

```
if ThisForm.Pageframe1.Page6.Chkprodprim.value==1
sMySql="UPDATE parametros SET " + CodTemp + " = ?nue_cod WHERE nombre =
'Productividad primaria"'
        sqlexec(gnconn,sMySql)
endif

if ThisForm.Pageframe1.Page6.Chkseston.value==1
sMySql="UPDATE parametros SET " + CodTemp + " = ?nue_cod WHERE nombre =
?ThisForm.Pageframe1.Page6.Chkseston.caption"
        sqlexec(gnconn,sMySql)
endif

if ThisForm.Pageframe1.Page6.Chksst.value==1
sMySql="UPDATE parametros SET " + CodTemp + " = ?nue_cod WHERE nombre =
?ThisForm.Pageframe1.Page6.Chksst.caption"
        sqlexec(gnconn,sMySql)
endif

if ThisForm.Pageframe1.Page6.Chkhumedad.value==1
sMySql="UPDATE parametros SET " + CodTemp + " = ?nue_cod WHERE nombre =
?ThisForm.Pageframe1.Page6.Chkhumedad.caption"
        sqlexec(gnconn,sMySql)
endif
```

En los formularios de *Análisis en el Laboratorio 1* y *Análisis en el Laboratorio 2*, se encuentran dos botones que permiten visualizar el *tiempo de almacenamiento* de la muestra desde el momento que se ha empezado a realizar el análisis y las *Técnicas y procedimientos* por medio de los cuales se realizan los análisis.

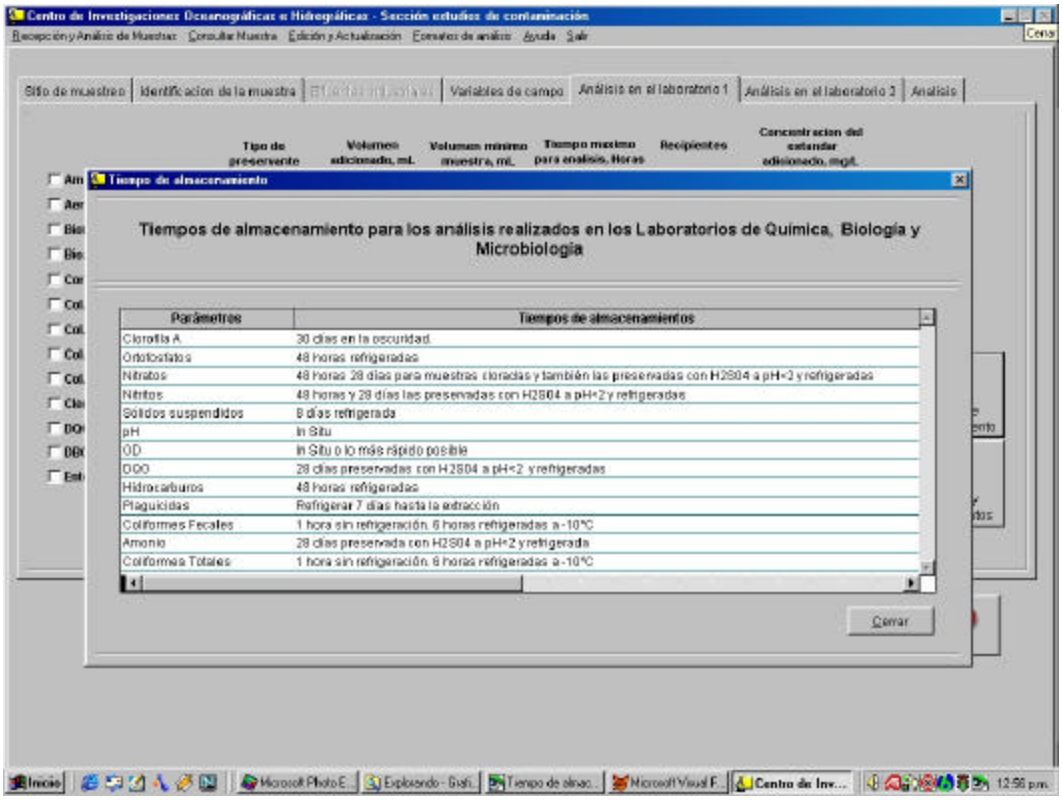

Figura 16 : Tiempos de almacenamientos.

Cuando el formulario *frmalmacenamiento* inicia su proceso de carga, se ejecuta el código correspondiente al procedimiento *INIT.* Con esta instrucción se solicitan los datos del tiempo de almacenamiento de las muestras a la tabla *almacenamiento.* El código necesario para lograr este fin es el siguiente.

* Generamos el cursor cAlmacena

SQLEXEC(gnconn,"SELECT parametros, tiempo FROM almacenamiento","cAlmacena")

go top && Va al inicio de la tabla temporal

Thisform.Grid1.RecordSource="cAlmacena"

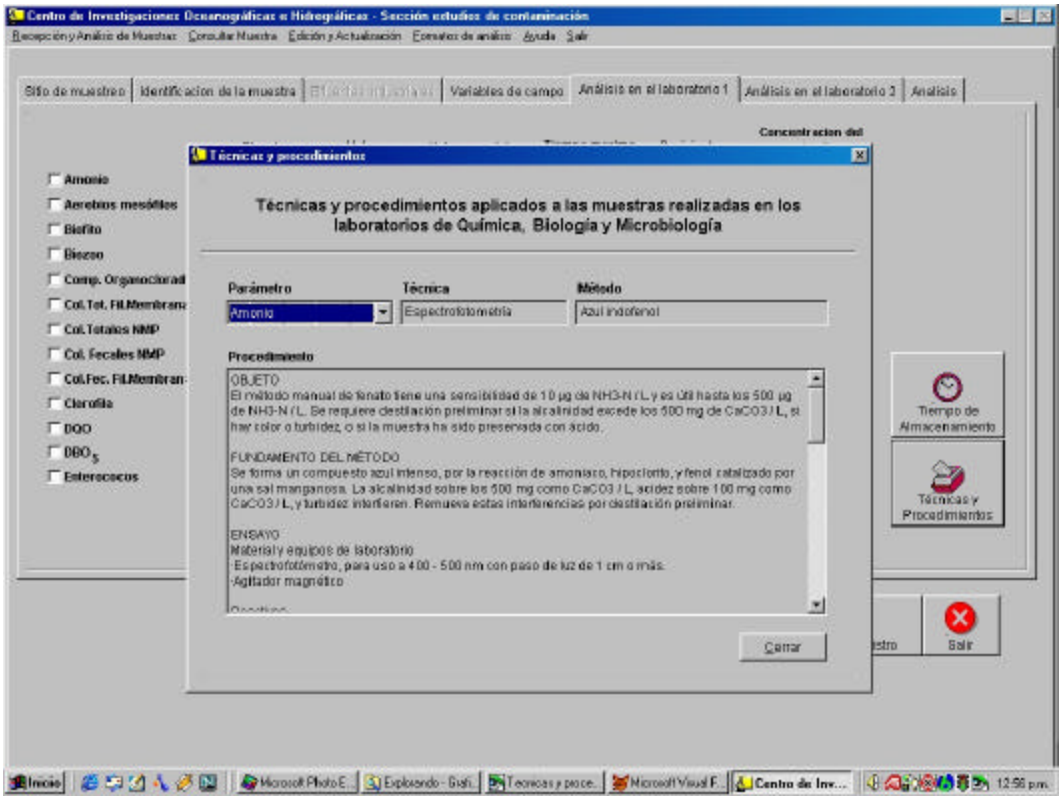

Figura 17 : Técnicas y procedimientos.

Al seleccionar algún parámetro de los que se encuentran en la lista despegable, se activa inmediatamente el código asignado al procedimiento *click* de este objeto, para así visualizar la Técnica, método y procedimiento de cada uno de los parámetros seleccionados. El código necesario para lograr este fin es el siguiente.

```
Local sParametro
sParametro = alltrim(This.value)
sMySql='SELECT nombre FROM parametros WHERE nombre = ?sParametro'
SQLEXEC(gnconn,sMySql)
ThisForm.TxtTecnica.value=cParametros.tecnica
ThisForm.TxtMetodo.value=cParametros.metodo
ThisForm.EdtProcedimiento.value=cParametros.procedimiento
```

- **Opción Analista**

Figura 18 : Analista.

Esta última opción de este formulario permite seleccionar por medio de una lista despegable el código del analista que va a realizar cada uno de los análisis para la muestra que se está ingresando en ese momento. Cuando se ejecuta el proceso de carga del formulario, se ejecuta el código correspondiente al procedimiento *INIT*. Con esta instrucción se solicita a la

tabla *analista* el código de todos los analistas encargados de realizar los análisis de las muestras, con el fin que puedan ser visualizados inmediatamente se carga el formulario. El código para lograr este fin se muestra a continuación.

```
*muestra en formulario el codigo del analista
sMySql='select Codigo from analista'
SQLEXEC(gnconn, sMySql, 'cCodAnalista')
```

Cuando se selecciona alguno de los códigos de los analistas que se encuentran en la lista despegable, inmediatamente se activa el código asignado al procedimiento *Click* de este objeto para permitir visualizar los datos del analista que va a realizar los análisis. El código necesario para lograr este fin es el siguiente.

```
local sCodigoA
sCodigoA = alltrim(This.value)

sMySql="SELECT * FROM analista WHERE Codigo = ?sCodigoA"
SQLEXEC(gnconn,sMySql,'cCodAna')

Thisform.pageframe1.page7.txtCargo.value=cCodAna.Cargo
Thisform.pageframe1.page7.txtNombre.value=cCodAna.Nombre
Thisform.pageframe1.page7.txtApellido.value=cCodAna.Apellido

ThisForm.BtnGuardar.enabled=.T.
```

Cómo la selección del analista es el último dato necesario para guardar toda la información de las muestras que se han ingresado previamente en los anteriores formularios, entonces es cuando se activa el botón *Guardar Registro*, para así de esta formar poder guardar toda la información de la muestra próxima a analizarse.

En el procedimiento *click* del botón *Guardar registro* se encuentra el código necesario para guardar en la tabla *analista* el código de la muestra que va a analizar. Esta tabla va cambiando su estructura a medida que se van ingresando nuevas muestras, ya que se van añadiendo nuevos campos con el código sin la fecha, de la muestra que va a ser analizada. El código necesario para lograr este fin es el siguiente.

```
*Crea código automatico para la tabla analista
sMySql="ALTER TABLE analista ADD "+ CodTemp +" VARCHAR(24)"
sqlexec(gnconn,sMySql)
sCodAnalista = alltrim(ThisForm.Pageframe1.Page7.cmbCodigoA.value)
SQLEXEC(gnconn,"UPDATE analista SET " + CodTemp + " = ?nue_cod WHERE Codigo =
?sCodAnalista")
```

La opción Edición y actualización consta de un submenú, en el que se puede escoger la opción deseada.

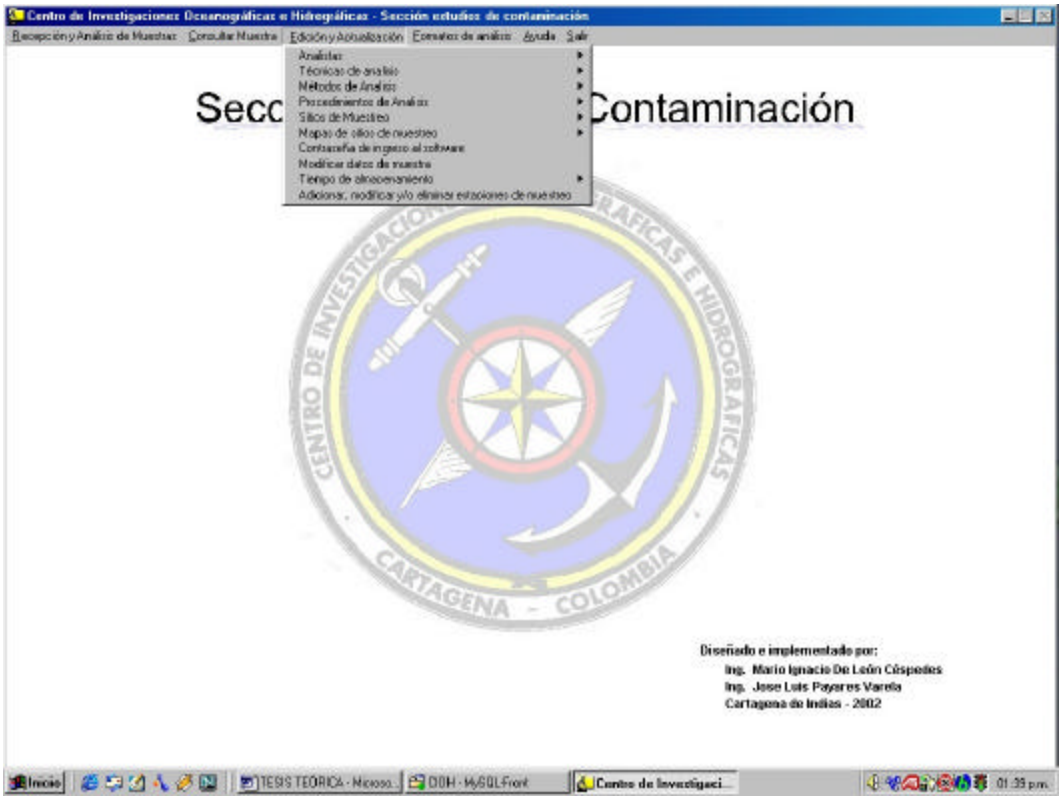

Figura 19 : Edición y Actualización.

Entre las opciones que aparecen se encuentran las siguientes.

- **Opción Adicionar método de análisis**

Figura 20 : Adicionar Método de Análisis.

En este formulario se pude ingresar un nuevo método de análisis para cada uno de los parámetros o análisis que son realizados a las muestras. Cuando se inicia el proceso de carga de este formulario, se ejecuta el código asignado al procedimiento *INIT*, el cual permite visualizar el listado de los parámetros y las técnicas que se encuentran almacenadas en la tabla *parametros*. El código dispuesto para tal fin es el siguiente.

*muestra en formulario el listado de los parámetros

```
sMySql='SELECT Nombre FROM parametros ORDER BY Nombre ASC'
SQLEXEC(gnconn, sMySql, 'cParametros')
*muestra en formulario el listado de las técnicas
sMySql='SELECT Tecnica FROM parametros ORDER BY Tecnica ASC'
SQLEXEC(gnconn, sMySql, 'cTecnica')
```

Cuando se han ingresado todos los datos necesarios para realizar el almacenamiento del nuevo método, se procede a presionar sobre el botón *Adicionar* y activar inmediatamente el código asociado al procedimiento *Click,* los datos del nuevo método son almacenados en la tabal *parametros.* El código para lograr tal fin es el siguiente.

```
sMySql="Update parametros set metodo=?ThisForm.Edit1.value where
nombre=?ThisForm.Combo1.value and tecnica=?ThisForm.Combo2.value"
sqlexec(gnconn, sMySql)
```

- **Opción Adicionar tiempos de almacenamiento**

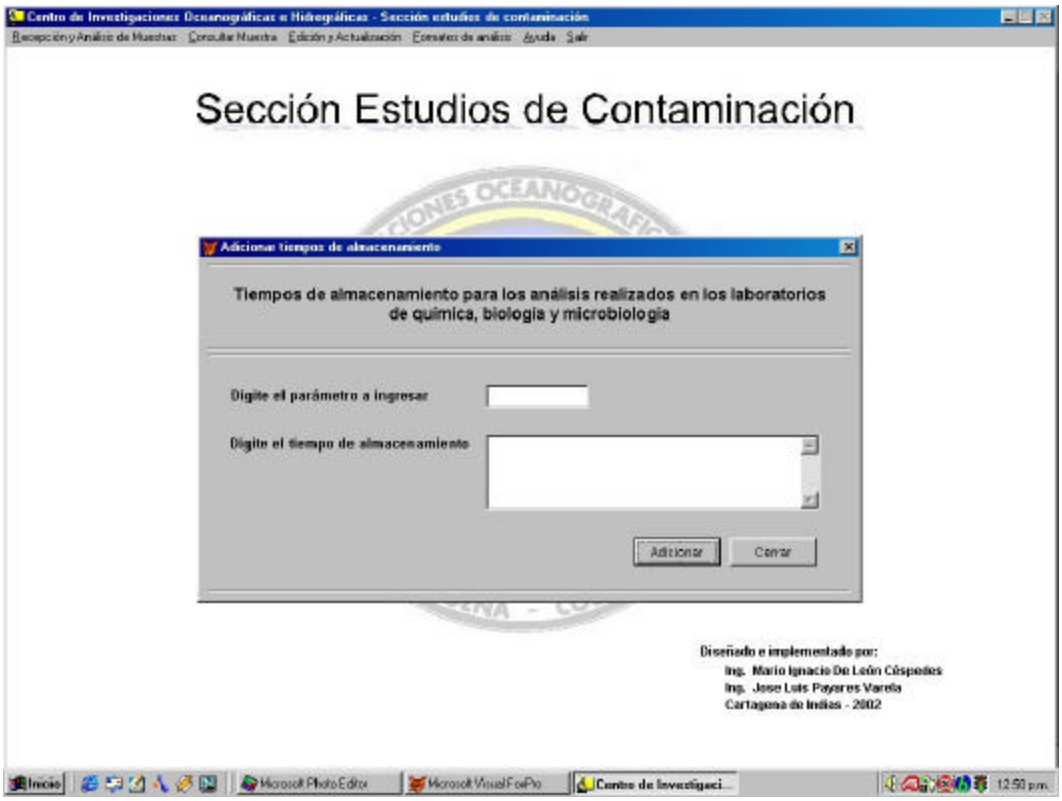

Figura 21 : Adicionar tiempos de almacenamiento.

Este formulario permite añadir un nuevo tiempo de almacenamiento de las muestras. Estos datos son almacenados en la tabla *almacenamiento* por medio del siguiente código que se encuentra asociado al procedimiento *click* del botón adicionar.

```
sMySql="INSERT INTO almacenamiento (parametros, tiempo) values
(?ThisForm.txtparametros.value, ?ThisForm.Txttiempoalma.value)"
sqlexec(gnconn, sMySql)
```

- **Opción Adicionar procedimiento**

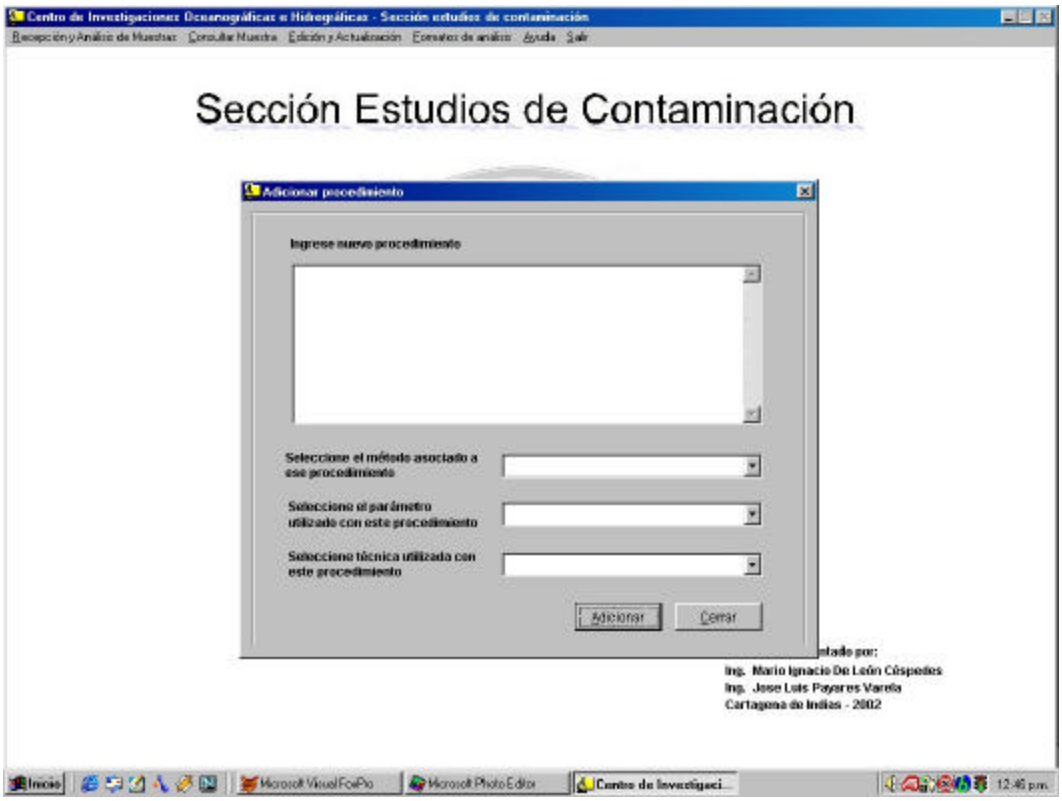

Figura 22 : Adicionar procedimientos.

En este formulario se pude ingresar un nuevo procedimiento para cada uno de los parámetros o análisis que son realizados a las muestras. Cuando se inicia el proceso de carga de este formulario, se ejecuta el código asignado al procedimiento *INIT*, el cual permite visualizar el listado de los métodos, parámetros y técnicas que se encuentran almacenadas en la tabla *parametros*. El código dispuesto para tal fin es el siguiente.

*muestra en formulario el listado de los metodos

sMySql='select metodo from parametros'

```
SQLEXEC(gnconn, sMySql, 'cMetodo')

*muestra en formulario el listado de los parámetros
sMySql='select Nombre from parametros'
SQLEXEC(gnconn, sMySql, 'cParametros')

*muestra en formulario el listado de las tecnicas
sMySql='select Tecnica from parametros'
SQLEXEC(gnconn, sMySql, 'cTecnica')
```

Para almacenar este nuevo procedimiento, se procede a presionar sobre el botón Adicionar, el cual tiene asignado al procedimiento *click* el siguiente código.

```
sMySql="UPDATE parametros SET procedimiento=?ThisForm.Edit1.value WHERE
nombre=?ThisForm.Combo2.value and tecnica=?ThisForm.Combo3.value and
metodo=?ThisForm.Combo1.value"
sqlexec(gnconn, sMySql)
```

- **Opción Adicionar Técnica**

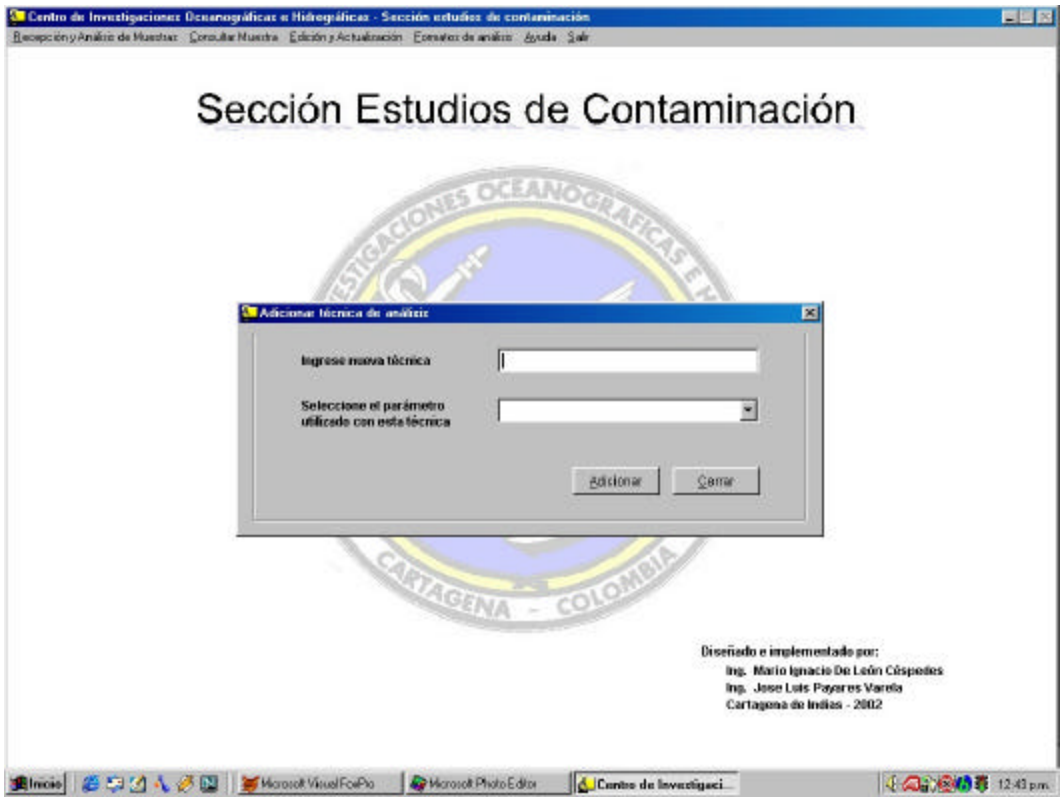

Figura 23 : Adicionar técnicas

En este formulario se pude ingresar una nueva Técnica para cada uno de los parámetros o análisis que son realizados a las muestras. Cuando se inicia el proceso de carga de este formulario, se ejecuta el código asignado al procedimiento *INIT*, el cual permite visualizar el listado de los parámetros que se encuentran almacenadas en la tabla *parametros*. El código dispuesto para tal fin es el siguiente.

```
*muestra en formulario el listado de los parámetros
sMySql='select Nombre from parametros'
SQLEXEC(gnconn, sMySql, 'cParametros')
```

Para almacenar esta nuevo técnica, se procede a presionar sobre el botón Adicionar, el cual tiene asignado al procedimiento *click* el siguiente código.

```
sMySql="UPDATE parametros SET tecnica=?ThisForm.Edit1.value WHERE
nombre=?ThisForm.Combo1.value"
        sqlexec(gnconn, sMySql)
```

- **Opción Añadir, modificar y/o eliminar estación de muestreo**

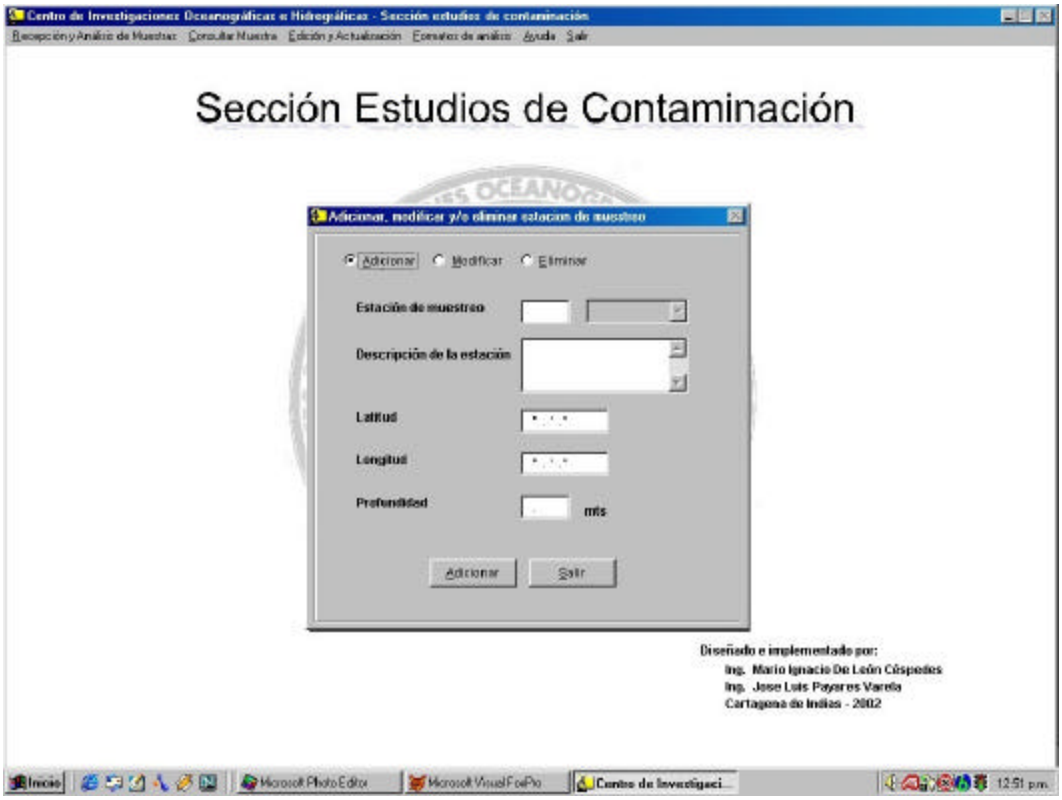

Figura 24 : Adicionar, modificar y/o eliminar estación de muestreo.

Este formulario permite Adicionar, Modificar y/o Eliminar una o varias estaciones de muestreo. Si se desea Adicionar una estación de muestreo, simplemente se selecciona la

opción Adicionar y se ingresan los datos relacionados a la estación de muestreo que se desea ingresar. Esta información es almacenada en la tabla *codestacion* una vez se ha presionado sobre el botón Adicionar. Para ingresar una nueva estación de muestreo es necesario primero verificar que se haya ingresado previamente el sitio de muestreo a la cual pertenece la nueva estación. El código necesario para lograr este fin es el siguiente.

```
Adsitiom=SUBSTR(ALLTRIM(ThisForm.TxtEstacion.value),1,2)
sMySql='Select sitio from codsitiomuestreo'
SQLEXEC(gnconn,sMySql,'cSitio')
DO WHILE .T.
        IF EOF( )
*Muestra mensaje preguntado si la nueva estación a añadir pertenece o no a un Sitio de Muestreo
cMessageTitle = 'Error...'
cMessageText = 'La estación a añadir no pertenece a uno de los Sitios de muestreo existentes!'+(CHR(13))+'Primero debe añadir el Sitio de muestreo al que pertenece esta estación'
nDialogType = 0 + 16 + 0
*  0 = Boton Aceptar.
*  16 = Icono del signo de error.
*  0 = El primer botón es el predeterminado.
nAnswer = MESSAGEBOX(cMessageText, nDialogType, cMessageTitle)
DO CASE
        CASE nAnswer = 1 &&Respuesta afirmativa
```

```
                    Thisform.release
ENDCASE
EXIT
ENDIF
    IF Adsitiom=cSitio.sitio
        *Adicionar Estación
      IF ThisForm.Optiongroup1.OptAdicionar.Value=1
        sMySql="Insert into codestacion(Estacion,Latitud,Longitud,Profundidad,Descripcion)
         values(?ThisForm.TxtEstacion.value,";
         +"?ThisForm.TxtLatitud.value,?ThisForm.TxtLongitud.value,
         ?ThisForm.TxtProfundidad.value, ?ThisForm.TxtDesEstacion.value)"
         SQLEXEC(gnconn,sMySql)
         ThisForm.TxtEstacion.value=''
         ThisForm.TxtDesEstacion.value=''
         ThisForm.TxtLongitud.value=''
         ThisForm.TxtLatitud.value=''
         ThisForm.CmbEstacion.value=''
         ThisForm.TxtProfundidad.value=''

*Muestra mensaje de que la estación fue añadida satisfactoriamente
cMessageTitle = 'Registro guardado...'
cMessageText = 'Estación guardada satisfactoriamente'+(CHR(13))+'Desea añadir otra
estación?'
nDialogType = 4 + 32 + 0
```

```
*  4 = Botones SI/NO.
*  32 = Icono del signo de interrogación.
*  0 = El primer botón es el predeterminado.
nAnswer = MESSAGEBOX(cMessageText, nDialogType, cMessageTitle)
DO CASE
            CASE nAnswer = 7 &&NO
                  Thisform.release
                  EXIT
             ENDCASE
             EXIT
ENDIF
******************
SKIP
LOOP
ENDIF
SKIP
ENDDO
```

Si se desea modificar los datos de una estación de muestreo, simplemente se selecciona la opción Modificar, para que se active la lista despegable con el listado de todas las estaciones de muestreo que ha sido ingresadas previamente. La visualización de las estaciones de muestreo en la lista despegable se logra con las siguientes instrucciones que son activadas en el procedimiento *INIT* cuando se realiza la carga del formulario.

```
sMySql='select * from codestacion'
SQLEXEC(gnconn, sMySql, 'cCodEstacion')
```

Cuando se escoge alguna de las estaciones de muestreo que aparecen en la lista, se activa el procedimiento *Click* de este objeto, para que se visualicen los datos de la estación seleccionada. El código para lograr este fin es el siguiente.

```
local sCodEstacion
sCodEstacion = alltrim(This.value)

sMySql="SELECT * FROM codestacion WHERE Estacion = ?sCodEstacion"
SQLEXEC(gnconn,sMySql,'cTempEstacion')

go top && Va al inicio de la tabla cTempEstacion
ThisForm.TxtDesEstacion.value=cTempEstacion.Descripcion
ThisForm.TxtLongitud.value=cTempEstacion.Longitud
ThisForm.TxtLatitud.value=cTempEstacion.Latitud
ThisForm.TxtProfundidad.value=cTempEstacion.Profundidad
```

Una vez se han hecho los cambios necesarios a la información de la estación de muestreo visualizada, se procede a guardar estos cambios. Para hacerlo se presiona sobre el botón guardar que ahora se llama Modificar, este cambio de nombre se hace con el siguiente código, cuando se selecciona la opción Modificar.

```
ThisForm.BtnGuardar.caption='\<Modificar'
```

Cuando se presiona sobre el botón modificar se activa el código asociado al procedimiento *click* necesario para lograr tal fin. Los cambios realizados se actualiza la tabla *codestacion*.

```
*Modificar estación
IF ThisForm.Optiongroup1.OptModificar.Value=1
sMySql="UPDATE codestacion SET Estacion=?ThisForm.CmbEstacion.value,
Latitud=?ThisForm.TxtLatitud.value, ";
+"Longitud=?ThisForm.TxtLongitud.value, Profundidad=?ThisForm.TxtProfundidad.value,
Descripcion=?ThisForm.TxtDesEstacion.value ";
+"WHERE estacion=?sCodEstacion"
SQLEXEC(gnconn,sMySql)
ENDIF
```

Si se desea eliminar una estación de muestreo, simplemente se selecciona la opción Eliminar, para que se active la lista despegable con el listado de todas las estaciones de muestreo que ha sido ingresadas previamente. La visualización de las estaciones de muestreo en la lista despegable se logra con las siguientes instrucciones que son activadas en el procedimiento *INIT* cuando se realiza la carga del formulario.

```
sMySql='select * from codestacion'
SQLEXEC(gnconn, sMySql, 'cCodEstacion')
```

Cuando se escoge alguna de las estaciones de muestreo que aparecen en la lista, se activa el procedimiento *Click* de este objeto, para que se visualicen los datos de la estación seleccionada. El código para lograr este fin es el siguiente.

```
local sCodEstacion
sCodEstacion = alltrim(This.value)

sMySql="SELECT * FROM codestacion WHERE Estacion = ?sCodEstacion"
SQLEXEC(gnconn,sMySql,'cTempEstacion')

go top && Va al inicio de la tabla cTempEstacion
ThisForm.TxtDesEstacion.value=cTempEstacion.Descripcion
ThisForm.TxtLongitud.value=cTempEstacion.Longitud
ThisForm.TxtLatitud.value=cTempEstacion.Latitud
ThisForm.TxtProfundidad.value=cTempEstacion.Profundidad
```

Una vez se ha decidido eliminar la estación de muestreo visualizada, se procede a eliminar la estación. Para hacerlo se presiona sobre el botón guardar que ahora se llama Eliminar, este cambio de nombre se hace con el siguiente código, cuando se selecciona la opción Eliminar.

```
ThisForm.BtnGuardar.caption='\<Eliminar'
```

Cuando se presiona sobre el botón Eliminar se activa el código asociado al procedimiento *click* necesario para lograr tal fin. La estación de muestreo a borrar se elimina de la tabla *codestacion*.

```
*Eliminar estación
IF ThisForm.Optiongroup1.OptEliminar.value=1
```

```
sMySql="DELETE FROM codestacion WHERE Estacion=?ThisForm.CmbEstacion.value"
SQLEXEC(gnconn,sMySql)
ENDIF
```

- **Opción Eliminar Analista**

Figura 25 : Eliminar analista.

En este formulario se pude eliminar un Analista que haya sido ingresado previamente a la tabla *analista*. Cuando se inicia el proceso de carga de este formulario, se ejecuta el código asignado al procedimiento *INIT*, el cual permite visualizar el listado de los analistas que se encuentran almacenados en la tabla *analista*. El código dispuesto para tal fin es el siguiente.

```
*muestra en formulario el codigo del analista
sMySql='select codigo from analista'
SQLEXEC(gnconn, sMySql, 'cCodAnalista')
```

Cuando se escoge alguno de los códigos de los analistas que aparecen en la lista despegable, se activa el procedimiento *Click* de este objeto, para que se visualicen los datos del analista seleccionado. El código para lograr este fin es el siguiente.

```
Local sCodigo
sCodigo = alltrim(This.value)
SQLEXEC(gnconn,"SELECT * FROM analista WHERE codigo = ?sCodigo ","cTemporal")

go top && Va al inicio de la tabla temporal
ThisForm.NomAnalista.value=cTemporal.Nombre
ThisForm.ApeAnalista.value=cTemporal.Apellido
ThisForm.CarAnalista.value=cTemporal.Cargo
```

Una vez se ha escogido y se ha visualizado la información del analista a eliminar, se procede a presionar sobre el botón *Eliminar Analista*, para que se active el procedimiento *click* con el código necesario para lograr tal fin.

```
sMySql="Delete from analista where codigo=?ThisForm.CmbCodAnalista.value"
sqlexec(gnconn, sMySql)
```

- **Opción Eliminar Método de Análisis**

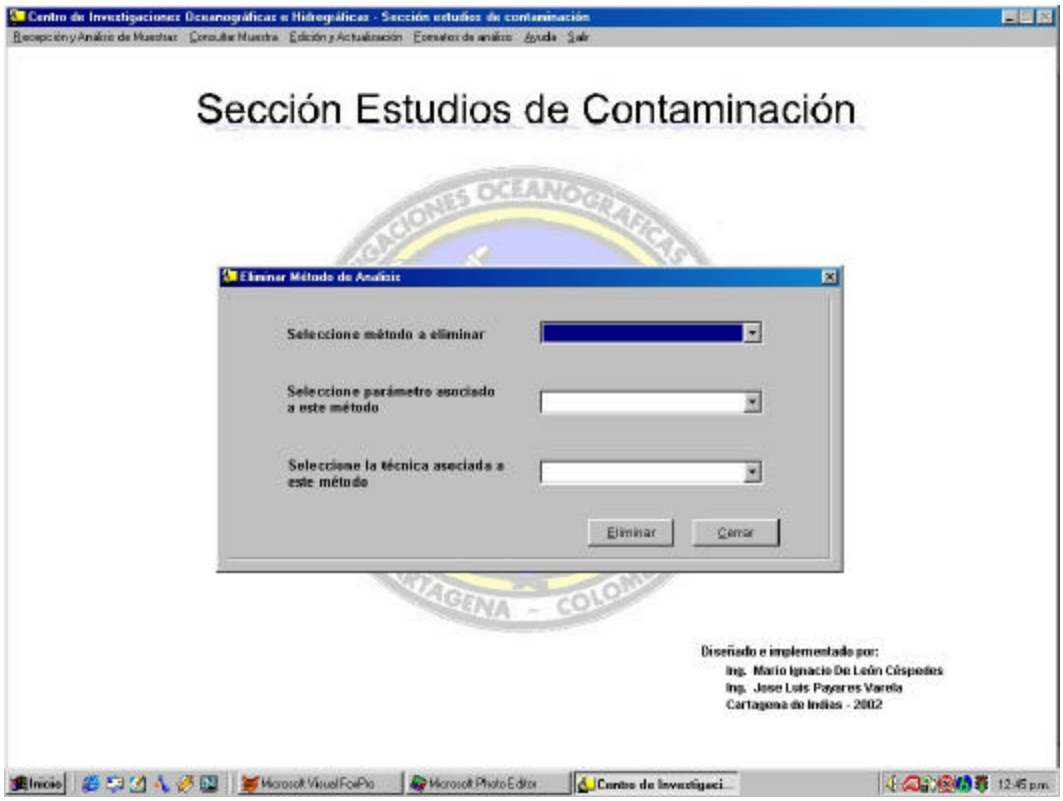

Figura 26 : Eliminar método de análisis.

En este formulario se pude eliminar un método de análisis para cada uno de los parámetros o análisis que son realizados a las muestras. Cuando se inicia el proceso de carga de este formulario, se ejecuta el código asignado al procedimiento *INIT*, el cual permite visualizar el listado de los métodos, parámetros y técnicas que se encuentran almacenadas en la tabla *parametros*. El código dispuesto para tal fin es el siguiente.

```
*muestra en formulario el listado de los metodos
sMySql='select metodo from parametros'
SQLEXEC(gnconn, sMySql, 'cMetodo')
```

```
*muestra en formulario el listado de los parámetros
sMySql='select Nombre from parametros'
SQLEXEC(gnconn, sMySql, 'cParametros')

*muestra en formulario el listado de las tecnicas
sMySql='select Tecnica from parametros'
SQLEXEC(gnconn, sMySql, 'cTecnica')
```

Para eliminar este Método de Análisis, se procede a presionar sobre el botón Eliminar, el cual tiene asignado al procedimiento *click* el siguiente código.

```
sMySql="UPDATE parametros SET metodo = '' where
nombre=?ThisForm.cmbParametros.value '';
+"and tecnica=?ThisForm.cmbTecnica.value and metodo=?ThisForm.cmbMetodo.value"
sqlexec(gnconn, sMySql)
```

- **Opción Eliminar Sitio de muestreo**

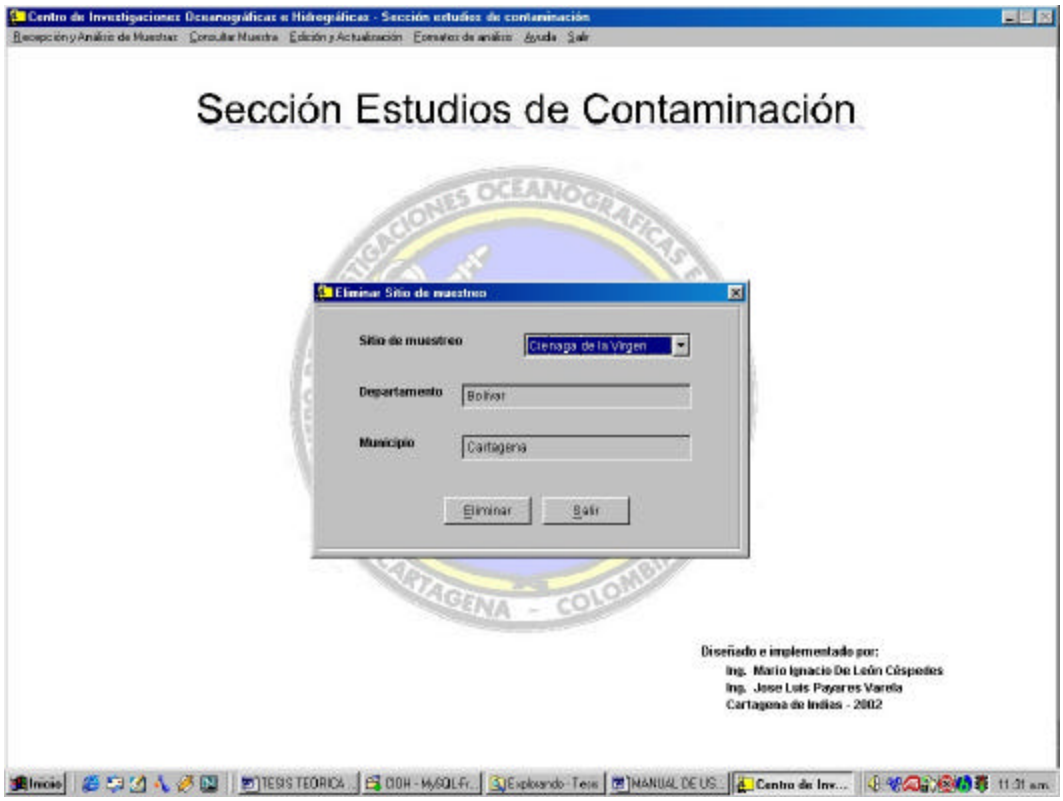

Figura 27 : Eliminar sitio de muestreo.

En este formulario se puede escoger un Sitio de Muestreo que se desee eliminar. Cuando se inicia el proceso de carga de este formulario, se ejecuta el código correspondiente al procedimiento *INIT,* con las siguientes instrucciones se solicitan los datos del sitio de muestreo a la tabla *codsitiomuestreo* con el fin de que puedan ser visualizados inmediatamente se carga el formulario.

```
*muestra en formulario los sitios de muestreo
sMySql='select Nombre from codsitiomuestreo'
SQLEXEC(gnconn, sMySql,'cSitMuestreo')
```

Una vez se ha seleccionado en la lista despegable el Sitio de Muestreo que se desea eliminar, se activa el procedimiento *click* de esta, el cual permite con el siguiente codigo mostrar la información del Sitio de Muestreo a Eliminar.

```
local sSitMuestreo
sSitMuestreo = alltrim(This.value)
sMySql="SELECT * FROM codsitiomuestreo WHERE Nombre = ?sSitMuestreo"
SQLEXEC(gnconn,sMySql,'cTempSitMuestreo')

go top && Va al inicio de la tabla cTempEstacion
ThisForm.TxtDptoSitMuestreo.value=cTempSitMuestreo.Departamento
ThisForm.TxtMunSitMuestreo.value=cTempSitMuestreo.Municipio
```

Después de haber seleccionado y visualizado la información del sitio de muestreo a eliminar, se procede a presionar el botón Eliminar, el cual activa en su procedimiento *click*, el siguiente código necesario para eliminar de la tabla *codsitiomuestreo* el sitio de muestreo seleccionado.

```
sMySql="Delete From codsitiomuestreo WHERE nombre=?ThisForm.cmbSitMuestreo.value"
SQLEXEC(gnconn,sMySql)
```

- **Opción Eliminar Técnica**

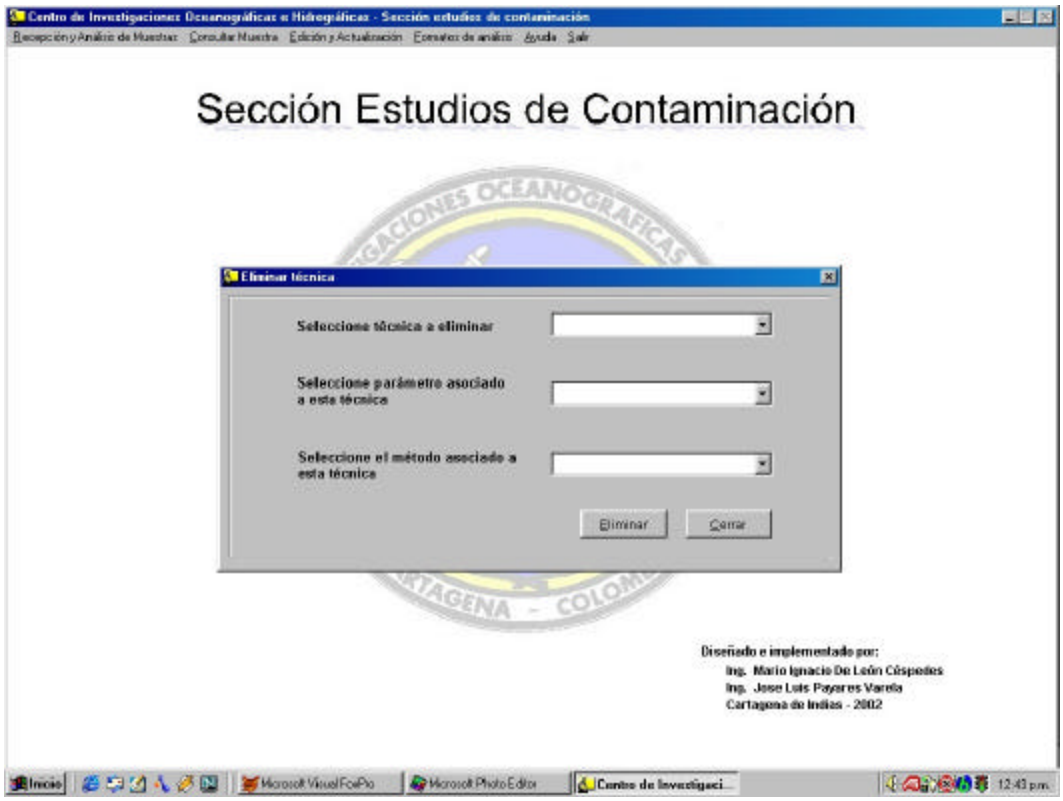

Figura 28 : Eliminar técnicas.

En este formulario se pude eliminar una Técnica de análisis para cada uno de los parámetros o análisis que son realizados a las muestras. Cuando se inicia el proceso de carga de este formulario, se ejecuta el código asignado al procedimiento *INIT*, el cual permite visualizar el listado de los métodos, parámetros y técnicas que se encuentran almacenadas en la tabla *parametros*. El código dispuesto para tal fin es el siguiente.

```
*muestra en formulario el listado de los metodos
sMySql='select metodo from parametros'
SQLEXEC(gnconn, sMySql, 'cMetodo')
```

```
*muestra en formulario el listado de los parámetros
sMySql='select Nombre from parametros'
SQLEXEC(gnconn, sMySql, 'cParametros')

*muestra en formulario el listado de las tecnicas
sMySql='select Tecnica from parametros'
SQLEXEC(gnconn, sMySql, 'cTecnica')
```

Para eliminar esta Técnica de Análisis, se procede a presionar sobre el botón Eliminar, el cual tiene asignado al procedimiento *click* el siguiente código.

```
NombreTemp=Alltrim(ThisForm.cmbParametros.value)
sMySql="UPDATE parametros SET tecnica = " WHERE
nombre=?ThisForm.cmbParametros.value and tecnica=?ThisForm.CmbTecnica.value and
Metodo=?ThisForm.CmbMetodo.value"
sqlexec(gnconn,sMySql)
```

- **Opción Agregar Sitio de Muestreo**

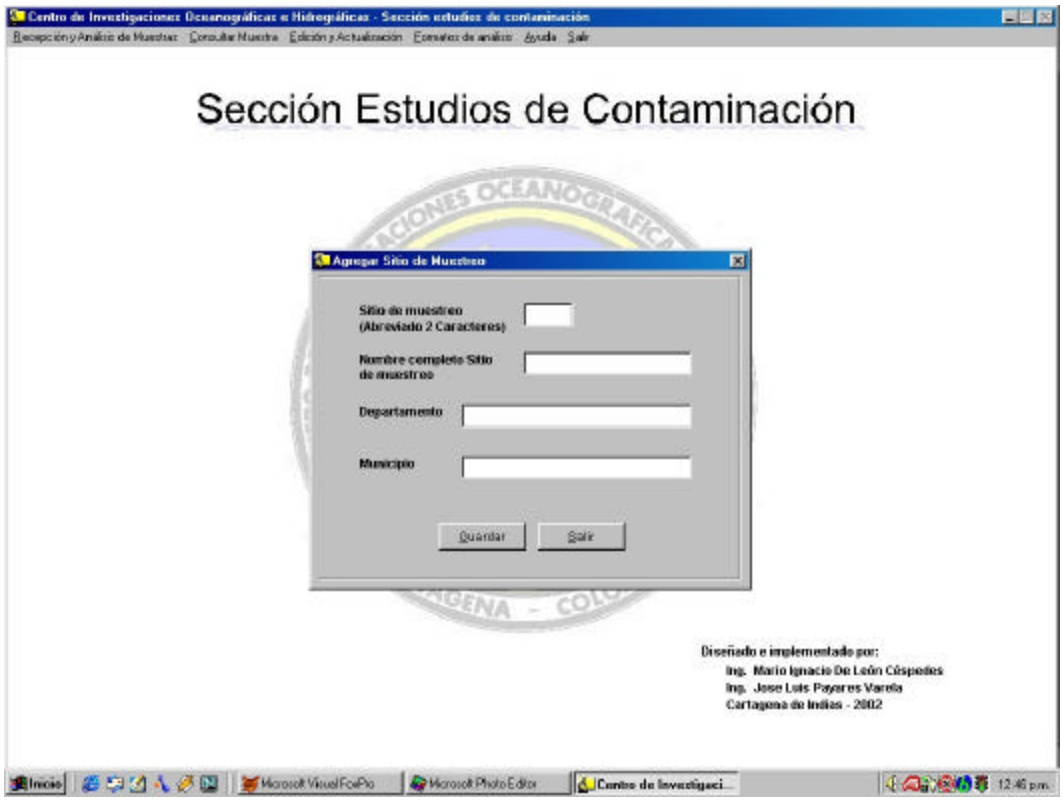

Figura 29 : Agregar sitio de muestreo.

En esta forma se puede añadir o agregar un Sitio de Muestreo; para realizar esto simplemente se ingresan los datos del nuevo Sitio de Muestreo y se presiona sobre el botón Guardar para que se active el procedimiento *click* el cual, por medio del siguiente código guarda los datos del Nuevo sitio de Muestreo en la tabla *codsitiomuestreo*.

```
sMySql="Insert into codsitiomuestreo(sitio, nombre, departamento, municipio)
values(?ThisForm.TxtSitMuestreo.value, ?ThisForm.TxtNomSitMuestreo.value, “;
+”?ThisForm.TxtDptoSitMuestreo.value, ?ThisForm.TxtMunSitMuestreo.value)"
SQLEXEC(gnconn,sMySql)
```

- **Opción Adicionar Analista**

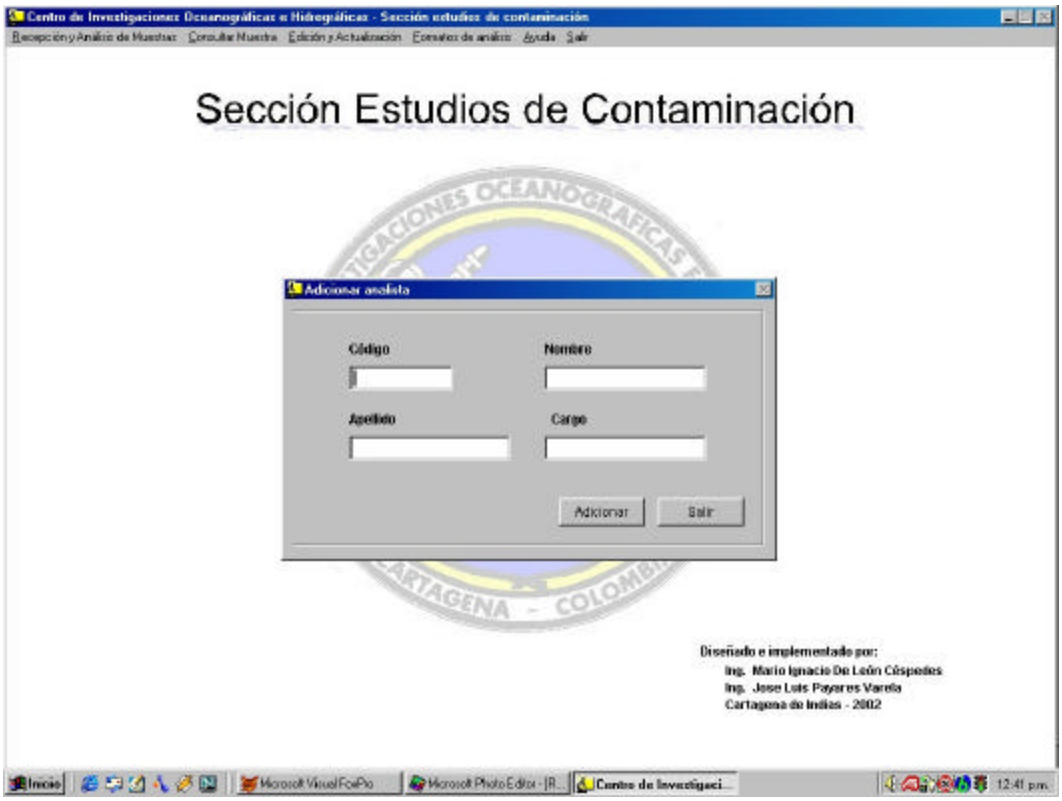

Figura 30 : Adicionar analista.

En el siguiente formulario se puede ingresar un Nuevo analista a la tabla *analista;* simplemente se ingresan los datos del nuevo analista, y se presiona sobre el botón Adicionar para que se active el siguiente código asignado al procedimiento *click* de este objeto.

```
sMySql="insert into analista(Codigo, nombre, apellido, cargo)
values(?ThisForm.CodAnalista.value, ?ThisForm.NomAnalista.value, ";
+"?ThisForm.ApeAnalista.value, ?ThisForm.CarAnalista.value)"
sqlexec(gnconn, sMySql)
```

- **Opción Modificar Tiempo de Almacenamiento**

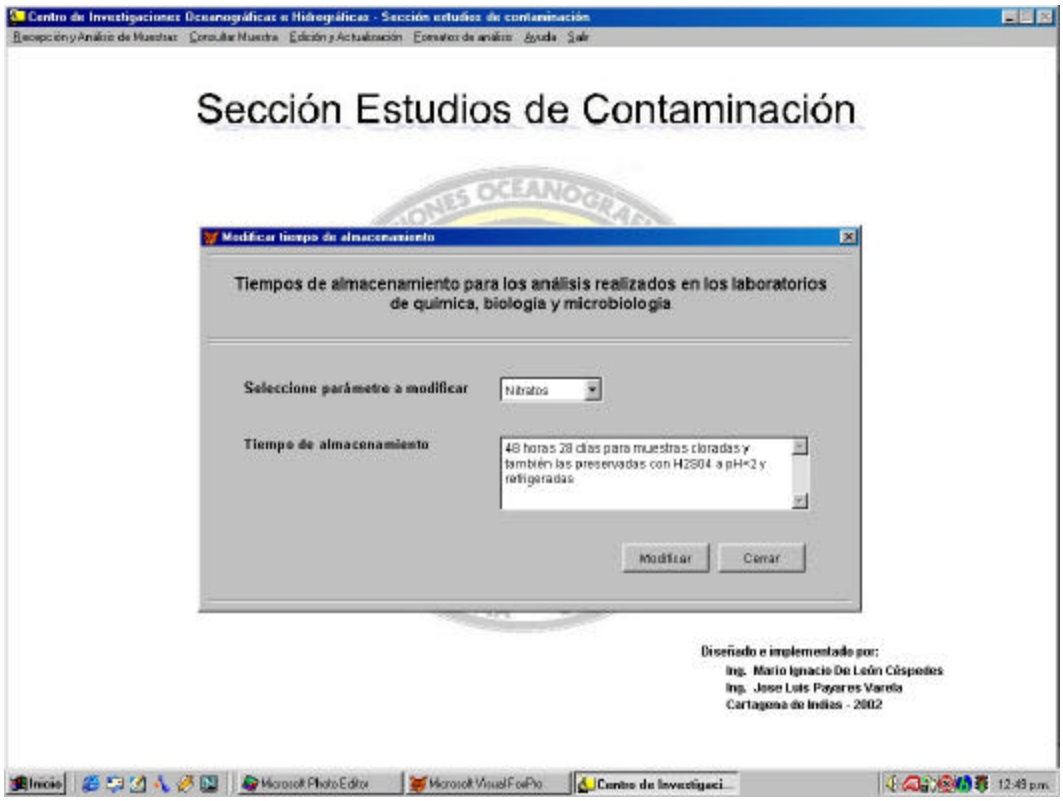

Figura 31 : Modificar tiempos de almacenamiento.

En este formulario se puede modificar el tiempo de almacenamiento de las muestras. Cuando se inicia el proceso de carga de este formulario se ejecuta el código correspondiente al procedimiento *INIT*, en el que con las siguientes instrucciones solicita a la tabla *almacenamiento* el listado de los parámetros que han sido previamente ingresados, con el fin de que puedan ser visualizados inmediatamente se carga el formulario.

*muestra en formulario los parámetros y los tiempos de almacenamiento

```
sMySql='select * from almacenamiento'
SQLEXEC(gnconn, sMySql, 'cAlma')
```

Cuando se selecciona en la lista despegable alguno de los parámetros que allí apararecen, se ejecuta el siguiente código en el procedimiento *click* de ese objeto, el cual permite que se visualice el tiempo de almacenamiento para el parámetro seleccionado.

```
ThisForm.Txttiempoalma.value=cAlma.tiempo
```

Cuando se ha modificado el tiempo de almacenamiento para el parámetro escogido, se procede a presionar sobre el botón Modificar para que se actualizar la tabla *almacenamiento* con los nuevos cambios realizados y de esta manera activar el siguiente código que se encuentra asociado al procedimiento *click* de ese objeto.

```
sMySql="UPDATE almacenamiento SET tiempo=?ThisForm.Txttiempoalma.value WHERE
parametros=?ThisForm.Cmbmodtiempo.value"
sqlexec(gnconn, sMySql)
```

- **Opción Modificar Muestra**

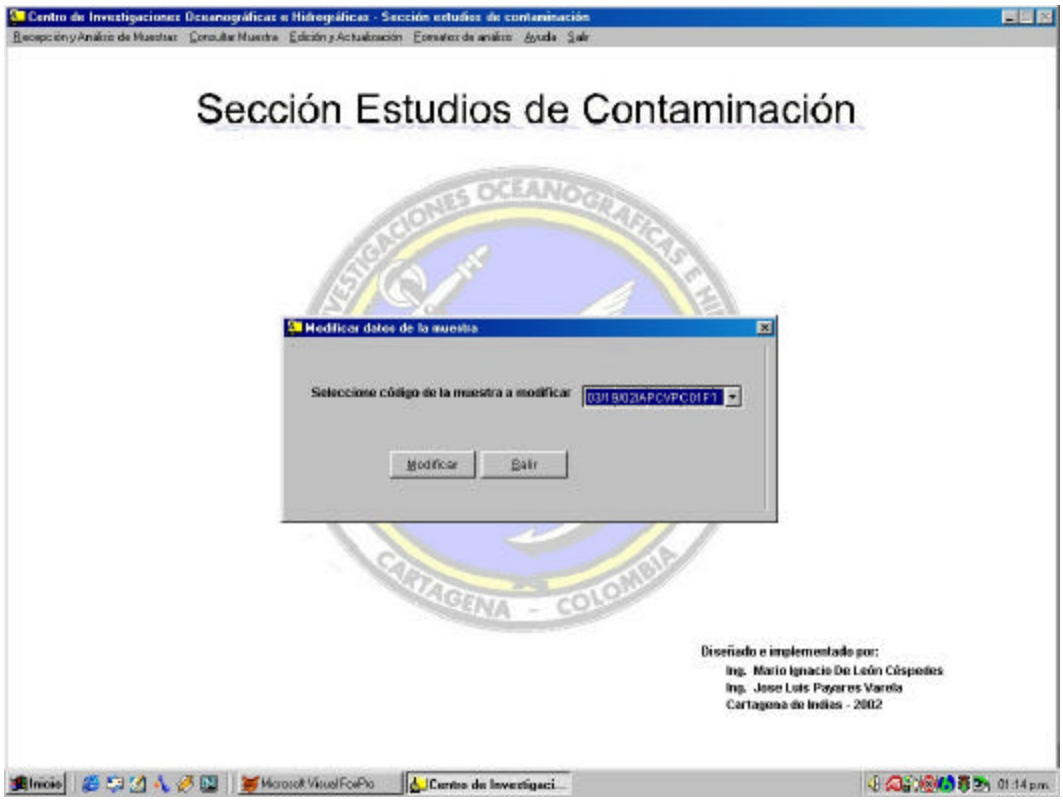

Figura 32 : Modificar muestra.

En el siguiente formulario aparecen el listado de los códigos de todas las muestras que han sido ingresadas. Cuando se inicia el proceso de carga de este formulario, se ejecuta el siguiente código correspondiente al procedimiento *INIT*, el cual permite que se puedan visualizar los códigos inmediatamente se carga el formulario.

*muestra en formulario el codigo de la muestra

sMySql='select codigo from muestra'

```
SQLEXEC(gnconn, sMySql, 'cMuestra')
```

Cuando se ha seleccionado el código de la muestra que se desea modificar, se procede a presionar sobre el botón Modificar, para que se active le procedimiento *click* de este botón, y se ejecute el siguiente código.

```
*Variable que indica si se ha seleccionado modificar datos de la muestra (0:No seleccionado,
1:Seleccionado)
SelModMuestra=1
TempCodMuestra=ThisForm.CmbCodMuestra.value
set defa to C:\quimicacioh\
Thisform.release
do form frmprincipal.scx
```

Con el anterior código se activa en uno (1) la variable *SelModMuestra* para que cuando se cargue la forma *frmprincipal* le indique a ésta que muestre los datos del código de la muestra seleccionada para que sean modificados.

- **Opción Modificar Información Analista**

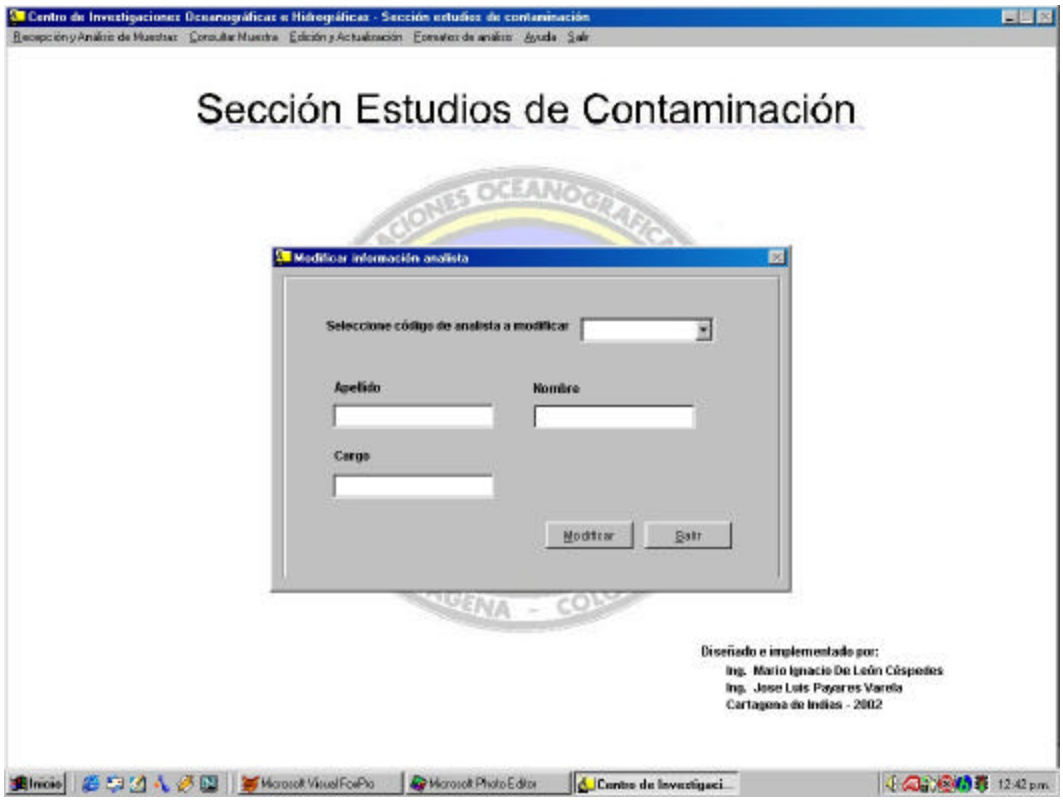

Figura 33 : Modificar información del analista

Con el siguiente formulario se puede modificar la información del analista que se encuentra almacenada en la tabla *analista*. Cuando se inicia el proceso de carga del formulario, se ejecuta el siguiente código correspondiente al procedimiento *INIT,* el cual permite la visualización inmediata de los códigos de los analista en la lista despegable que se encuentra en el formulario.

```
*muestra en formulario el codigo del analista
sMySql='select codigo from analista'
SQLEXEC(gnconn, sMySql, 'cCodAnalista')
```

Una vez se ha seleccionado en la lista despegable el código del analista al cual se le desea modificar su información, se ejecuta el siguiente código que se encuentra asociado al procedimiento *Click* de este objeto, el cual permite visualizar en el formulario la información del analista seleccionado, para de estar formar proceder a realizar las modificaciones en la información del mismo.

```
Local sCodigo
sCodigo = alltrim(This.value)
sMySql="SELECT * FROM analista WHERE codigo = ?sCodigo "
SQLEXEC(gnconn,sMySql,'cTemporal1')

go top && Va al inicio de la tabla temporal
ThisForm.NomAnalista.value=cTemporal1.Nombre
ThisForm.ApeAnalista.value=cTemporal1.Apellido
ThisForm.CarAnalista.value=cTemporal1.Cargo
```

Después de haber realizado las modificaciones pertinentes a la información del analista, se procede a actualizar la tabla *analista*, presionando sobre le botón Modificar, para que se ejecute el siguiente código asociado al procedimiento *click*.

```
sMySql="Update analista set Apellido=?ThisForm.ApeAnalista.value,
Nombre=?ThisForm.NomAnalista.value, Cargo=?ThisForm.CarAnalista.value ";
+"where codigo=?ThisForm.CmbCodAnalista.value"
sqlexec(gnconn, sMySql)
```

- **Opción Modificar Método**

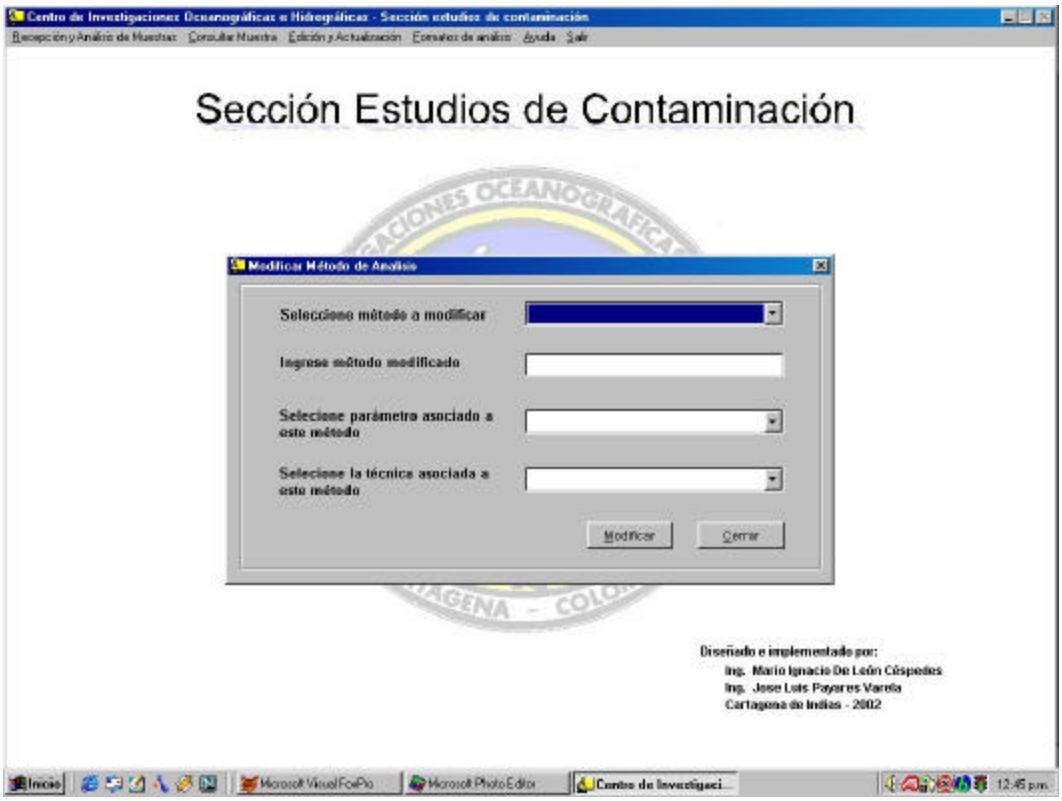

Figura 34 : Modificar métodos.

En este formulario se pude modificar el método de análisis para cada uno de los parámetros o análisis que son realizados a las muestras. Cuando se inicia el proceso de carga de este formulario, se ejecuta el código asignado al procedimiento *INIT*, el cual permite visualizar el listado de los métodos, parámetros y técnicas que se encuentran almacenadas en la tabla *parametros*. El código dispuesto para tal fin es el siguiente.

*muestra en formulario el listado de los metodos

```
sMySql='select metodo from parametros'
SQLEXEC(gnconn, sMySql, 'cMetodo')

*muestra en formulario el listado de los parámetros
sMySql='select Nombre from parametros'
SQLEXEC(gnconn, sMySql, 'cParametros')

*muestra en formulario el listado de las tecnicas
sMySql='select Tecnica from parametros'
SQLEXEC(gnconn, sMySql, 'cTecnica')
```

Una vez se ha seleccionado el Método de análisis a modificar y se ha ingresado el nuevo método, así como la selección del parámetro y la técnica asociada a ese Método, se procede a presionar sobre el botón Modificar, el cual tiene asignado al procedimiento *click* el siguiente código.

```
sMySql="Update parametros set metodo=?ThisForm.MetodoModificado.value ";
+"where Nombre=?ThisForm.CmbParametro.value and tecnica=?ThisForm.CmbTecnica.value
and metodo=?ThisForm.CmbMetodo.value"
sqlexec(gnconn, sMySql)
```

- **Opción Modificar Técnica**

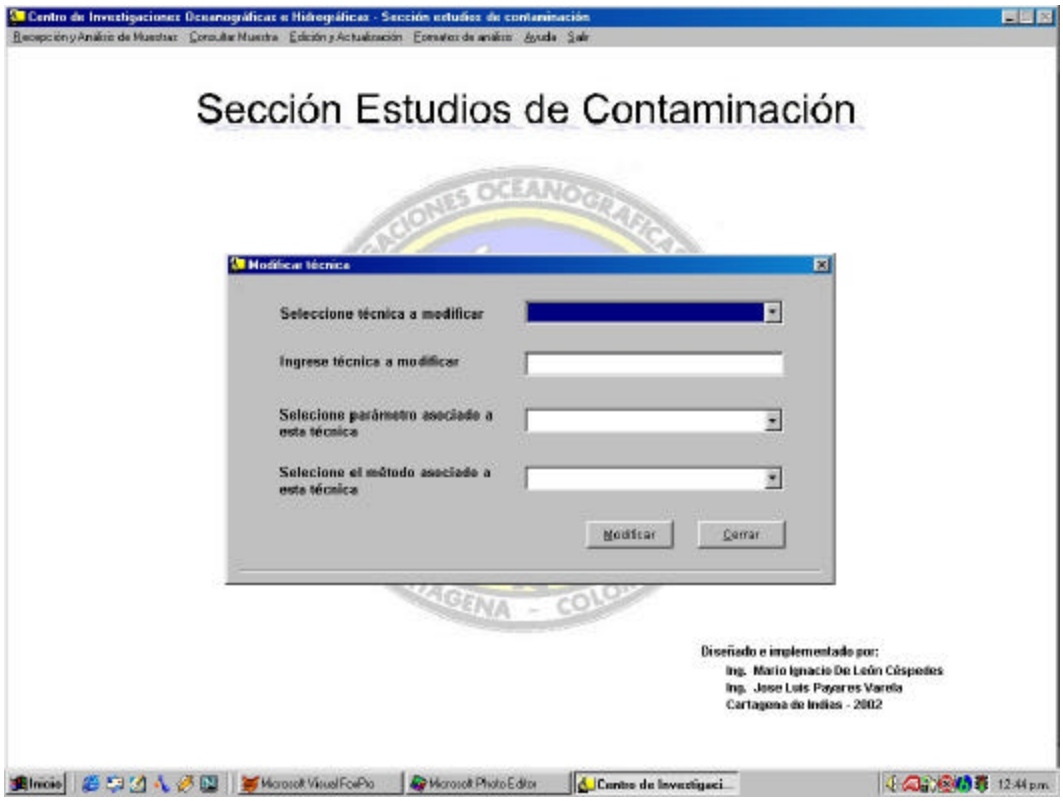

Figura 35 : Modificar técnicas.

En este formulario se pude modificar la técnica de análisis para cada uno de los parámetros o análisis que son realizados a las muestras. Cuando se inicia el proceso de carga de este formulario, se ejecuta el código asignado al procedimiento *INIT*, el cual permite visualizar el listado de los métodos, parámetros y técnicas que se encuentran almacenadas en la tabla *parametros*. El código dispuesto para tal fin es el siguiente.

```
*muestra en formulario el listado de los metodos
sMySql='select metodo from parametros'
SQLEXEC(gnconn, sMySql, 'cMetodo')
```

```
*muestra en formulario el listado de los parámetros
sMySql='select Nombre from parametros'
SQLEXEC(gnconn, sMySql, 'cParametros')

*muestra en formulario el listado de las tecnicas
sMySql='select Tecnica from parametros'
SQLEXEC(gnconn, sMySql, 'cTecnica')
```

Una vez se ha seleccionado la técnica a modificar y se ha ingresado la nueva técnica, así como la selección del parámetro y el método asociado a esa Técnica, se procede a presionar sobre el botón Modificar, el cual tiene asignado al procedimiento *click* el siguiente código.

```
sMySql="Update parametros set Tecnica=?ThisForm.EdtTecnica.value ";
+"where nombre=?ThisForm.CmbParametros.value and
Tecnica=?ThisForm.CmbTecnica.value and Metodo=?ThisForm.CmbMetodo.value"
sqlexec(gnconn, sMySql)
```

La *Formatos de análisis* es otra de las opciones que se encuentra en el menú principal.

Figura 36 : Opción Formatos de Análisis.

Al escoger esta opción se muestra al usuario el siguiente formulario.

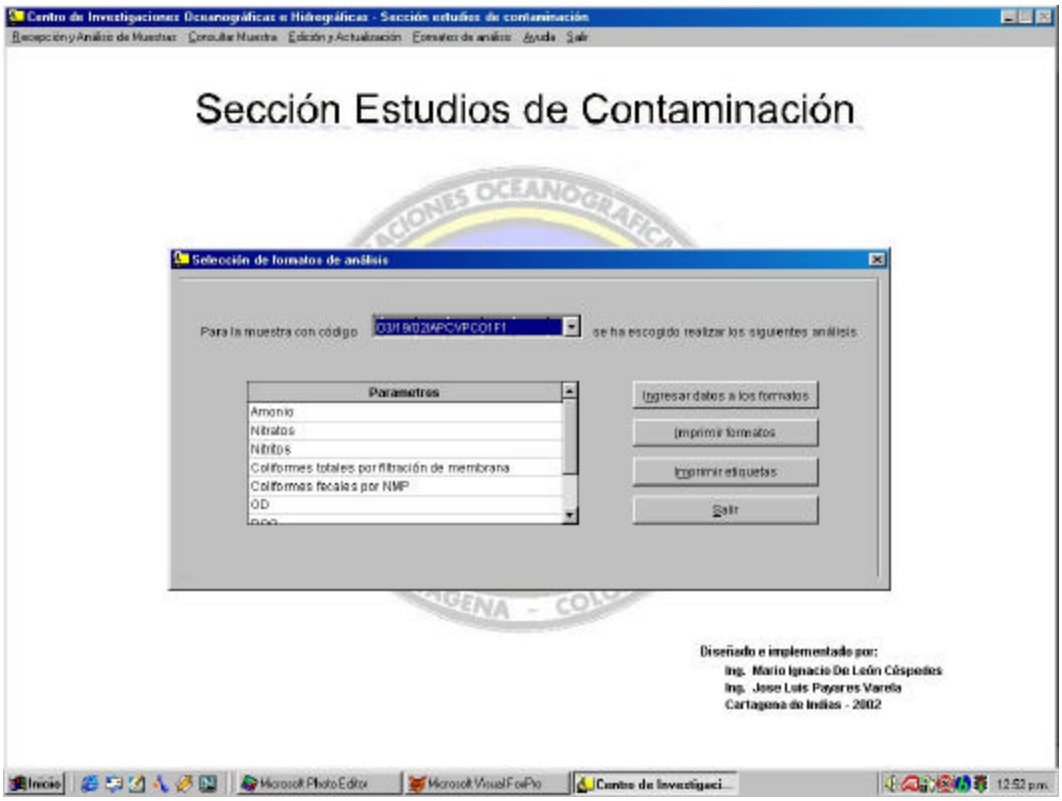

Figura 37 : Selección código para imprimir formatos.

En este formulario se pueden escoger los parámetros asociados a cada unos de los códigos de las muestras que han sido ingresadas al sistema. Cuando se inicia el proceso de carga de este formulario, se ejecuta el código asignado al procedimiento *INIT*, el cual permite visualizar el listado de los códigos de las muestras que se encuentran almacenadas en la tabla *muestra*. El código dispuesto para tal fin es el siguiente.

```
*muestra en formulario el codigo de la muestra
sMySql='select codigo from muestra'
SQLEXEC(gnconn, sMySql, 'muestra')
```

El listado de los códigos aparece en una lista despegable desde donde el usuario puede seleccionar el código de la muestra deseada para visualizar los parámetros asociados a ese código. Cuando se escoge el código de la muestra se ejecuta el siguiente código, asociado al procedimiento *click.* de ese objeto.

```
PUBLIC nue_cod
Local sCodigo

sCodigo = alltrim(This.value)
if TYPE("CodTemp") = "U"
   * generamos el CodTemp a partir del valor actual del combo menos la fecha
   CodTemp = SUBSTR(sCodigo,9,11)
endif

SQLEXEC(gnconn,"SELECT nombre FROM parametros WHERE " + CodTemp + " =
?sCodigo",'cTemporal')
go top && Va al inicio de la tabla temporal
Thisform.Grid3.RecordSource='cTemporal'

With ThisForm.Grid3
    .Column1.Header1.alignment = 2
    .Column1.Header1.FontBold = .T.
    .Column1.Header1.Caption = "Parametros"
    .Column1.Width = 304
EndWith
```

5. PHP

PHP fue creado por Rasmus Lerdorf a finales de 1994, aunque no hubo una versión utilizable por otros usuarios hasta principios de 1995. Esta primera versión se llamó, *Personal Home Page Tools*.

Al principio, PHP sólo estaba compuesto por algunas macros que facilitaban el trabajo a la hora de crear una página Web. Hacia mediados de 1995 se creó el analizador sintáctico y se llamó PHP/F1 Versión 2, y sólo reconocía el texto HTML y algunas directivas de MySQL. A partir de este momento, la contribución al código fue pública.

El crecimiento de PHP desde entonces ha sido exponencial, y han surgido versiones nuevas como la actual, PHP3 y la incipiente PHP4.

PHP (Acrónimo de PHP Hipertext Preprocessor) es un lenguaje de programación de páginas web que funciona en el lado del servidor. Técnicamente es un lenguaje interpretado de alto nivel, similar en construcciones léxicas y sintácticas a Perl, C e incluso Java, y embebido en páginas HTML.

Está diseñado especialmente para trabajar con sistemas de bases de datos relaciónales.

El código PHP no se mezcla con las etiquetas HTML, ya que está acotado siempre por los símbolos <? y ?>, de forma que para los programas compositores de páginas web son etiquetas que no soporta y las deja tal y como están. De esta forma, puede trabajarse a la vez en el diseño visual de la página y en la funcionalidad (programación) de la misma cómodamente.

Los navegadores no tienen ningún problema con PHP, ya que nunca llegan a verlo, el código PHP se interpreta para generar la página HTML solicitada antes de ser transmitida al navegador. Eso sí, el navegador debe ser capaz de reconocer las extensiones .php, .php3 y antiguamente .phtml, como documentos de tipo text/html, o tratará de bajar las páginas como si fueran ficheros en vez de visualizarlas.

Afortunadamente, PHP está lo suficientemente extendido para que no tengamos que preocuparnos de ese problema, puesto que los principales navegadores ya reconocen tales extensiones.

5.1 PHP y HTML

Para escribir código PHP dentro de una página html, tenemos varias alternativas:

- Incluir el código entre <? y ?>
- Incluir el código entre <?PHP y ?>
- Incluir el código entre bloques <SCRIPT LANGUAGE="php"> y </SCRIPT>

Una forma manejable de conocer que opciones están disponibles en PHP y que hacen en tu servidor es usando la función *phpinfo()* . Crea un script con lo siguiente:

```
<html>
<body>
<?php
  phpinfo();
?>
</body>
</html>
```

El resultado de la salida estándar de ese código será escrito en esa misma posición de la página html.

Ejemplo:

```
<html>
<body>
<?php
$mivariable = "Este es un ejemplo de cómo utilizar PHP en la Web. Este es mi primer script en PHP";
echo $ mivariable;
?>
</body>
</html>
```

Para que el servidor envíe texto utilizaremos la instrucción echo, aunque también podemos utilizar printf de uso similar al del C o Perl.

Finalmente, vemos que la palabra mivariable comienza con el signo peso ($). Este símbolo le indica a PHP que es una variable. Se le ha asignado un texto a esta variable, pero también pueden contener números o arrays. Es importante recordar que todas las variables comienzan con el signo pesos.

La página anterior, si se salva como ejemplo.phtml y se carga con el navegador, produce como resultado una página HTML Con el texto " Este es un ejemplo de cómo utilizar PHP en la Web. Este es mi primer script en PHP ". Valga decir que para que funcione, es necesario tener instalado un servidor web con soporte para PHP y asociar la interpretación de PHP a la extensión phtml.

Comentarios

Los comentarios en PHP se escriben:

- Con // o # para comentarios de una sola línea.
- Entre /* y */ para comentarios de una o más líneas.

Ejemplo:

```
/*  Título: Mi Primera página PHP
    Autor: Yo
*/
```

```
// Saludamos
echo("¡Hola a todos!<BR>");
```

La ventaja que tiene PHP sobre otros lenguajes de programación que se ejecutan en el servidor (como podrían ser los script CGI Perl), es que nos permite intercalar las sentencias PHP en las paginas HTML, es un concepto algo complicado de entender si no se ha visto nunca como funciona una pagina PHP o ASP.

5.2 Seguridad en PHP.

PHP es un interprete que puede ser incluido en un servidor Web como un módulo o como un CGI binario, con él se pueden realizar accesos a ficheros, conexiones de red, etc.

PHP está diseñado para ser más seguro que cualquier otro lenguaje de programación de CGIs, como Perl o C.

CGI binario

Este método lo que hace es instalar PHP en el directorio *cgi-bin*. Esto permite a PHP reaccionar ante diversos tipos de ataques.

En http, todo lo que se pase detrás del símbolo ? es la línea de argumentos que el interfaz CGI interpreta. Curiosamente, si a un sistema Linux, se le pasa la instrucción /etc/passwd, el sistema intenta ejecutar este comando y puede ser un fallo en la seguridad.

Otro posible ataque, es cuando se intenta acceder a los ficheros del servidor web a los que no se debe tener acceso. Para evitar esto, existen opciones de configuración que redirigen todas las peticiones al intérprete de PHP forzando un chequeo de acceso al fichero que se pide.

Algunas de estas opciones de seguridad son:

Si se activa la opción *disable-force-cgi-redirect* se obliga a que tanto las peticiones del tipo *http://my.host.cgi-bin/php/dir/script.php3* como las peticiones del tipo *http://my.host/dir/script.php3* sean analizadas por el intérprete PHP.

Otras opciones posibles en la configuración es combinar la directiva *Action* y *AddHandler;* mediante estas opciones se configura la redirección de las llamadas para que sean interpretadas.

Esta opción ha sido probada en Apache y a este servidor se refiere.

Action php3-script /cgi-bin/php Addhandler php3-script.php3

La tercera opción es utilizar las directivas *doc_root* y *user_dir*.

Estas directivas se utilizan en servidores Web que no disponen de la facilidad del redireccionamiento.

Supongamos que un script no se ejecuta correctamente, en este caso, el código se muestra en pantalla y esto puede violar la propiedad intelectual de ese script.

Para solucionar esto, se colocan todos los scripts PHP ejecutables en un directorio, que indica la directiva *doc_root* asegurando así que todo lo que esté en ese directorio será ejecutado y nunca mostrado al usuario.

Si esta directiva se combina con *user_dir* se permitirá ejecutar, ante llamadas del tipo: *http://my.host/~user/doc.php3* ficheros que estén en el directorio que indica *user_dir* bajo el directorio */home/user/*.

Otra práctica muy segura es mantener la instalación del intérprete fuera del árbol web. Si esto es así, se deberán hacer los ficheros php ejecutables, modificando los atributos del fichero y además se deberá incluir en la primera línea del script la dirección del intérprete, *#!/usr/local/bin/php* por ejemplo.

Módulo

En el caso de tener PHP instalado como un módulo del servidor Apache, este hereda todas las características del Servidor. Esta opción es la menos utilizada.

5.3 IMPLEMENTACIÓN DE UNA APLICACIÓN PHP / MySQL

Esta aplicación Cliente/Servidor realizada en PHP y MySQL permitirá realizar consultas a la base de datos que está ubicada en el Departamento de Informática del Centro de Investigaciones, desde cualquier lugar que se desee.

Esta página Web tendrá un enlace desde el sitio Web que mantiene actualmente el CIOH. Para ingresar a realizar consultas relacionadas a la base de datos diseñada en este proyecto se ingresará a la página principal del CIOH y solo podrán realizar consultas los usuarios autorizados para tal fin.

Cuando el usuario intente realizar alguna consulta a la base de datos encontrara una página como lo indica la figura 38 la cual permitirá controlar el acceso de los usuarios, esto se hace pidiéndole al usuario que ingrese un nombre de usuario y contraseña validos, con el fin de verificar si tiene permiso para consultar la base de datos. La página que permite lograr este fin es la siguiente:

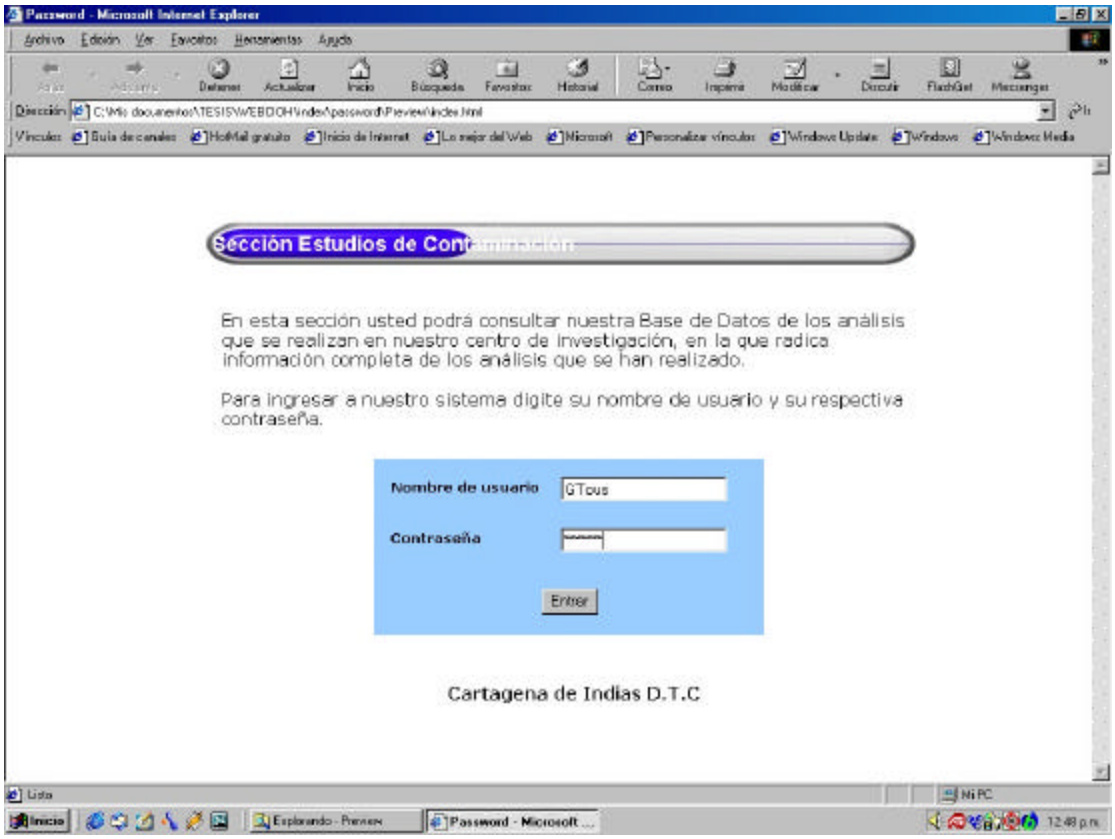

Figura 38 : Contraseña página Web PHP.

El siguiente código que se encuentra asociado al botón Entrar permite realizar el control de acceso de los usuarios a la base de datos.

```
<?
// conectar con el servidor de MySql
$Conecta = mysql_connect("172.200.24.33","rootlq","Laboratorio");

// seleccionamos la base de datos con la cual trabajaremos
$dbase = mysql_select_db("BDLABQUIM", $Conecta);
```

```
// tabla de usuarios
$query = mysql_query("select * from passwords where Login = '$NUsuario' and password =
'$Contraseña'");

if ($NUsuario==" ") {
  echo("<h2><br><br><div align=center>Debe ingresar un nombre de usuario
válido<br><br>");
}
else{
    if ($Contraseña==" "){
       echo("<div align=center><h2>Debe ingresar una Contraseña válida<br><br>");
     }
    else{
          if ($res = mysql_fetch_array($query)) {
             echo ("<div align=center><h1>Bienvenido $NUsuario</h1><hr></div>");
             echo ("<h2><div align=center>Consultas<br><br>");
             echo ("<div align=center><br><a href='consulta.php?user=$user'><br><br>");
          }
     else {
        echo ("<div align=center><br><h2> Nombre de usuario no válido. <br><br></h2>");
     }
   }
}
```

?>

Después de haber verificado la autenticidad de los datos ingresados por el usuario en la página de entrada, éste encontrara la página Web que le permitirá realizar las búsquedas. La figura número 39 permite realizar consultas por medio de dos parámetros que son: por fecha de análisis de muestras y por estación de muestreo.

Figura 39 : Página consultas PHP.

El siguiente código se encuentra asociado al botón buscar de la opción “búsqueda por fecha” , la cual permite realizar consultas de los análisis de la muestra, que se encuentran entre el rango de fechas ingresados por el usuario.

```
<?
//Se conecta con el Servidor MySql
$Conecta = mysql_connect("172.200.24.33","rootlq","Laboratorio");

//Selecciona la base de datos a utilizar
$dbase = mysql_select_db("DBLABQUIM", $Conecta);

if ($FechaI==" "){
    echo ("<h2><div align=center>Error. El campo fecha no debe estar
    vacio.</h2><hr></div>");
}
if ($FechaF==" "){
    echo ("<h2><div align=center>Error. El campo fecha no debe estar
    vacio.</h2><hr></div>");
}

else{
     if($query = mysql_query("select parametros.nombre from parametros,muestra where
     (muestra.fecharecep between(fechaI,FecahF))")){
```

```
      echo ("<div align=center><h1>Resultados de la consulta</h1><h2>Análisis que fueron
     realizados entre las fechas solicitadas: $fechaI,$fechaF</h2><hr><br>");
      echo ("<table border=1><tr><td><h3>Nombre</h3></td><td></tr>");
     while ($result = mysql_fetch_array($query)) {
          $fecha = $result["fecha"];


       }
       echo ("\n\n");
     }
  }
 else{
    echo ("<h1><div align=center>No hay análisis realizados entre estas fechas $fecha</h1>");
  }

 mysql_close($Conecta);
}
?>
```

El siguiente código asociado al botón buscar de la opción “búsqueda por estación” permite realizar consultas a la base de datos relacionadas a la estación de muestreo en particular donde fue tomada una muestra.

```
<?
//Se conecta con el Servidor MySql
```

```
$Conecta = mysql_connect("172.200.24.33","rootlq","Laboratorio");


//Selecciona la base de datos a utilizar
$dbase = mysql_select_db("DBLABQUIM", $Conecta);
if ($Estación==" "){
    echo ("<h2><div align=center>Error. El campo estación no debe estar
    vacio.</h2><hr></div>");
}
else{
     if($query = mysql_query("select nombre from parametros where
     (codestación.estación='estación')")){
     echo ("<div align=center><h1>Resultados de la consulta</h1><h2>Análisis que fueron
     realizados en la estación seleccionada: $fechaI,$fechaF</h2><hr><br>");
     echo ("<table border=1><tr><td><h3>Nombre</h3></td><td></tr>");
     while ($result = mysql_fetch_array($query)) {
        $estación = $result["estación"];
       }
       echo ("\n\n");
     }
  }
 else{
    echo ("<h1><div align=center>No hay análisis realizados en esta estación
$estación</h1>");
  }
```

```
  mysql_close($Conecta);
}
?>
```

6. RECOMENDACIONES

Cuando se diseña una aplicación Cliente/Servidor, se debe llevar a cabo el análisis habitual del sistema, así como un análisis adicional relacionado específicamente con las aplicaciones Cliente/Servidor. Es necesario plantearse donde se ubicaran los datos utilizados por las consultas, los formulario, los menús y los informes, y cómo se tendrá acceso a esta información. Se pueden plantear inquietudes tales como:

- ¿Qué tablas se almacenara en el servidor remoto una vez implantada la aplicación?
- ¿Qué tablas se almacenarían de forma más eficaz como tablas de búsquedas locales?
- ¿Qué reglas corporativas exige el servidor y como interactúa su aplicación con estas reglas?

Cuando se hayan determinado los componentes básicos de la aplicación Cliente/Servidor, se puede empezar a diseñar la forma en que la aplicación tendrá acceso a los datos y los actualizara; y prever todos los posibles inconvenientes que pueden presentarse en le desarrollo de la aplicación tanto con el lenguaje Cliente como con el motor de base de datos. Y especialmente con la interfaz y conexión de dichos componentes para evitar problemas futuros.

Un factor importante para generar una aplicación Cliente/Servidor rápida y eficiente es reducir al mínimo la cantidad de datos que necesita extraer del servidor. Para acelerar el rendimiento se utilizan técnicas de acceso a datos basadas en conjuntos para filtrar la cantidad de datos descargados.

Es importante saber en que sistema operativo debe ejecutarse el servidor por dos razones. La primera: da una medida de la flexibilidad que tenemos a la hora de seleccionar una configuración de la red. En segundo lugar, no todos los sistemas operativos se comportan igual de eficientes actuando como servidores de bases de datos.

El manejo que se da para la conexión entre un Cliente y un Servidor que trabajen bajo plataformas diferentes es mucho más complejo que los que trabajen bajo la misma plataforma por lo cual es necesario tomar todas las prevenciones del caso, para que se pueda realizar la conexión a cabalidad y no tener problemas futuros.

Cuando la aplicación esta plenamente implementada para datos remotos y ha completado la fase de prueba y depuración, se puede ajustar la velocidad y el rendimiento de toda la aplicación.

Se puede combinar la eficacia de las reglas de validación de datos y los procedimientos almacenados de Visual FoxPro con las reglas de validación de datos y los procedimientos almacenados del origen de datos con el fin de generar aplicaciones Cliente/Servidor que protejan la integridad de los datos.

Puede utilizar las transacciones de Visual FoxPro para prototipos locales y para procesos de datos locales. Utilice las transacciones del Servidor para actualizaciones, inserciones y eliminaciones de datos remotos.

CONCLUSIONES.

Al terminar la investigación resulta evidente la importancia que tienen las aplicaciones Cliente/Servidor. Se hizo énfasis en un lenguaje de programación como es Visual FoxPro, el cual presenta sus ventajas y desventajas.

Con la arquitectura Cliente/Servidor los usuarios no están limitados a un tipo de sistemas o plataforma. La gama de acciones que se realizan entre un Cliente y el Servidor es casi ilimitada. Es decir mientras se definan y realicen debidamente todos los parámetros necesarios para la conexión entre el Cliente y el Servidor, tales como la creación de DSN, la fuente de datos, los drivers etc. La aplicación desarrollada en determinado lenguaje Cliente debe correr en cualquier servidor sin importar la plataforma en que se encuentre, claro está que dependiendo del Cliente y de la plataforma del Servidor así mismo será la complejidad en la conexión.

Los componentes claves de una aplicación Cliente/Servidor (Cliente, red, Servidor) trabajan juntos (Interoperatividad), cualquiera de éstos puede ser reemplazado cuando se necesite, ya sea para crecimiento o para reducir el procesamiento, sin tener mayor impacto en los otros elementos (Escalabilidad) como nuevas tecnologías (multimedia, redes de banda ancha, bases de datos distribuidas) las cuales pueden ser incorporadas al sistema (adaptabilidad).

El servidor de base de datos mantiene la integridad de la información; la forma de acceder a los datos es tan directa y fácil como se desee, los datos pueden ser accesados a través de WAN's con múltiples aplicaciones Clientes (Accesibilidad). La distribución de tareas permite aumentar el desempeño con unos costos menores, el desempeño puede ser optimizado tanto por hardware como por procesos, la seguridad de los datos es centralizada en el Servidor.

Es necesario que tanto el usuario final como el desarrollador de una aplicación Cliente/Servidor tengan claro lo que desean implementar, ya que de esto depende el éxito de la misma. Con base en esto se deben buscar las mejores herramientas que permitan obtener una aplicación optima, rápida y eficiente; garantizando así una aplicación escalable y un máximo rendimiento en cuanto al volumen de tráfico se refiere, permitiendo así que todos los usuarios estén satisfechos con el acceso y manipulación de la información.

En una aplicación Cliente/Servidor robusta, gran parte del código debe ocuparse de los errores y mensajes devueltos desde el Servidor o el Sistema Operativo en el que se ejecuta la aplicación. La aplicación no solo debe manejar todos estos errores, sino además debe estar diseñada para hacerlo desde el principio.

El máximo rendimiento en una aplicación se obtiene cuando los datos y otros atributos de la base de datos se almacena en la plataforma óptima. La mejor plataforma para un elemento en concreto depende de la forma en que se tiene acceso y se actualizan dichos elementos.

En el estudio detallado de la arquitectura Cliente/Servidor se puede observar que dicha arquitectura a pesar de tener muchas alternativas de trabajo tiene pautas generales a seguir,

sin embargo presenta variaciones dependiendo de las características del Cliente y del Servidor que se utilice.

El modelo relacional permite despreocuparse completamente del complejo almacenamiento de los datos, pues los presenta en forma de sencillas tablas, en las cuales solo es necesario especificar las columnas, filas y nombre de tablas para obtener los datos deseados mediante instrucciones SQL. Es decir, el modelo relacional muestra una clara diferencia entre el aspecto físico de los datos y su representación lógica.

En las bases de datos relacionales, los datos se encuentran interrelacionados, es decir existen relaciones lógicas entre ellos. Existe independencia de los datos respecto de los procesos que los explotan, los datos se almacenan sin redundancias. Estas a su vez proporcionan mecanismos de seguridad e integridad.

Visual FoxPro es un lenguaje Cliente que permite la conexión con un determinado motor de base de datos de una manera sencilla presentando varias alternativas para la realización de dicha conexión. Además, tiene como ventaja que solo necesita conectarse e inmediatamente se accede al Servidor mediante una instrucción SQL, permitiendo así el acceso a los datos que se necesitan tomar del Servidor.

Son muchas las conclusiones a las cuales se llegó a través del desarrollo de esta investigación, pero en general se puede decir que en este momento en el mercado prima esta tecnología, ya que es una herramienta muy útil para el desarrollo de grandes y medianas empresas a nivel mundial, también es importante resaltar que aunque va en marcha con los

avances tecnológicos día a día aún faltan muchos campos por explorar referentes a esta arquitectura.

BIBLIOGRAFÍA.

CAIRO. Guardati. Estructura de datos. México. McGrawHill. 1997. Páginas 190-220

FREDMAN Alan. Diccionario de Computación. Madrid. MacGrawHill.1999. Páginas 85,110,150

KORTH Henry. Fundamentos de Bases de Datos. México. MacGrawHill. 1993. Páginas 50-120

KENDALL Kenneth, Julie Kendall. Análisis y Diseño de Sistemas de Información. España. Prentice-Hall. 1995. Páginas 100-130

IGLESIAS Ruben. Visual FoxPro 5. Fundamentos y técnicas de programación. Madrid España. Ra-ma editorial. 1997. Páginas 80-150

Microsoft Corporation. Visual FoxPro. Manual del programador. USA. Microsoft. 1996.Páginas 80,145-183

PRESSMAN. Roger. Ingenieria de software. Un enfoque práctico. 4ª edición. México. McGrawHill. 1997.

SENN James A. Análisis y Diseño de Sistemas de Información. México. McGrawHill. 1993. Páginas 124-159

Modelo Cliente / Servidor

http://www.inei.gob.pe/cpi-mapa/bancopub/libfree/lib616/cap0303.HTM

http://atenea.udistrital.edu.co/estudiantes/olmerol/aplicaciones.html

http://ar.geocities.com/r_niella/Document/t_cap1.htm

http://ar.geocities.com/r_niella/Document/t_cap2.htm

http://ar.geocities.com/r_niella/Document/t_cap3.htm

http://www.map.es/csi/silice/Global76.html

http://www.map.es/csi/silice/Global75.html

http://www.map.es/csi/silice/Global74.html

PHP

http://www.programacion.net/cursos/php/datosconsulta.htm

http://www.programacion.net/cursos/php/mysql.htm

Anexos

DESARROLLO E IMPLEMENTACIÓN DE UN SISTEMA DE INFORMACIÓN PARA EL LABORATORIO DE QUÍMICA Y MICROBIOLOGIA DEL CENTRO DE INVESTIGACIONES OCEANOGRÁFICAS E HIDROGRÁFICAS

MARIO IGNACIO DE LEON CÉSPEDES

JOSE LUIS PALLARES VARELA

UNIVERSIDAD TECNOLÓGICA DE BOLÍVAR INSTITUCIÓN UNIVERSITARIA

FACULTAD DE INGENIERÍA DE SISTEMAS

CARTAGENA

2002

Anexo A

Funciones del personal de la Sección de Estudios de Contaminación (Laboratorio de Química, Biología y Microbiología)

Anexo B

Formatos utilizados anteriormente para registros de toma, recepción de muestras y entrega de resultados.

Los siguientes formatos son una muestra o un bosquejo de los 30 formatos utilizados en los laboratorios de química, biología y microbiología, los formatos restantes tienen la misma estructura que los descritos a continuación variando únicamente las formulas o procedimientos para obtener los resultados de los análisis de la muestra.

Anexo C

Esquema de la Red Interna del Centro de Investigaciones Oceanográficas e Hidrográficas

ESQUEMA DE LA RED INTERNA DEL CENTRO DE INVESTIGACIONES OCEANOGRÀFICAS E HIDROGRÀFICAS (CIOH)

El Centro de Investigaciones cuenta en la actualidad con una serie de departamentos que están conectados entre sí, brindando una mayor eficiencia tanto en la comunicación de dependencias como la relación con el exterior.

Para llevar acabo todo eso se tiene una infraestructura de comunicación bien definida que le permite realizar sus labores adecuadamente y para ello dispone de los siguientes elementos:

Un servidor, encargado de brindar todos los servicios que en la institución se prestan y el cual tiene las siguientes características: Pentium II de 400 Mhz, disco duro de 10 GB y memoria RAM de 196 MB.

La plataforma computacional que maneja el Servidor es Linux Mandrake 8.3.

La conexión entre las terminales y el Servidor se hacen mediante la topología estrella.

Para la conexión entre las terminales y el Servidor dentro de las instalaciones del CIOH se utiliza cable UTP RJ45.

Además de todo esto cuenta con un Switch 10/100 marca 3com para hacer el switcheo entre todas los puntos conectados a la Red.

Para la comunicación con el exterior se tiene un HUB de 10 Mbps, conectado a un Transeiver que permite cambiar la señal de fibra óptica a UTP, este es marca Olicom.

Tiene un convenio con IMPSAT quien es la empresa encargada de proporcionarle el servicio de Internet con un ancho de banda de 768 Kbps permitiéndole mayor rapidez a la hora de realizar las transmisiones de datos.

Cuenta con una unidad zip de 250 MB, así como también con una UPS para mayor seguridad de la información, la cual es de 10 KVA y tiene la capacidad de soportar hasta 30 equipos conectados a la vez.

GLOSARIO

C.I.O.H: CENTRO DE INVESTIGACIÓN OCEANOGRAFICAS E HIDROGRAFICAS

SOFTWARE: Software es el conjunto de instrucciones que las computadoras emplean para manipular datos (Los Programas) Sin el software, la computadora sería un conjunto de medios sin utilizar. Al cargar los programas en una computadora, la máquina actuará como si recibieran a una educación instantánea; de pronto "sabe" cómo pensar y cómo operar.

Software es un conjunto de programas, documentos, procedimientos, y rutinas asociados con la operación de un sistema de computo. Distinguiéndose de los componentes físicos llamados hardware.

Comúnmente a los programas de computación se les llama software; el software asegura que el programa o sistema cumpla por completo con sus objetivos, opera con eficiencia, esta adecuadamente documentado, y es suficientemente sencillo de operar.

Es simplemente el conjunto de instrucciones individuales que se le proporciona al microprocesador para que pueda procesar los datos y generar los resultados esperados.

ORDENADOR: Es el dispositivo electrónico capaz de recibir un conjunto de instrucciones y ejecutarlas realizado cálculos sobre los datos numéricos, o bien compilando y correlacionado otros tipos de información.

SERVIDOR: Un servidor es un ordenador de gran potencia que se encarga de "prestar un servicio" a otros ordenadores (por ejemplo, el suyo) que se conectan a él. Varios servidores juntos, es decir conectados entre sí forman una red IRC.

Estas máquinas se encargan de hacer que los mensajes que usted escriba lleguen a su destino y también de hacer llegar hasta usted, los mensajes del resto de usuarios. En definitiva, se ocupan de gestionar todo el tráfico de información que se produce en la red IRC.

CLIENTE: Un cliente es un consumidor de los servicios del servidor; en otras palabras, un cliente utiliza espacio en disco, impresora o MODEM proporcionando por el servidor.

BASES DE DATOS: Una base de datos no es más que un conjunto de archivos con unas características propias que los hacen especiales. Entre estas características se puede destacar su facilidad para almacenar datos de diversos formatos en un mismo archivo y su ordenación si se desea.

PROCESAMIENTO: Sometimiento de algo a un proceso de elaboración, transformación, de tal manera que se obtengan al final los resultados esperados.

PHP: PHP es el acrónimo de Hipertext Preprocesor. Es un lenguaje de programación del lado del servidor gratuito e independiente de plataforma, rápido, con una gran librería de funciones y mucha documentación.