Escuela Politécnica del Ejército

Facultad de Ingeniería Electrónica

Proyecto de Grado para la Obtención del Título en Ingeniería Electrónica

Monitoreo, Control y Adquisición de Datos de Energía Eléctrica mediante Internet

Denyss Estévez Montalvo

Marzo - 2005

Certificación

Certificamos que el presente trabajo ha sido realizado completamente y enteramente por Denyss Estévez Montalvo, para constancia de lo cual firmamos a continuación

Ing. Alejandro Chacón	Ing. Rodolfo Gordillo
DIRECTOR	CODIRECTOR

Agradecimientos

Agradezco a mis compañeros y amigos de profesión y de trabajo por sus oportunas ayudas, a mis padres que me ayudado para dar el último paso para culminar mi carrera y a mi esposa por apoyarme en la consecusión de mi título profesional.

Dedicatoria

Este trabajo se lo dedico a mis padres, quienes a pesar de todos los contratiempos surgidos en el desarrollo del mismo han estado presentes para brindar su apoyo; pero principalmente a mi esposa quien ha estado presente en los momentos más duros, difíciles y extraviados de mi vida siempre tendiéndome una mano, dándome su apoyo y sus palabras de aliento para llegar a la culminación de mi carrera profesional y permitiéndome escalar un peldaño más en la formación diaria de la compleja existencia humana.

ÍNDICE GENERAL

Prólogo

Este proyecto busca realizar todo el proceso de configuración de un sistema de adquisición de datos, el procesamiento de los mismos así como su manipulación y almacenamiento de una forma más técnica, eficiente y eficaz.

También busca la economización de los costos de licenciamiento para las interfaces de cliente y, que las mismas puedan ser modificadas sin que ello represente la redistribución y reinstalación en cada computadora de cliente.

En el capítulo 1 se realiza un análisis exhaustivo de tecnología y técnicas empleadas en la solución implementada en la primera fase del proyecto global.

En el capítulo 2 se propone técnicas de solución al problema con tecnologías de punta y metodologías probadas en el ámbito comercial. Luego se realiza un análisis técnico – económico de las posibles vías de implementación y mejoramiento en la segunda fase del proyecto global. Aquí se escogerá una tecnología para cada tipo de actividad a desarrollarse en la nueva implementación del proyecto.

En el capítulo 3 se detalla la implementación del nuevo proyecto con las tecnologías seleccionadas en el capítulo anterior. Se describen las 3 formas de conexión actuales a los Sistemas de Bases de Datos y escoge la que mejor tecnología y rendimiento proporcione, realizando ejemplos de cómo se realizará la conexión a la base de datos construida para el sistema. A continuación se habla sobre las estructuras de almacenamiento y manipulación de datos, los tipos de lenguajes existentes y las instrucciones estándares que utilizan estos

lenguajes, todas en conformidad con un estándar internacional definido para los Sistemas de Bases de Datos.

Luego se detalla el proceso para modelamiento y creación de las estructuras de almacenamiento de los datos; después del respectivo análisis y abstracción de la información que se necesita de la planta industrial se aplicará la metodología antes descrita, mediante la cual se crearán las mismas para el sistema que se está desarrollando. Al finalizar este proceso se obtendrá el script de creación de todas las estructuras de almacenamiento.

A continuación se analiza y detalla el funcionamiento del sistema de transmisión de datos en tiempo real de Nacional Instruments – DataSocket, con su programa en el lado del servidor como en el lado del cliente, el mismo que permite que los datos de la adquisición sean transmitidos en tiempo real y sin importar cuales y cuantos sean los receptores mientras puedan utilizar el protocolo dstp (Data Socket Transport Protocol).

Seguidamente se realiza un análisis del flujo de información de la aplicación de servidor que será realizada en LabVIEW 6.0i con el acceso a Sistemas de Bases de Datos mediante MDAC (Microsoft Data Access Components) versión 2.8 utilizando el soporte de ActiverX de LabVIEW, ya que no posee soporte nativo para manejo de bases de datos. Se describe la forma en que debe configurarse la aplicación en la primera ejecución en el servidor, así como también el proceso detallado de la implementación de los circuitos virtuales para lectura y escritura del registro de Windows, circuitos virtuales para implementación de las funciones más utilizadas y estándares para la manipulación de datos, la configuración de la aplicación en sus cuatro secciones vitales: áreas, circuitos, puntos de medición y usuarios; se examinará también la forma de manipulación de los usuarios y permisos de visibilidad de la aplicación. Adicionalmente se verá algunas formas de monitoreo interno que posee la aplicación para realizar chequeos del correcto funcionamiento de las tarjetas de adquisición de datos, así también se revisará la implementación de un registro interno de las entradas y salidas que han realizado los usuarios al sistema durante el funcionamiento del mismo. A continuación se explica el funcionamiento de la aplicación en cuanto al proceso de adquisición, análisis, selección, almacenamiento y visualización de los datos obtenidos. Finalmente se indica el proceso de

desconexión del Sistema de Bases de Datos para evitar la pérdida de integridad, confidencialidad y disponibilidad de la información.

Inmediatamente después se realiza también un análisis del flujo de información en la aplicación de cliente que será realizada con una mezcla de varias tecnologías como LabVIEW 6.0i, Visual Basic 6.0, páginas web estáticas y dinámicas con controles ActiveX y cliente de DataSocket. Se verá que la aplicación estará dividida en tres partes: una informativa con respecto a los sistemas de adquisición de datos, otra con el monitoreo de los circuitos con sus formas de onda de voltaje y corriente en tiempo real y, finalmente la página que permite el despliegue de los datos históricos almacenados de los circuitos existentes en la planta monitoreada. Se examinará el código utilizado en cada una de las partes de la aplicación de cliente.

Finalmente se revisarán las instrucciones para la implantación del sistema en los equipos de servidor y clientes, para que el funcionamiento y rendimiento del mismo sea el esperado de acuerdo al diseño.

En el capítulo 4 se muestran las tablas de valores y el análisis de las mediciones realizadas en la fase de pruebas de rendimiento del sistema. Las pruebas del sistema se efectuaron con los Sistemas de Bases de Datos tanto en forma local como remota con velocidades de acceso de redes de área local (10 Mbps y 100 Mbps) y redes de área extendida (56 Kbps analógicos y 128 Kbps digitales) para poder establecer tiempos aproximados de respuesta del sistema en los distintos ambientes computacionales existentes.

Finalmente en el capítulo 5 se encuentran las conclusiones que se pueden apreciar después de realizadas todas las pruebas pertinentes de rendimiento y funcionalidad; así como también las recomendaciones que se ofrece al usuario final del sistema para que el mismo funcione de la manera más óptima y adecuada para que el usuario pueda tomar decisiones acertadas y a tiempo sobre la compra y el uso de su energía eléctrica.

CAPÍTULO 1

PROYECTO PRELIMINAR

1. PROYECTO PRELIMINAR

1.1. INTRODUCCIÓN

El proyecto preliminar que antecede al presente trabajo trata sobre la Implementación de un Sistema de Monitoreo de Energía Eléctrica en la Cervecería Andina S.A.

La visualización del monitoreo de parámetros fue realizada mediante el software LabVIEW en su versión 5.01. La aplicación desarrollada está instalada en 2 computadores en la parte de monitoreo de planta y en el cuarto de ingeniería, la comunicación de los datos está realizada sobre el protocolo estándar TCP/IP.

1.2. DESCRIPCIÓN DE LA PROBLEMÁTICA

El presente proyecto analizará la problemática que se encuentra en el proyecto preliminar desde el punto de vista de los procesos y almacenamiento de los datos, la comunicación entre la aplicación del servidor de adquisición de datos y la aplicación de los clientes, finalmente la situación del costo total de propiedad debido a las licencias de software que se tiene que adquirir para cada computador donde se desea ejecutar la aplicación de monitoreo desarrollada. A continuación se analizarán y detallarán cada uno de los puntos anteriormente mencionados.

Primero analizaremos el problema que existe con la manipulación, procesos y almacenamiento de los datos. Como primer paso se realiza la adquisición de los datos y se tiene los mismos como datos de tipo formas de onda que son matrices, el problema que tenemos aquí es que la aplicación está diseñada para un número fijo de canales, de manera que si se necesita hacer una ampliación de los puntos de medición hay que cambiar la aplicación. Para poder obtener la fecha completa con la información del milenio se utiliza una función de proceso que se ejecuta cada vez que se ejecuta un ciclo de monitoreo, restando tiempo que se podría utilizar para otras actividades de proceso. Luego se hace un análisis de las condiciones de tiempo para poder realizar un almacenamiento de datos cada 15 minutos, para lo cual se crean unas variables adicionales que son llamadas banderas, resultando en un uso más elevado de la memoria que crea una saturación de la misma y reducción del rendimiento del computador.

Como se realiza una medición de varios puntos al mismo tiempo, luego se necesita realizar una indexación de los datos obtenidos y multiplicar cada dato por el factor de atenuación que se aplicó para obtener la medición, esto implica de igual manera que si aumentan los puntos de medición o cambia el factor de atenuación por alguna causa fortuita se deben realizar cambios en la programación. Además este algoritmo de aplicar en forma fija las constantes de los factores de atenuación implica que existe mucho código repetido que se debe optimizar. Finalmente para formar los archivos de datos se lo realiza en una forma también repetitiva porque es la misma función para cada punto de carga en la planta; la información se almacena en archivos planos de texto donde no existe ningún tipo de compresión de datos y se crea un archivo de días que almacena información cada 15 minutos, un archivo de meses que almacena la información relevante de cada día pero aquí ya existe la primera redundancia de datos y luego un archivo de años que almacena la información relevante de cada mes donde nuevamente existe redundancia de datos. Todos estos archivos existen para cada punto de carga y se tienen alrededor de 26 puntos de carga en la planta, si el número de puntos de carga aumenta también aumentan el número de archivos volviéndose prácticamente inmanejable la situación. Finalmente debido a que son archivos planos de texto el acceso y lectura de estos se lo realiza en forma secuencial causando una pérdida de tiempo cuando se necesita acceder a los mismos para obtener información histórica para estadísticas, sin mencionar que se debe abrir cada uno de ellos por separado para poder realizar la búsqueda.

Como segundo punto analizaremos el problema de la comunicación que existe entre la aplicación del servidor de adquisición de datos y la aplicación de los clientes. En el proyecto preliminar la comunicación entre el servidor y la estación cliente está realizada con el conjunto de protocolos TCP/IP estándar lo cual garantiza que pueda comunicarse con cualquier sistema operativo, sin embargo como se utiliza el protocolo TCP que está orientado a conexión tenemos el problema de que primero se debe establecer la conexión para poder transmitir los datos. Para establecer la conexión se debe saber el nombre o dirección IP del computador con el que se desea conectar y además se debe saber el número de puerto TCP que se va a utilizar de acuerdo a la aplicación que se use, cuando se usa un puerto TCP se debe tener cuidado de no utilizar un puerto que esté utilizado por otra aplicación pues causaría conflictos con la otra aplicación y no se lograría la comunicación deseada.

La aplicación actual del servidor está escaneando la red cada cierto tiempo para averiguar si la aplicación cliente está a la espera de una comunicación mediante TCP. Este método primero empieza a saturar la red de comunicaciones con las constantes y periódicas averiguaciones y, además la función que establece la comunicación está en la aplicación del servidor por lo cual se necesitaría aumentar todas las funciones de comunicación y transferencia de datos para cada cliente que se encuentre en la red y desee conectarse al servidor para realizar el monitoreo. Esto generaría una duplicación de código y una pérdida de desempeño de la aplicación.

Como último punto hablaremos del costo total de propiedad que se tiene por la adquisición de licencias de uso del software de desarrollo que se debe realizar. El paquete de desarrollo en la versión full tiene un costo de USD. 2195 para cualquier plataforma, para la plataforma Windows existe una versión básica que tiene un costo de USD. 1095; de acuerdo a las funciones utilizadas dependerá la versión del paquete de desarrollo que se debe adquirir.

Suponiendo que se esté utilizando la versión básica del software tenemos un costo total de USD. 2190 por los 2 computadores. El problema se presenta el momento en que se desea aumentar más computadores que realizarán el monitoreo, pues se debe aumentar una licencia de USD. 1095 para cada computador. Sabemos que existe la versión professional

del software que permite realizar paquetes ejecutables que pueden funcionar solos, sin embargo el precio de esta versión es de USD. 3845 y existe el problema de que los ejecutables creados sirven sólo para la plataforma seleccionada, en este caso Windows. Por lo tanto, si se desea instalar la aplicación sobre otra plataforma como Linux debido a que es de licencia libre y más estable y segura que Windows, se necesitaría primero disponer de la versión full debido a que no existe versión básica para Linux con el costo antes mencionado dando un total de USD. 4390 para los 2 computadores. Otro problema que se presentaría es la necesidad de instalar la aplicación en cada computador que se una a la red para realizar el monitoreo, creando la pérdida de tiempo que esto significa. Finalmente puede darse el caso de que se realicen actualizaciones sobre la aplicación original para aumentar puntos de carga, aumentar funcionalidades a la aplicación, optimizar alguna parte del algoritmo aumentando o retirando procesos innecesarios entre otras cosas lo cual implicaría una recompilación de la aplicación y su debida redistribución; la misma se puede realizar a través de la red de comunicación pero debemos tomar en cuenta que las estaciones de trabajo pueden estar con claves de usuario restringido que no permitirían la descarga de la aplicación actualizada y la instalación de la misma en el sistema operativo.

Una vez analizados detalladamente los problemas que presenta la aplicación que se encuentra actualmente en funcionamiento, en el capítulo 2 procederemos a plantear las posibles soluciones y escogiendo la más adecuada realizando el debido análisis de costos vs. beneficios de cada una de las mismas. En resumen, lo que se plantea con este trabajo es mejorar el rendimiento y desempeño de la aplicación actual, así como también reducir los costos de una futura expansión del sistema.

CAPÍTULO 2

SOLUCIÓN PROPUESTA

2. SOLUCIÓN PROPUESTA

2.1.DESCRIPCIÓN GENERAL

Para solucionar los problemas que muestra el proyecto preliminar en las áreas antes mencionadas en el capítulo 1, se utilizarán técnicas y tecnologías conocidas e implementadas exitosamente en el área comercial.

Entre las técnicas y tecnologías mencionaremos primero a las bases de datos, las cuales nos permiten realizar una administración rápida, ordenada y segura de la información que genera la empresa, en este caso los datos del monitoreo de la energía eléctrica. Con el uso de las bases de datos solucionaremos el problema de la creación y mantenimiento de innumerables archivos planos de texto en los que se almacenan los datos, así como también la redundancia de información; además podemos considerar que el 90% de las bases de datos empresariales son multiplataforma, lo cual nos permite crear una aplicación de monitoreo que sea independiente del sistema de administración de datos y del sistema operativo donde se encuentra funcionando el mismo, así como también la posibilidad de poder compartir la información de forma transparente en el entorno de la red empresarial.

Como segunda tecnología a utilizarse mencionaremos al internet, más conocido como Web, en la forma de implementación de interfaces y aplicaciones. El uso de esta tecnología nos permite solucionar en forma conjunta los dos últimos problemas que muestra el

proyecto preliminar. Primero indicaremos que al tener la aplicación en un entorno Web, la Intranet empresarial, se solucionará el problema que se presenta en la comunicación ya que se usa una combinación de técnicas orientadas y no orientadas a la conexión para la transmisión de datos entre la estación servidor y las estaciones clientes que pueden visualizar los mismos en forma transparente, pues ahora se utilizan protocolos más avanzados y fáciles de manejar que funcionan sobre el estándar TCP/IP para garantizar una interoperabilidad entre las distintas plataformas de sistemas operativos.

Mediante esta misma tecnología se realizará la aplicación para los clientes; de esta manera logramos que la misma sea multiplataforma, de fácil y reducido costo de mantenimiento e implantación, no requiere licencias de uso, lo único que se necesita es un explorador de internet que se puede descargar en forma gratuita del Internet. Con el uso de esta técnica solucionamos finalmente el problema del costo total de propiedad, pues la única licencia que se necesita es la del servidor de adquisición de datos; además nos permite una gran interoperabilidad entre distintos sistemas operativos con mucha facilidad y transparencia.

2.2.ANÁLISIS TÉCNICO – ECONÓMICO DE HERRAMIENTAS

Para el análisis técnico – económico de herramientas se tomará en cuenta la plataforma de sistema operativo sobre la que está implementada la aplicación del proyecto preliminar, para que los cambios a realizar sean mínimos y el impacto causado sea el mínimo posible sobre los usuarios.

Se realizará un completo análisis de características implementadas, rendimiento y costo de los sistemas de administración de bases de datos disponibles en el mercado. Para la construcción de la aplicación cliente se tomarán en cuenta algunas alternativas factibles considerando su facilidad de implementación, facilidad de mantenimiento, rendimiento, interoperabilidad entre plataformas y costos totales de licenciamiento.

Sistema de Administración de Base de Datos

Entre los sistemas de bases de datos más conocidos existentes para la plataforma Windows que poseen licencia comercial tenemos:

- Microsoft SQL Server 2000
- Sybase Adaptive Enterprise Server 12.5
- Oracle Database 9i
- IBM DB2 7.1
- IBM Informix 12
- Ingres 2.6

Entre las bases de datos existentes para la plataforma Windows con licencia pública general tenemos:

- MySQL 4.1
- PostgreSQL 7.3.4

No se tomarán en cuenta las bases de datos personales o de escritorio debido a que su funcionamiento es el de simples manejadores de archivos porque no están diseñadas para un trabajo pesado ni tienen un completo soporte para comunicaciones en redes; entre estos podemos mencionar a los más conocidos que son:

- Microsoft Access
- Microsoft Foxpro
- Corel Paradox
- Lotus Approach
- dBase
- Clipper
- Filemaker Pro

Se realizarán los respectivos análisis a las bases de datos empresariales antes mencionadas sobre parámetros importantes y decisivos como:

- Estabilidad
- Escalabilidad
- Disponibilidad
- Rendimiento
- Compatibilidad
- Portabilidad
- Facilidad de Instalación
- Facilidad de Uso
- Interacción con otras Aplicaciones
- Comunicación en Redes
- Conformidad con el Estándar ANSI/SQL 92
- Plataformas Operativas Soportadas
- Precio

De los primeros análisis realizados se pueden obtener 2 tipos de resultados basándose en un parámetro importante y limitante a la vez que es la variedad de plataformas de sistema operativo soportadas en las que puede funcionar el motor de base de datos. La calificación que se realizará utilizará puntos cualitativos de 1 a 5 de la siguiente forma:

- Excelente (5 puntos)
- Bueno (4 puntos)
- Medio (3 puntos)
- Bajo (2 puntos)
- Deficiente (1 punto)

Luego de un completo análisis realizado sobre cada uno de los motores de bases de datos se presentan los resultados en la tabla 2.1 con los puntajes obtenidos. Para el parámetro de precio no se pueden aplicar los valores antes definidos, por lo tanto se utilizarán los 3 siguientes:

- Bajo
- Razonable
- Elevado

Luego de analizar los puntajes obtenidos se escogerán 3 opciones tomando en cuenta el parámetro de plataformas de sistema operativo soportadas antes referido y el precio del motor de base de datos debido a que éste un factor también muy importante que influye sobre la decisión de inversión en adquirir un sistema informático de procesamiento automático de datos.

Base de . Datos / Caraterística	Microsoft SQL Server 2000	Sybase Adaptive Server Enterprise 12.5	Oracle Database 9i	IBM DB2 7.1	IBM Informix 12	Ingres 2.6	MySQL 4.1	Postgre SQL 7.3.4
Estabilidad	Excelente	Buena	Buena	Excelente	Excelente	Buena	Buena	Buena
Escalabilidad	Excelente	Excelente	Excelente	Excelente	Excelente	Excelente	Buena	Buena
Disponibilidad	Buena	Excelente	Excelente	Excelente	Buena	Excelente	Excelente	Excelente
Rendimiento	Buena	Buena	Excelente	Excelente	Excelente	Buena	Excelente	Buena
Compatibilidad	Excelente	Buena	Excelente	Excelente	Excelente	Buena	Excelente	Buena
Portabilidad	Bajo	Buena	Excelente	Excelente	Excelente	Buena	Excelente	Buena
Facilidad de Instalación	Excelente	Buena	Buena	Buena	Buena	Buena	Excelente	Buena
Facilidad de Uso	Excelente	Buena	Buena	Buena	Buena	Buena	Excelente	Buena
Interacción con otras Aplicaciones	Excelente	Buena	Excelente	Excelente	Excelente	Excelente	Excelente	Buena
Comunicación en Redes	Excelente	Buena	Excelente	Excelente	Excelente	Excelente	Excelente	Excelente
Conformidad con el Estándar ANSI/SQL 92	Excelente	Buena	Excelente	Excelente	Buena	Excelente	Excelente	Excelente

Tabla 2.1 – Puntajes obtenidos por cada Base de Datos

Las opciones escogidas por sus características y precio son las siguientes:

- Microsoft SQL Server 2000, por su completa compatibilidad con Windows
- Oracle Database 9i, por su excelente rendimiento y capacidad multiplataforma
- MySQL 4.1, por su licencia pública general y capacidad multiplataforma

Sistema de Desarrollo de Aplicaciones

La aplicación del proyecto preliminar se encuentra desarrollada sobre la versión 5.01 de LabVIEW, partiendo de esta premisa presentaremos 2 alternativas de solución de la problemática y que serán detallados a continuación.

Primer Método.- consiste en la actualización a la versión 7 Express de LabVIEW, debido a que ésta ya nos permite interactuar directamente en el Web, con un costo de USD. 3.845 para la versión profesional. Además se requiere el paquete de SQL Toolkit para acceso directo desde LabVIEW a bases de datos mediante ODBC, con un costo de USD. 545.

Mediante estas herramientas podemos realizar la programación completa dentro del entorno de LabVIEW, sin embargo se debe tomar en cuenta que para el funcionamiento de los paneles remotos de LabVIEW se necesita tener activado el servidor Web integrado de LabVIEW, el cual no es muy seguro ni dispone de muchas opciones configurables como los servidores Web de los sistemas operativos.

Para otorgar un mayor nivel de funcionalidad y seguridad a la aplicación se debe acompañarlo del Internet Toolkit que tiene un costo de USD. 545, dándonos un monto total aproximado de USD. 4.900 para esta alternativa de solución.

Segundo Método.- consiste en la utilización de DataSocket para el monitoreo en tiempo real. DataSocket es una nueva tecnología de programación basada en el estándar industrial TCP/IP, que simplifica el intercambio de datos en tiempo real entre diferentes aplicaciones en una computadora o entre varias computadoras conectadas en una red. Se lo puede descargar en forma gratuita del sitio Web de National Instruments.

Para la creación de las páginas Web de la aplicación cliente se lo puede realizar mediante Visual Basic con la edición de creación de controles que tiene un costo aproximado de USD. 100 pues viene incluida con cursos de aprendizaje de Visual Basic que se puede conseguir con facilidad en el mercado. Luego, para poder realizar una interfase amigable con el usuario se puede hacer uso de cualquier programa de edición de páginas HTML que se puede descargar de Internet con un costo de hasta USD. 50.

Finalmente, para la manipulación de información en la base de datos se utilizarán las capacidades de LabVIEW para utilizar controles y objetos ActiveX, mediante esta comunicación nos conectaremos con los objetos ADO (ActiveX Data Objects) de Microsoft. Estos objetos y controles están incluidos en el paquete MDAC (Microsoft Data

Access Components) que se lo puede descargar en forma gratuita del sitio Web de Microsoft.

Sumando todos los costos tenemos un valor de USD. 150, lo cual es un valor muy reducido en comparación con la primera alternativa. Sin embargo, debemos considerar el costo de un servidor Web que sea capaz de procesar páginas de contendido dinámico para el acceso a la base de datos desde el explorador de internet; podemos utilizar la primera opción que sería el servidor Web integrado con los sistemas Windows NT versión 4.0 en adelante o como segunda opción tenemos el servidor Apache entre las mejores opciones que poseen gran estabilidad y no tienen costo alguno, teniendo así un precio totalmente bajo y accesible.

Entre las múltiples ventajas que presentan el uso de bases de datos y la implementación de interfaces Web mencionaremos a continuación las más importantes:

- Independencia de los clientes del software de desarrollo, aún poseyendo la versión básica de LabVIEW que no tiene la capacidad de generar aplicaciones independientes.
- Ahorro en costos de adquisición de licencias de uso debido a que un explorador de internet es suficiente y tiene una licencia comercial gratuita
- Independencia de la versión y más aún de la plataforma de sistema operativo, debido a que el explorador de internet permite una interacción y comunicación transparente para los usuarios dándoles muchas combinaciones entre servidor y cliente como:
  - Servidor: Windows; clientes: Windows, Macintosh, Linux y sus variantes
  - Servidor: Macintosh; clientes: Windows, Macintosh, Linux y sus variantes
  - Servidor: Linux, Solaris; clientes: Windows, Macintosh, Linux y sus variantes
- Posibilidad de comunicación y monitoreo en forma remota virtualmente desde cualquier parte del mundo (el servidor debe estar configurado y conectado al Internet en forma pública) sin necesidad de realizar instalaciones adicionales de software o configuraciones largas y complicadas para los clientes

Con respecto al uso de las bases de datos podemos mencionar las siguientes:

Ø Gestión más rápida y eficiente de la información debido a que la función primordial de los motores de bases de datos es la manipulación organizada de los datos, permitiendo de esta manera que el trabajo con los mismos sea más fácil y consistente.

Ø Las consultas, adiciones, modificaciones y eliminaciones de datos utilizando el estándar ANSI SQL serán independientes del motor de base de datos que se utilice, creando de esta manera una total flexibilidad y transparencia para la aplicación y el usuario y, permitiendo que se implemente con el motor de base de datos favorito por motivos de costos o de prestaciones ofrecidas por el mismo.

Ø Al utilizar un motor de base de datos con capacidades de comunicaciones en red se pueden crear aplicaciones más complejas y masivas en datos utilizando un esquema distribuido. Este permite que el servidor de base de datos se dedique a esa única tarea dejando que sus clientes enfoquen su tiempo de proceso en la adquisición, análisis y/o visualización de los datos.

Ø Los motores de base de datos también poseen sus propios esquemas de seguridad y control que dan a la aplicación un valor agregado excelente, manejando el acceso recurrente de los clientes evitando inconsistencias de información y a la vez permitiendo que existan diversos conjuntos de datos de aplicaciones diferentes al mismo tiempo, por ejemplo: Sistema de Contabilidad, Sistema de Monitoreo y Adquisición de Datos, Sistema de Ventas, Sistema de Atención al Cliente, etc. siempre y cuando ninguno de estos sea de misión crítica.

2.3.DISEÑO DEL SISTEMA

De acuerdo al proyecto preliminar sabemos que se toman los datos de voltaje, corriente y factor de potencia RMS instantáneos que sirven para crear una matriz y poder obtener el valor medio. Una vez obtenidos estos parámetros se procede a realizar los cálculos de las potencias y energías para luego poder almacenar los mismos.

Considerando que Windows no es un sistema operativo de tiempo real y LabVIEW no es un software de aplicación crítica no se podría dejar que LabVIEW realice todo el trabajo. Debido a que LabVIEW y Windows manejan de una forma diferente las prioridades de ejecución, no es conveniente acelerar mediante Windows las aplicaciones en

primer plano porque al estar LabVIEW como aplicación de este tipo, se crean conflictos entre los dos programas y la ejecución en realidad se vuelve más lenta.

Debemos considerar además que la fortaleza de LabVIEW es realizar adquisición y análisis de datos y la fortaleza de una base de datos es la manipulación de los mismos. Por esta razón es conveniente dejar la manipulación de los datos al software que posee esta fortaleza, sin embargo, es conveniente realizar pruebas de tiempos de ejecución entre LabVIEW y las bases de datos tentativas escogidas para el proyecto para poder tener parámetros de comparación y colocar el código de manipulación de datos donde se ejecute en la forma más óptima posible.

Una vez analizado todo el proceso que se está realizando en el sistema actual, podemos dividirlo en las siguientes etapas:

- Adquisición de los datos de voltaje, corriente y factor de potencia instantáneos en cada uno de los puntos de medición
- Clasificación y agrupación de los datos para su posterior empleo en los cálculos de potencia y energía dependiendo del tipo de circuito al que pertenece
- Verificación de las condiciones de tiempo para la manipulación de las distintas banderas que controlan la formación de los distintos archivos de datos
- Clasificación y agrupación de la información para el envío a los distintos archivos de datos dependiendo del conjunto de puntos monitoreados de la planta

Debido a que la aplicación está destinada al monitoreo se puede optar por realizar los cálculos de potencia en la base datos cada cierto tiempo o al insertar los registros y de ésta manera liberar del proceso adicional a la aplicación de monitoreo.

Adicionalmente podemos mencionar que al utilizar un motor de base de datos podemos obtener la fecha y la hora para la medición realizada del mismo motor de una manera más fiable; también es muy importante indicar que como la información se encontrará en un solo lugar es necesario agregar algunos códigos para poder identificar de que punto provienen los datos y a que tipo de circuito pertenece para poder realizar los cálculos con las fórmulas apropiadas. Aunque puede parecer que en vez de reducir la cantidad de información a almacenar estamos aumentando la misma, esto no es así debido

a que utilizamos un solo archivo y no varios al mismo tiempo y además los motores de bases de datos utilizan estos códigos para poder acelerar sus búsquedas con algoritmos especiales que los mismos poseen en su programación interna.

Aprovechando de las características especiales que poseen los motores de bases de datos para el manejo de información podemos aumentar algunos datos que serán muy útiles y cruciales en el momento de buscar la información que necesitamos para obtener estadísticas y tomar las decisiones técnicas y gerenciales, como por ejemplo: el momento en que suceden picos o caídas de voltaje o corriente y la frecuencia con estos fenómenos suceden en diversos puntos de la planta.

Finalmente, después de haber analizado todos los posibles datos y haber escogido los que puedan aportarnos con información objetiva y precisa, enumeraremos los que se utilizarán y almacenarán en la base de datos:

- Fecha
- Hora
- Número del Punto
- Área del Punto
- Tipo de Circuito
- Tipo de Medición
- Factor de Atenuación
- Frecuencia
- Alarma (valores fuera de rango)
- Voltaje RMS Mínimo
- Corriente RMS Máxima
- Factor de Potencia Mínimo
- Potencia Máxima
- Potencia Reactiva Máxima
- Potencia Aparente Máxima
- Energía
- Energía Reactiva

Para organizar la información de la manera más adecuada evitando la redundancia así como la falta de datos se realizará un diseño lógico y físico de las tablas que almacenarán la información, para que de esta manera también guarden relación entre sí. Los diseños se realizarán utilizando el estándar de los modelos Entidad-Relación de Codd. En la actualidad existen herramientas llamadas CASE que nos ayudan a realizar todo este proceso y que nos permiten generan una buena porción de código o interactuar directamente con las bases de datos, para que de esta manera los modelos sean consistentes y sea más eficiente el diseño de una aplicación.

En las figuras 2.1 y 2.2 se mostrará el modelo lógico y físico de datos respectivamente. En el capítulo 3 tendremos el script de creación de la base de datos en ANSI SQL 92 para que sea aplicable en cualquier motor de base de datos y finalmente el diccionario de datos que nos dará una explicación más concisa y detallada de lo que almacenará cada campo en cada tabla y el tipo de dato que posee cada uno.

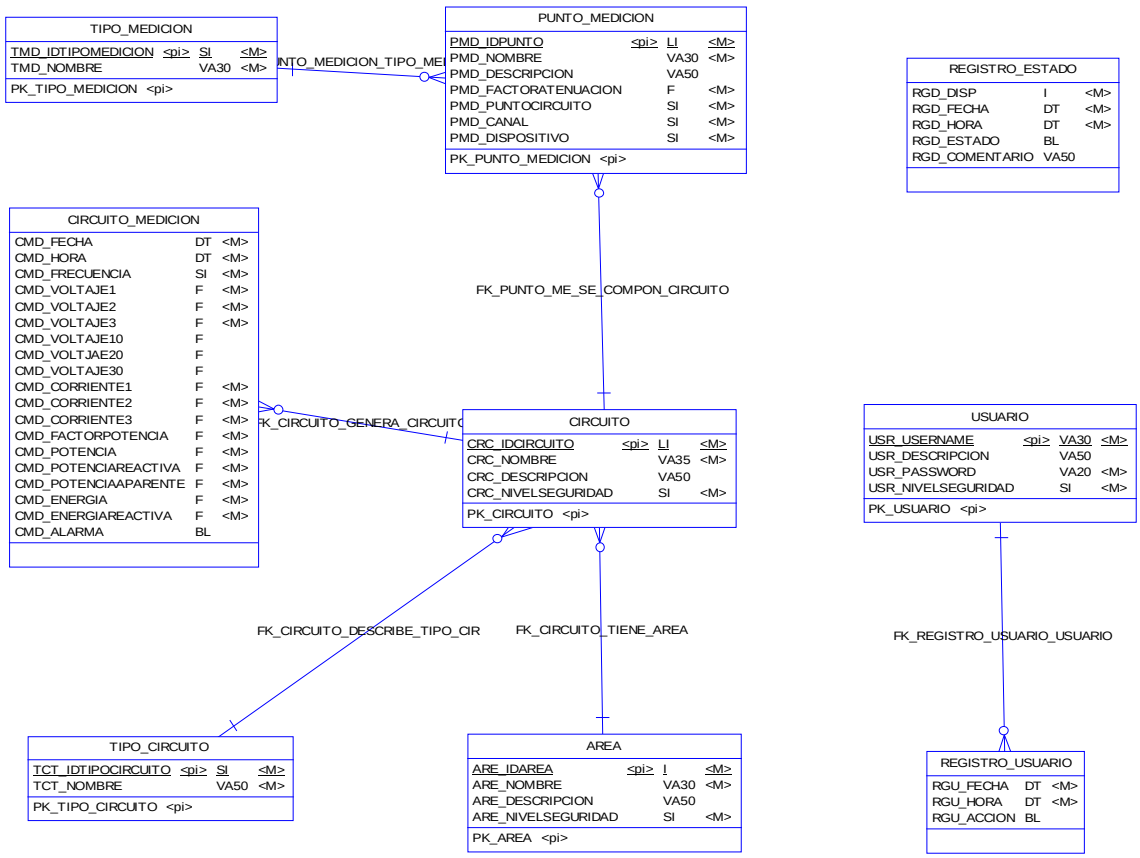

Figura 2.1 – Modelo Lógico de Datos

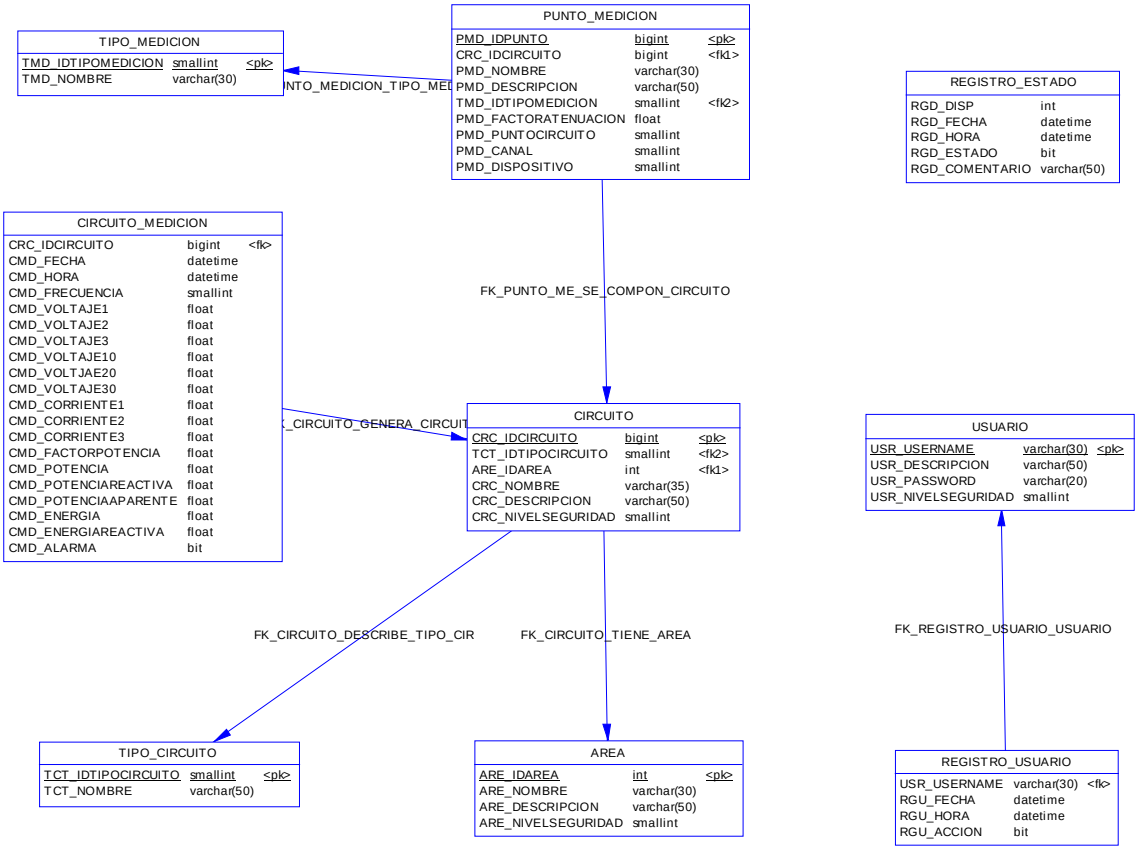

Figura 2.2 – Modelo Físico de Datos

La aplicación constará prácticamente de 3 partes o mejor dicho capas debido a su forma de creación e implementación y que las describiremos a continuación. Según el nuevo estándar de desarrollo para aplicaciones que manipulan datos, tenemos al modelo distribuido de N capas, pudiendo ser éstas 3 o más.

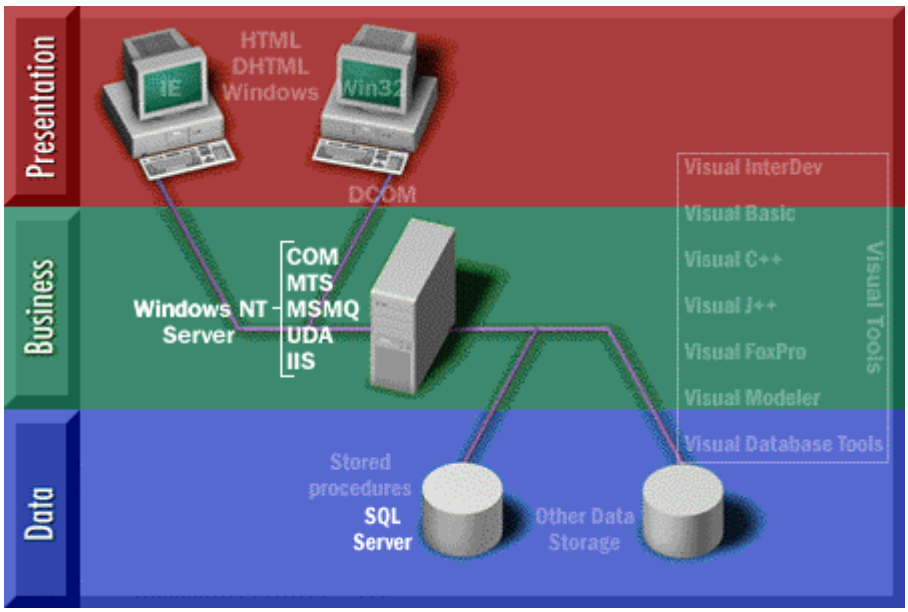

Figura 2.3 – Modelo de N Capas para Aplicaciones Distribuidas

Capa de Datos.- en esta capa se encuentra el motor de la base de datos que puede estar instalado en 1 o más servidores, es decir que la base de datos también puede ser distribuida siendo del mismo sistema y, la comunicación se la realiza mediante enlaces de red. Dependiendo de la base de datos esta sincronización y réplica la realiza el sistema operativo o el sistema de administración de la base de datos.

Para nuestra aplicación aquí se encuentra instalado el motor SQL Server 2000, Oracle Database 9i o MySQL 4.1; ésta capa puede estar instalada sobre cualquier sistema operativo en el que se soporte la base de datos elegida y aquí se realizará toda la administración y control de la información generada por la aplicación.

Mediante la utilización de la base de datos la aplicación se vuelve más flexible y personalizable permitiendo agregar algunos datos y crear más puntos de medición si la aplicación así lo requiere, siempre y cuando existan los puntos físicos de conexión disponibles en la tarjeta de adquisición de datos.

Capas de Aplicación.- en estas capas se encuentra toda la infraestructura de la aplicación que se está realizando y pueden estar instaladas en 1 o más servidores, es decir que existen 1 o más capas de software enlazadas entre sí en forma física por medio de enlaces de red y en forma lógica por medio de memoria compartida, archivos compartidos,

llamadas a procedimientos externos y; cada una puede estar desarrollada en un lenguaje diferente.

Para nuestra aplicación, en esta capa tendremos una mezcla entre LabVIEW y otros programas; debido a que LabVIEW se encargará de realizar la adquisición, análisis y almacenamiento de datos, siendo considerada la aplicación servidora de datos. Adicionalmente aquí vamos a tener el servidor Web que puede ser Internet Information Server de Microsoft, Apache o el servidor Web G integrado de LabVIEW porque éste será el servidor de aplicaciones que procesará las páginas Web que manipulan datos o imágenes y, entre las más conocidas tenemos:

- Common Gateway Interfase (*.cgi)
- Active Server Pages (*.asp)
- Java Server Pages (*.jsp)
- Dynamic Hyper Text Markup Language (*.dhtml)
- Phyton Pages (*.php)
- Hyper Text Markup Language con parámetros (*.html?parm1?parm2)

Este servidor es el segundo más importante debido a que las páginas Web tanto estáticas como dinámicas no pueden funcionar por sí solas y, la totalidad de la interfase del cliente está basada en documentos Web.

Estas capas son las que contienen lo que se conoce como las reglas del negocio, es decir que aquí se realizan todos los cálculos, procesos, recepción de datos validados y envío al servidor de base de datos, recepción de peticiones del cliente para luego tener una recuperación de datos del servidor y finalmente el envío de los mismos a los clientes.

Capa de Cliente.- en esta capa se encuentran todas las interfases de cliente que pueden funcionar sobre un sistema operativo específico y estar desarrolladas en lenguajes como Visual Basic, Visual C++, Visual J++, Delphi, Power Builder, Visual Age, etc. o también pueden ser páginas Web estáticas o dinámicas desarrolladas en Visual Basic Script, Java Script, Java, Phyton, Perl, CGI, etc.

La gran ventaja de las interfases desarrolladas en páginas Web es que son multiplataforma debido a que el servidor Web es el que se encarga de procesar las aplicaciones, sin importar el sistema operativo que está usando el cliente.

Estas interfases no realizan ningún cálculo ni proceso sólo se ocupan de la validación de los datos ingresados, agruparlos y enviarlos al servidor de aplicación; por esta razón son muy ligeras para visualizarse e ideales para aplicaciones en redes de uso masivo como es el internet.

El único software que se necesita disponer para poder utilizar estas aplicaciones es un explorador de internet, que se lo puede descargar en forma gratuita desde el mismo internet o que lo entregan los proveedores de servicios de internet (ISP); entre los exploradores más conocidos tenemos:

- Microsoft Internet Explorer (Windows, Macintosh), descargable desde http://www.microsoft.com
- Netscape Navigator (Windows, Macintosh, Linux), descargable desde http://www.netscape.com
- Mozilla (Windows, Macintosh, Linux) , descargable desde http://www.mozilla.org
- Opera (Windows, Macintosh, Linux) , descargable desde http://www.opera.com

Debido a que el internet es una red de uso masivo se ha vuelto demasiado insegura por el aumento descontrolado de hackers y crackers, por lo tanto se ha hecho necesario que además de un nombre de usuario y contraseña se tenga que implementar mecanismos adicionales de seguridad como:

- Servidores seguros de protocolo de hipertexto
- Encriptación de datos
- Protocolos de cifrado
- Servidores de certificados de identidad
- Combinaciones de clave pública y clave privada
- Protocolos de entunelamiento

Para nuestra aplicación todo lo realizaremos en una sola máquina; es decir que tendremos el servidor de base de datos, el servidor de aplicaciones y las interfases de cliente en el mismo lugar pero manteniendo la estructura de diseño de 3 capas. Debemos aclarar que la aplicación quedará construida y lista para que funcione en 2 o más máquinas que tengan instalado un explorador de internet y una tarjeta de red para comunicaciones, que pueden ser cableadas o inalámbricas.

CAPÍTULO 3

CONSTRUCCIÓN DEL SISTEMA

3. CONSTRUCCIÓN DEL SISTEMA

Para la construcción de la aplicación se necesita poseer buenos conocimientos sobre la manipulación de los datos y sus estructuras mediante instrucciones del Lenguaje Estructurado de Consulta (SQL), las diversas formas de conectarse a estas fuentes de datos para conversar con las mismas. Así como también se necesita tener un buen dominio del protocolo TCP/IP y sus aplicaciones con las respectivas limitaciones.

3.1. FORMAS DE CONEXIÓN A BASES DE DATOS

Actualmente existen tres formas de conexión a una base de datos y son las siguientes:

- ODBC (Open DataBase Conectivity)
- JDBC (Java DataBase Conectivity)
- OLEDB (OLE DataBase)

Analizaremos con detenimiento los métodos de ODBC y OLEDB, el método JDBC no será analizado debido a que no es nativo de los Sistemas Operativos Windows sino basado en Java y además porque LabVIEW no soporta Java.

3.1.1. ODBC.- significa la conectividad abierta de bases de datos. Es una especificación para la API de bases de datos. Esta API es independiente de cualquier DBMS o sistema operativo y

además es independiente del lenguaje. La API de ODBC está basada en las especificaciones CLI de X/Open e ISO/TEC. El ODBC 3.x implementa completamente estas dos especificaciones y adiciona características comúnmente necesitadas por los desarrolladores de aplicaciones de bases de datos basadas en pantallas como los cursores enrollables.

Las funciones en la API ODBC son implementadas por los desarrolladores de los controladores específicos de una DBMS. Las aplicaciones llaman a las funciones en estos controladores para acceder a los datos en una forma independiente del DBMS. Un administrador de controladores maneja la comunicación entre las aplicaciones y los controladores.

A pesar de que Microsoft provee un administrador de controladores para computadores que corren sistemas Windows de 16, 32 y 64 bits, ha escrito varios controladores de ODBC y llama a las funciones de ODBC desde algunas de sus aplicaciones, cualquiera puede escribir aplicaciones y controladores ODBC.

Para ayudar a los desarrolladores de aplicaciones y controladores, Microsoft ofrece un Kit de desarrollo de software para computadores que corren sistemas Windows de 16, 32 y 64 bits; que provee el Administrador de controladores, la DLL de instalación, herramientas de prueba y aplicaciones de ejemplo.

Es importante entender que ODBC está diseñado para exponer las capacidades de la base de datos, no para suplementarla. Por lo tanto, los escritores de aplicaciones no deben esperar que usando ODBC se transformará súbitamente una base de datos simple en un motor de bases de datos relacional con el máximo de características. Ni tampoco esperen los escritores de controladores la implementación de la funcionalidad no encontrada en la base de datos subyacente. Una excepción a esto es que los desarrolladores que escriben controladores que acceden directamente a los archivos de datos (como los datos en un archivo de Xbase) son requeridos para escribir un motor de base de datos que soporte al menos la funcionalidad mínima de SQL. Otra excepción es que el componente ODBC de los Componentes de Acceso

a Datos de Microsoft (MDAC) SDK provee una librería de cursor que simula cursores enrollables para los controladores que implementan un cierto nivel de funcionalidad.

Las aplicaciones que usan ODBC son responsables por cualquier funcionalidad de base de datos cruzada. Por ejemplo, ODBC no es motor heterogéneo de unión, ni tampoco es un procesador de transacciones distribuidas. Sin embargo, debido a que esto es independiente del DBMS, puede ser usado para construir esas herramientas de bases de datos cruzadas.

ODBC tiene una arquitectura de cuatro componentes: aplicación, administrador de controladores, controladores y fuentes de datos.

El componente de aplicación, realiza el procesamiento y las llamadas a las funciones de ODBC para enviar las expresiones de SQL y recuperar los resultados. El componente del administrador de controladores, carga y descarga los controladores a nombre de una aplicación; procesa las llamadas a funciones de ODBC o pasa las mismas hacia el controlador. El controlador, procesa las llamadas a funciones ODBC, envía las expresiones de SQL a una fuente de datos específica y retorna los resultados a la aplicación; si es necesario, el controlador modifica un requerimiento de la aplicación de tal manera que el requerimiento esté de acuerdo a la sintaxis soportada por el Sistema de Base de Datos asociado. La fuente de datos, consiste en los datos que el usuario desea acceder y su sistema operativo, sistema de base de datos y plataforma de red asociada (si existe) utilizada para acceder al Sistema de Administración de Base de Datos. La figura 3.1 muestra la relación que existe entre estos cuatro componentes.

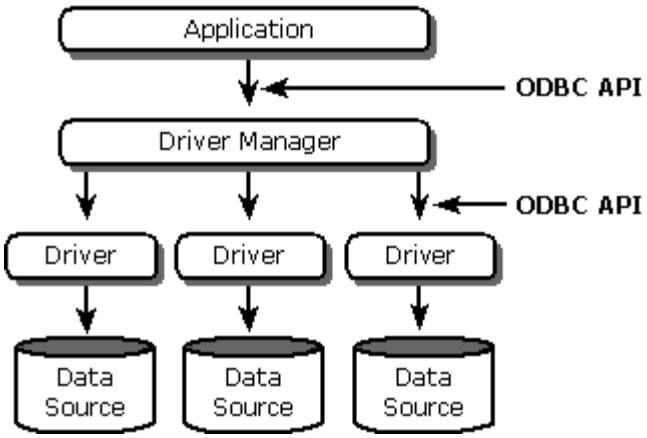

Figura 3.1 – Arquitectura ODBC

Lo importante a notar de esta arquitectura lo describiremos a continuación. Primero, pueden existir múltiples controladores y fuentes de datos, los cuales permiten a la aplicación acceder simultáneamente a los datos de más de una fuente de datos. Segundo, la API de ODBC es usada en dos lugares; entre la aplicación y el administrador de controladores y entre el administrador de controladores y cada controlador. La interfase entre el administrador de controladores y los controladores es algunas veces referida como la interfase del proveedor de servicios (SPI). Para ODBC, la interfase de programación de aplicaciones (API) y la interfase del proveedor de servicios (SPI) es la misma; esto quiere decir que el administrador de controladores y cada controlador tienen la misma interfase hacia las mismas funciones.

Debido a las capacidades y funciones antes mencionadas que soporta ODBC, podríamos hacer uso de esta tecnología para la construcción de nuestra aplicación; sin embargo ahora es un estándar antiguo que ha sido reemplazado por OLEDB debido a su rapidez de conexión, ejecución y demás funciones nuevas incorporadas.

Para realizar una conexión mediante ODBC en una aplicación primero debemos configurarla dentro del sistema operativo; existen tres tipos de conexión ODBC: DSN de usuario, DSN de sistema y DSN de archivo. El nombre de fuente de datos (DSN) es el conjunto de parámetros específicos para una conexión determinada, es decir; el controlador de base de datos, el nombre de instancia, el host que hospeda la base de datos, los parámetros de

comunicación, la seguridad de conexión, etc. Existen diferencias marcadas entre los tres tipos de DSN y son las siguientes:

- DSN de Usuario.- almacena la información de cómo conectarse al proveedor de datos indicado. Una fuente de datos de usuario sólo es visible para si mismo y sólo puede ser usada en la computadora actual.
- DSN de Sistema.- almacena la información de cómo conectarse al proveedor de datos indicado. Una fuente de datos de sistema es visible para todos los usuarios incluyendo los servicios de NT y sólo puede ser usada en la computadora actual.
- DSN de Archivo.- permite la conexión al proveedor de datos indicado. Los DSN de archivo pueden ser compartidos por los usuarios que tengan los mismos controladores instalados.

A continuación se mostrarán los pasos para realizar una conexión hacia un proveedor de datos de Microsoft SQL Server 2000 y de Oracle Database 9i debido a que utilizaremos estos sistemas de base de datos para la aplicación. Primero se debe colocar en el Panel de Control y buscar el icono que indica orígenes de datos (ODBC) o fuentes de datos de 32 bits (ODBC), dependiendo de la versión de Windows que se utilice se encontrará dentro de la carpeta de Herramientas Administrativas o en la raíz del panel de control.

Figura 3.2 – Icono del Panel de Control para acceder a ODBC

En la figura 3.3 se muestra la pantalla inicial cuando se ejecuta el administrador de fuentes de datos, que se encontrará en la pestaña de DSN de usuario y listará todos los DSN's configurados hasta el momento.

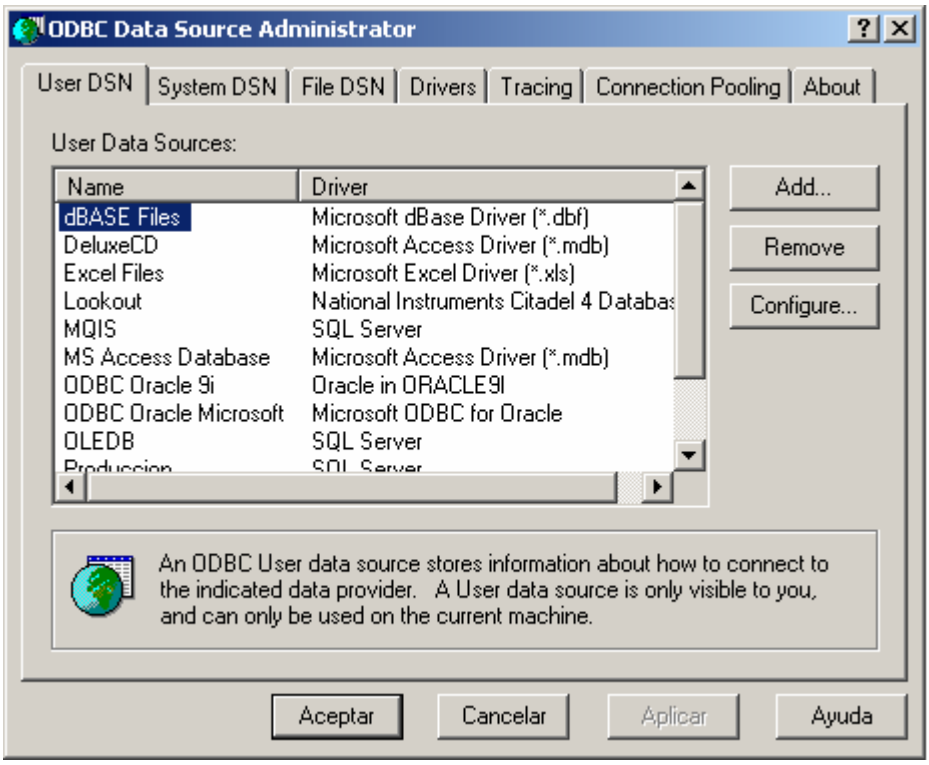

Figura 3.3 – Pantalla inicial del Administrador de Fuentes de Datos

A continuación se pulsará el botón de agregar (Add…) y aparecerá un cuadro de dialogo que listará todos los controladores instalados en el computador que se muestra en la figura 3.4.

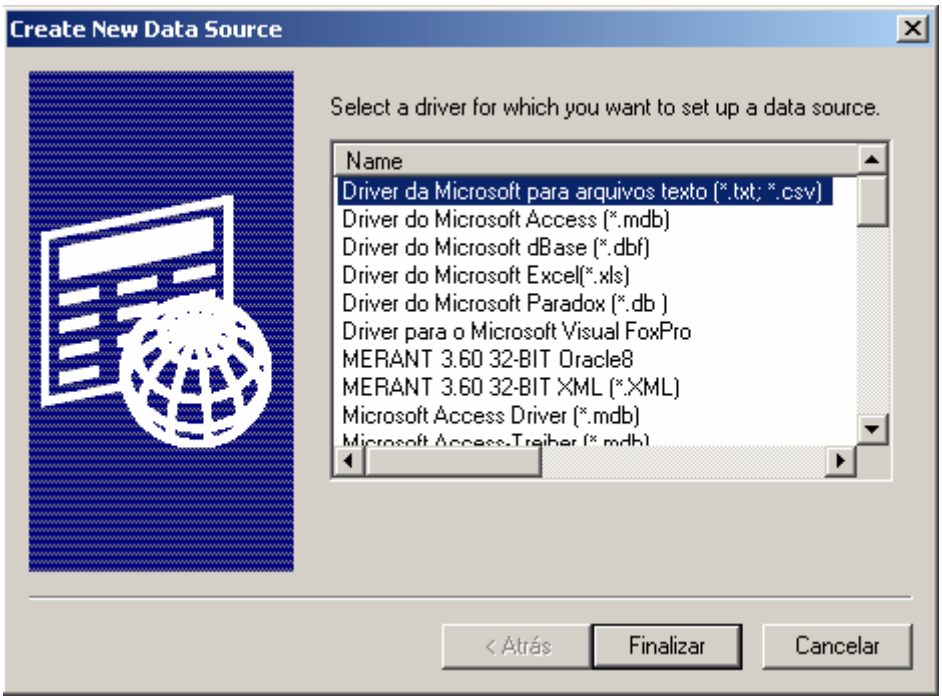

Figura 3.4 – Pantalla de selección de controladores para un DSN nuevo

Luego se selecciona el controlador de acuerdo al sistema de base de datos a utilizar en la conexión y se pulsa el botón Finalizar. Inmediatamente después aparece el cuadro de dialogo de configuración del controlador escogido; en la figura 3.5 se muestra para un controlador de SQL Server donde se ingresa el nombre de DSN, una descripción y el nombre del servidor en el que se encuentra instalado SQL Server. Para la aplicación que se está realizando existen dos opciones: (local) y PENTIUM4; PENTIUM4 es el nombre del computador donde se está realizando la aplicación, (local) siempre aparecerá refiriéndose al computador actual, luego se pulsa siguiente.

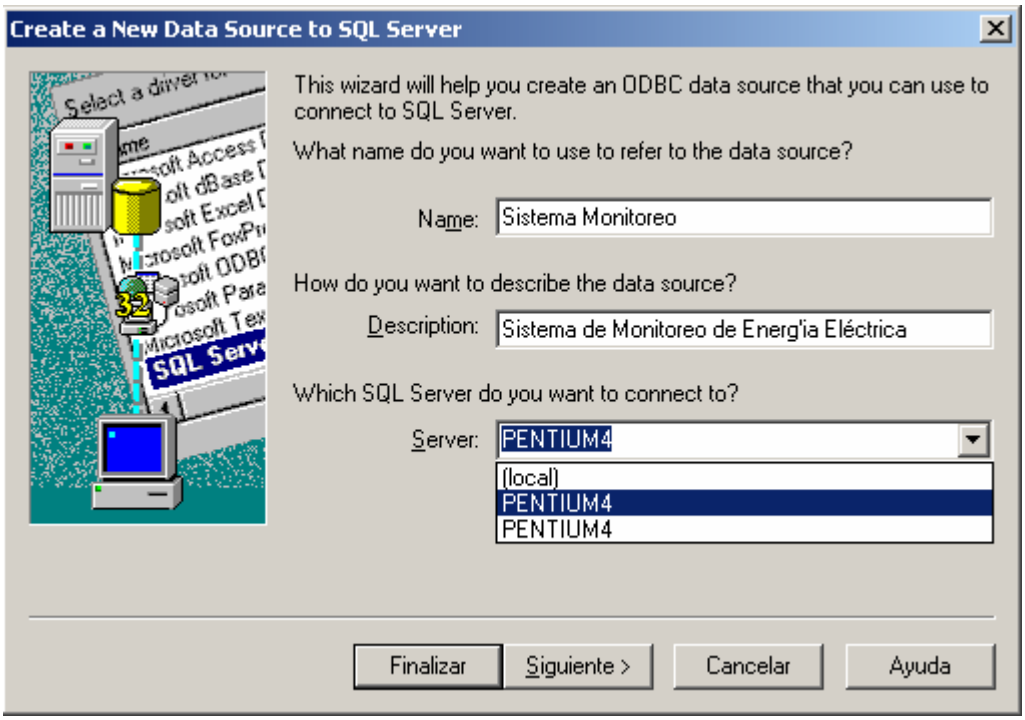

Figura 3.5 – Parámetros básicos del controlador de SQL Server

En la figura 3.6 se muestra el segundo cuadro de diálogo para SQL Server, donde se ingresan las opciones de seguridad y autentificación del usuario de la base de datos.

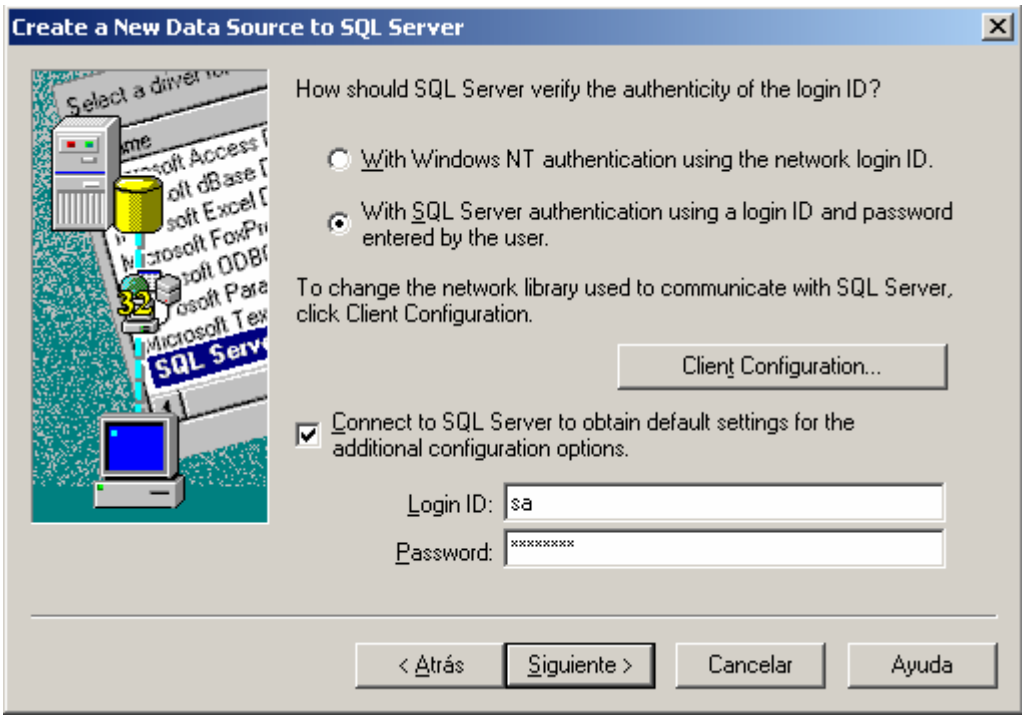

Figura 3.6 – Opciones de seguridad y autentificación del controlador de SQL Server

El método de autentificación con el ID de red sólo sirve cuando el cliente es un usuario del dominio de Windows NT, por lo tanto este método no sirve cuando no existe un dominio en una red Microsoft o los clientes utilizan sistemas operativos heterogéneos. Además al requerir una clave de acceso a la red y otra clave para ingreso al sistema de datos hace más difícil el romper la seguridad.

En la figura 3.7 se muestra el tercer cuadro de dialogo donde se selecciona la base de datos inicial, que puede ser una ya existente o que se adjunten los archivos en el momento de la conexión. Además aquí se indica como se utilizarán ciertos símbolos de puntuación y especiales como las comillas simples, los nulos, los rellenos y las advertencias; lo más recomendable es ajustarse a los estándares de ANSI para que la aplicación sea lo más independiente del sistema de base de datos que se planee utilizar.

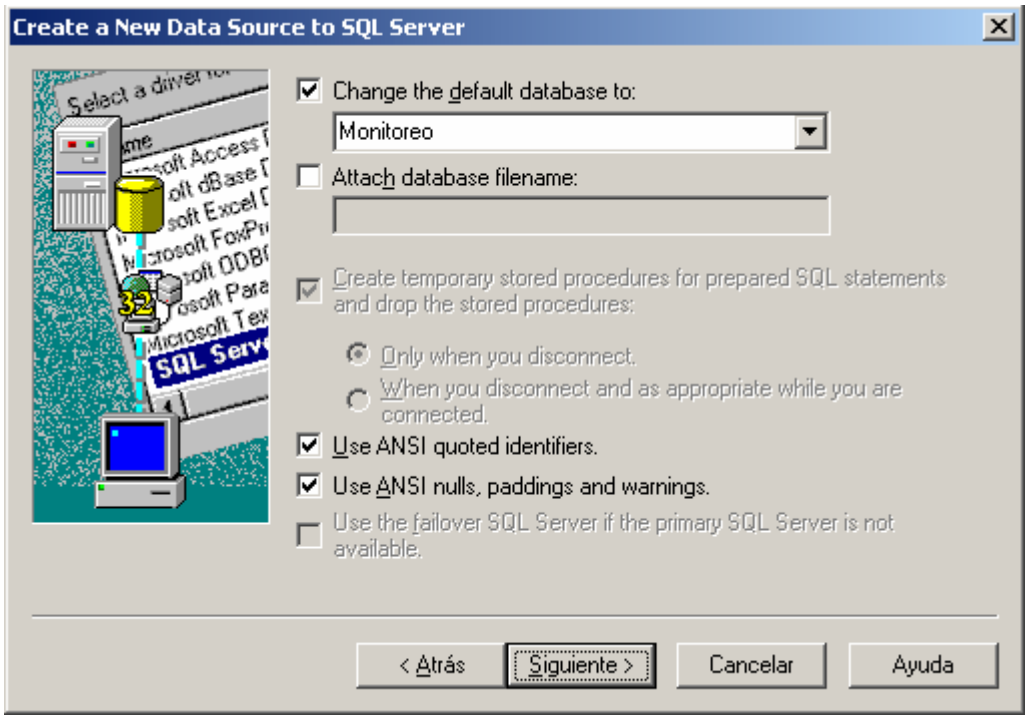

Figura 3.7 – Selección de la base de datos inicial del controlador de SQL Server

En la figura 3.8 se muestra el cuarto cuadro de dialogo del controlador de SQL Server, donde se ingresan opciones de idioma de los mensajes del sistema, encriptación de datos, traducción de los caracteres de datos, configuración regional para tiempo, fechas, números y moneda y: el almacenamiento de estadísticas del controlador en su ejecución. Con estos parámetros se ha terminado de configurar los parámetros de conexión y comportamiento del controlador de SQL Server, a continuación se pulsa finalizar y aparece un cuadro de dialogo de resumen que permite verificar la configuración del mismo.

En la figura 3.9 se muestra el quinto cuadro de dialogo del controlador de SQL Server, donde se indica la versión del controlador de ODBC, un resumen de los parámetros ingresados y además permite realizar una prueba de conexión para verificar que ésta se ejecuta sin problemas, para continuar con el desarrollo de la aplicación

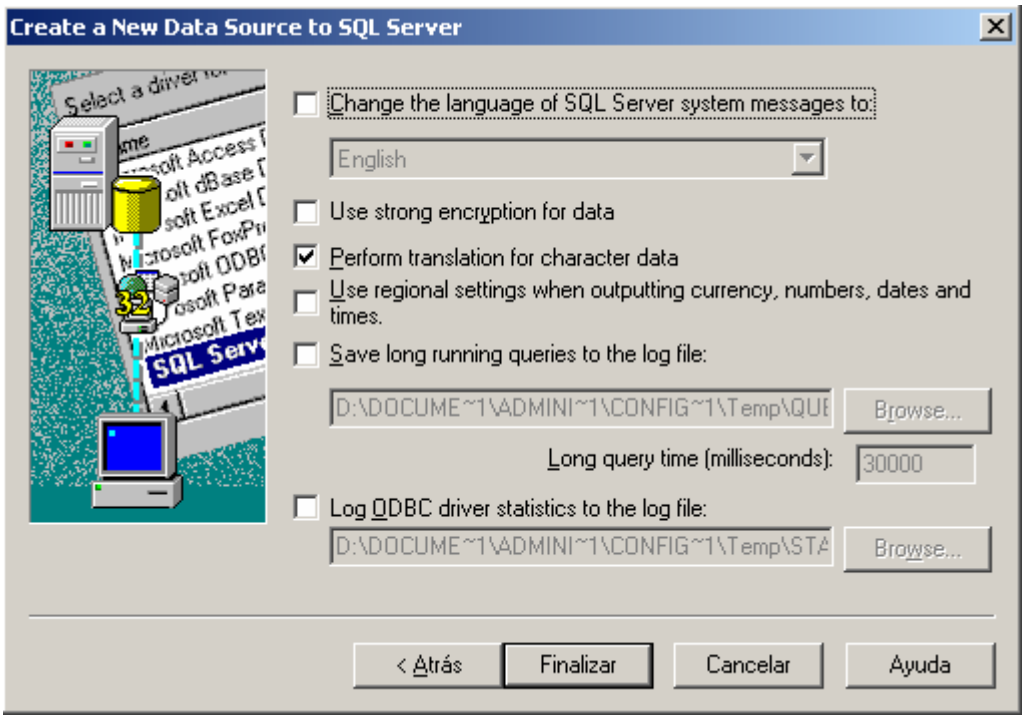

Figura 3.8 – Opciones de idioma, encriptación y configuración regional del controlador de SQL Server

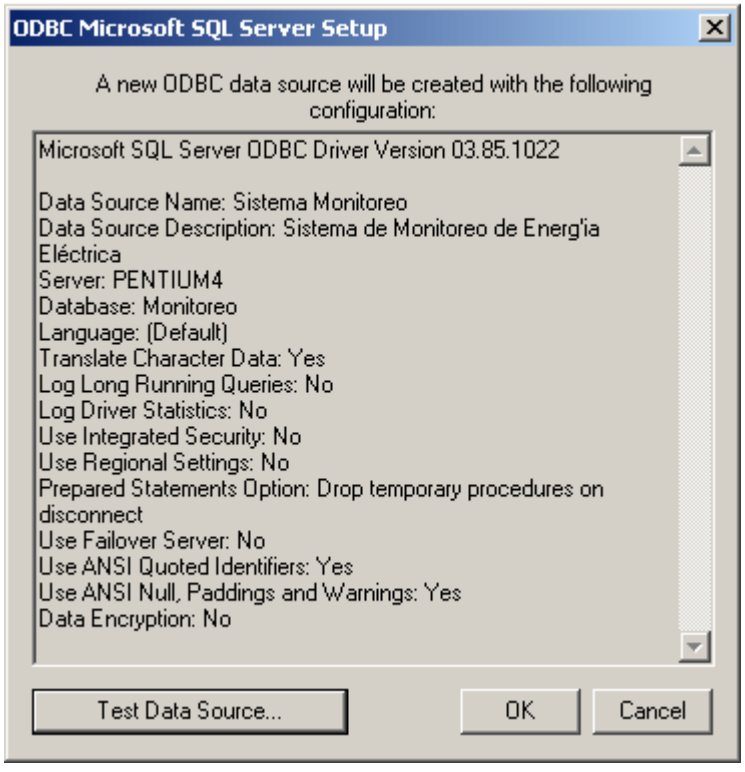

Figura 3.9 – Resumen de los parámetros del controlador de SQL Server.

En la figura 3.10 se muestran las fases de prueba que se ejecutan y un mensaje indicando el éxito o fallo de la misma, con las razones por las que no se pudo realizar la conexión a la base de datos.

Figura 3.10 – Informe del éxito de conexión a la base de datos mediante SQL Server

Una vez terminada la configuración y prueba de la conexión se pulsa el botón aceptar, que automáticamente cierra el cuadro de dialogo del controlador y envía al usuario al cuadro de dialogo de los DSN. En la figura 3.11 se puede visualizar que el DSN ya está creado con el nombre de ‘Sistema Monitoreo’, además indica el controlador que se ha utilizado para realizar la conexión.

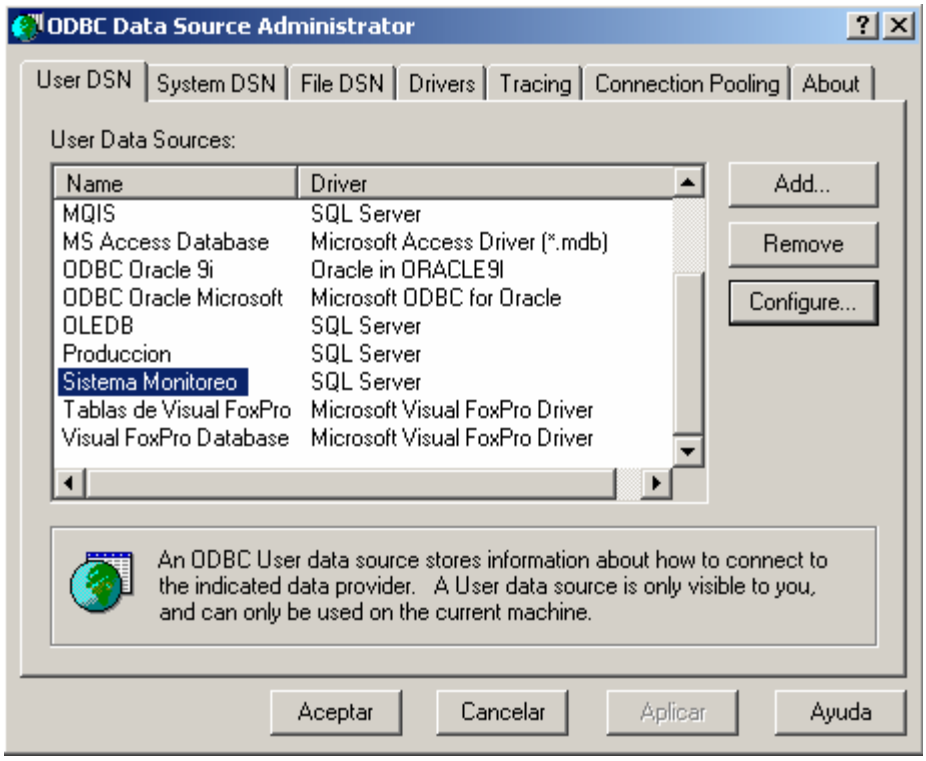

Figura 3.11 – DSN creado con el controlador de SQL Server

Para visualizar la diferencia de los cuadros de dialogo de cada controlador y debido a que también se utilizará Oracle Database 9i para la aplicación, se mostrará también paso a paso la configuración del mismo. En la figura 3.12 se visualiza el único cuadro de dialogo que utiliza el controlador de Oracle.

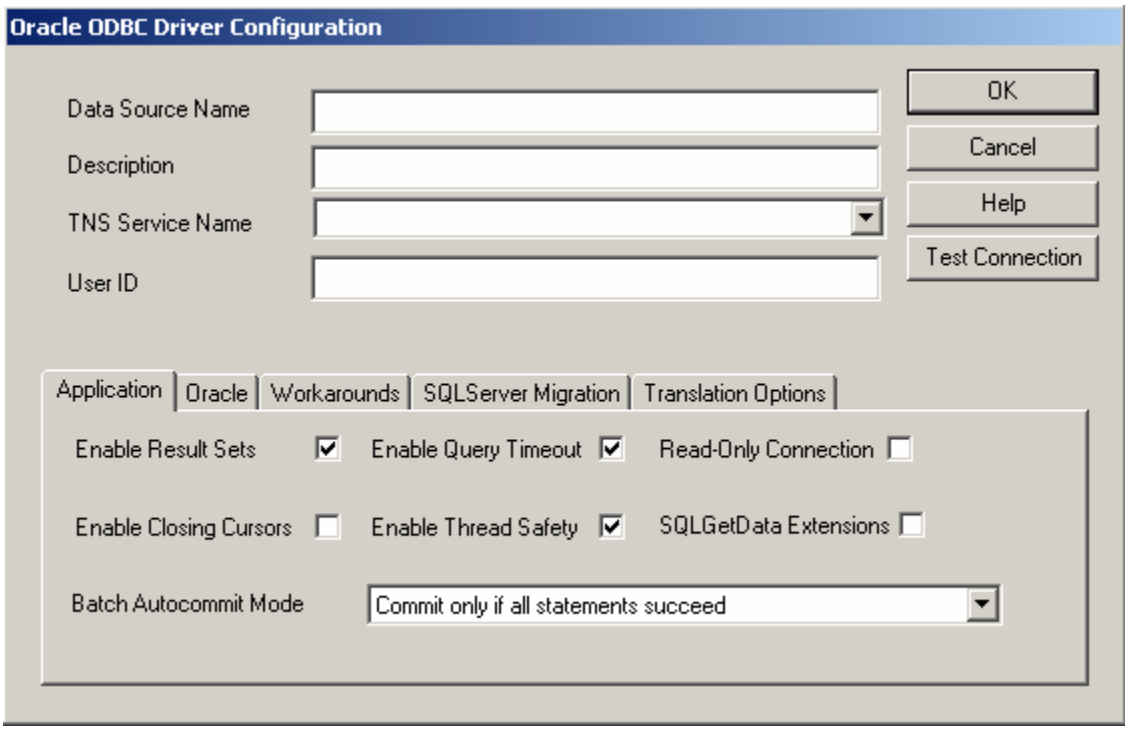

Figura 3.12 – Cuadro de dialogo del controlador de Oracle

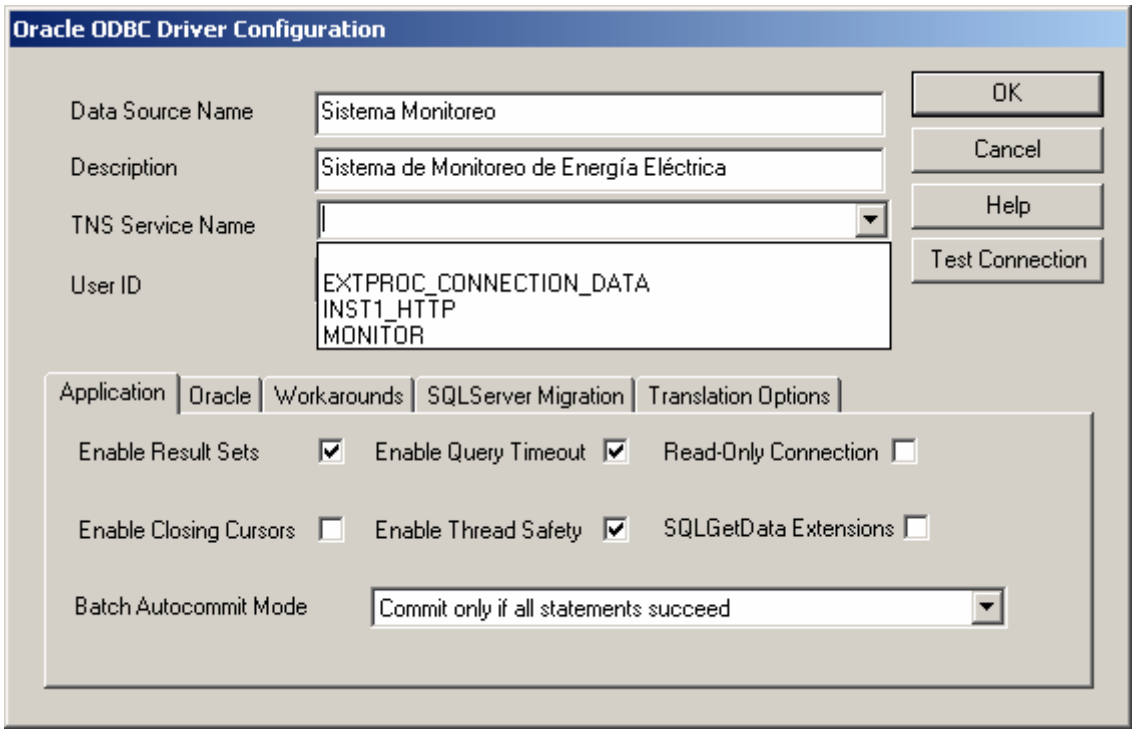

Figura 3.13 – Opciones más comunes del controlador de Oracle

En la figura 3.13 se muestra el cuadro de dialogo del controlador con algunos parámetros ingresados como el nombre de la fuente de datos, la descripción y una lista con los nombres de servicio de TNS. El sistema de base de datos de Oracle se instaló con el nombre de MONITOR, por lo tanto se seleccionará ese como el nombre de servicio TNS. Finalmente existe la casilla para el ID de usuario; las demás opciones que se muestran en las otras pestañas se puede dejarlas intactas porque de esta manera se conserva el estándar ANSI. Al pulsar el botón de probar conexión aparece el cuadro de dialogo de la figura 3.14 donde se debe ingresar la clave del usuario. En la figura 3.15 se muestra el estado de la prueba de conexión a la base de datos, indicando éxito o fallo con las debidas razones del mismo.

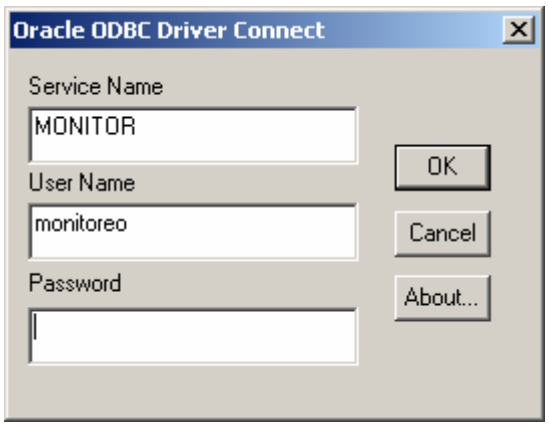

Figura 3.14 – Ingreso de la clave para probar la conexión del controlador de Oracle

Figura 3.15 – Informe del éxito de conexión a la base de datos mediante Oracle 9i

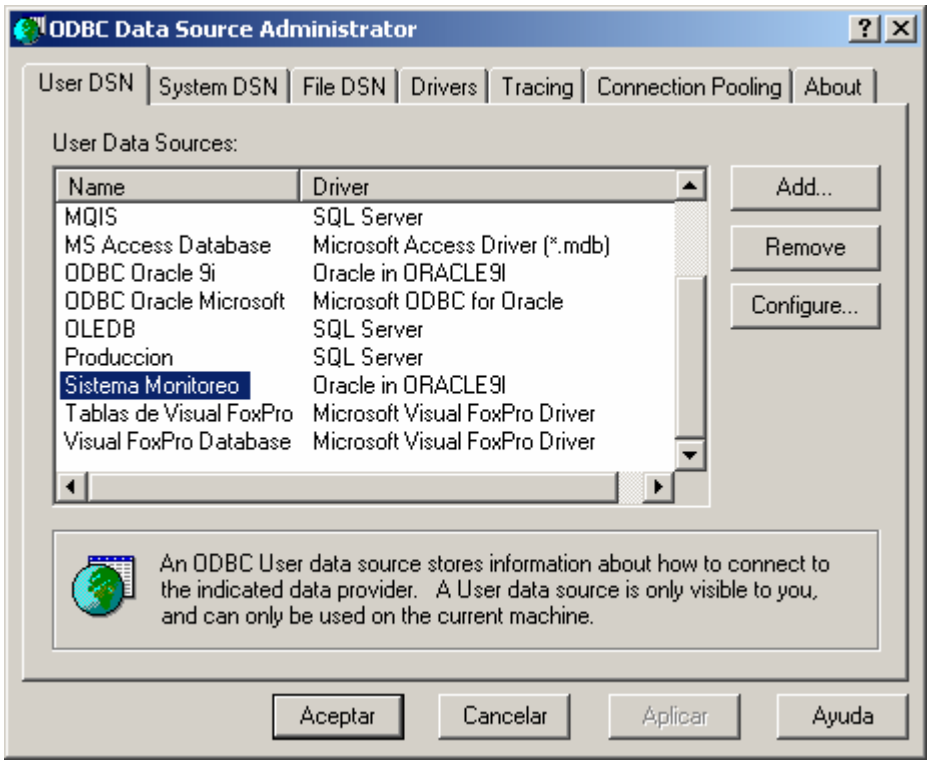

Figura 3.16 – DSN creado con el controlador de Oracle Database 9i

Una vez terminada la configuración y prueba de la conexión se pulsa el botón aceptar, que automáticamente cierra el cuadro de dialogo del controlador y envía al usuario al cuadro de dialogo de los DSN. En la figura 3.16 se puede visualizar que el DSN ya está creado con el nombre de ‘Sistema Monitoreo’, además indica el controlador que se ha utilizado para realizar la conexión.

3.1.2. JDBC.- significa la conectividad de bases de datos en Java. Actualmente se encuentra en la versión 3.0 de la API de JDBC creada por Sun, soporta conjuntos de resultados enrollables y los tipos de datos de SQL 1999. La API de JDBC está escrita totalmente en Java a diferencia de la API de ODBC que está escrita en C y es muy difícil de manejar, por lo cual una traducción del ODBC no es recomendable, además JDBC posee menos controladores que su opuesto. Sin embargo, se debe utilizar JDBC cuando se desea crear una solución puramente de Java.

Los controladores de JDBC sirven de forma adecuada para un modelo de dos capas como para un modelo de N capas, donde los mismos se encargan de conectarse y comunicarse con el sistema de base de datos. JDBC consta de tres módulos principales que son:

- Administrador de controladores de JDBC.- es la médula espinal de la arquitectura de JDBC, es bastante pequeño y simple. Su función primaria es conectar las aplicaciones de java al controlador de JDBC correcto y salir.
- Suite de pruebas del controlador de JDBC.- proporcionan una relación de confianza entre los controladores de JDBC y la aplicación. Sólo pueden utilizarse los controladores de la suite de JDBC.
- Puente JDBC-ODBC.- permite a los controladores de ODBC ser usados como controladores de JDBC. A largo plazo proporcionará una manera de acceder a algunos DBMS menos populares si no se crean los controladores de JDBC para esos sistemas de bases de datos.

Por las razones que se han mencionado anteriormente, como estar desarrollado sobre Java y no permitir un manejo con lenguaje C, no es apto para el desarrollo de la aplicación.

3.1.3. OLEDB.- significa objetos enlazados y empotrados para bases de datos. Es una especificación de interfaces. La capa OLEDB expone interfaces COM (Component Object Model) de bajo nivel a las que se puede llamar desde lenguajes como Visual Basic, expone también interfaces COM de alto nivel. Estos componentes implementan por lo general servicios como el de agrupación de recursos.

Un proveedor de OLEDB implementa las interfaces de OLEDB. Permite al usuario tener acceso a cualquier tipo de origen de datos de manera uniforme. En cierto sentido, un proveedor de OLEDB es similar a un controlador de ODBC que ofrece un mecanismo de acceso uniforme a datos relacionales, además también proporcionan la misma clase de acceso a los datos no relacionales. Los proveedores OLEDB se han creado partiendo del modelo COM, mientras que los controladores de ODBC se basan en una especificación de API en C.

OLEDB no pretende reemplazar a ODBC, ya que una conexión ODBC es ideal para acceder a bases de datos en SQL, sin embargo existe un controlador para OLEDB que permite a un cliente 'hablar' con un proveedor ODBC.

Los proveedores de OLEDB más conocidos son los siguientes:

- OLEDB para ODBC, llamada KAGERA o MDASQL
- OLEDB para SQL Server, llamado SQLOLEDB
- OLEDB para Oracle, llamado MSDAORA
- OLEDB para Jet, llamado Jolt

OLEDB está entre la capa de ODBC y la aplicación. En una aplicación de internet, con páginas ASP por ejemplo, ADO es la "aplicación" que está sobre OLEDB. Las llamadas a ADO son enviadas primero a OLEDB, las cuales son luego enviadas a la capa de ODBC. Se puede conectar directamente a la capa de OLEDB, y haciendo esto, se verá un incremento en el rendimiento de los cursores del lado del servidor (el tipo de conjunto de registros predeterminado, y el tipo de cursor más usado). Por lo tanto, la recomendación más obvia es utilizar conexiones OLEDB en lugar de conexiones ODBC en la aplicación, para de esta manera obtener un mejor rendimiento.

Debido a que OLEDB es multiplataforma, Windows no dispone de un método directo ni lugar específico para realizar una conexión de las mismas. Para realizar una conexión OLEDB se crea un archivo de texto vacío, luego se lo renombra con la extensión 'udl' de Universal Data Link. En la figura 3.17 se puede observar un ejemplo.

Figura 3.17 – Icono de un Enlace Universal de Datos (OLEDB)

Inmediatamente después se da un doble clic sobre el icono del enlace de datos, se abre un cuadro de dialogo que corresponde a las propiedades del mismo y que se encuentran vacías porque el enlace de datos no se encuentra configurado aún. Debido a que los programadores de aplicaciones todavía usan los controladores de ODBC, este cuadro de dialogo se coloca en la segunda pestaña con el proveedor de OLEDB para ODBC seleccionado. En la figura 3.18 se puede observar esto debido a que indica que se seleccione el data source name (DSN) de ODBC.

En la figura 3.19 se puede observar que se encuentra la configuración en la pestaña del proveedor de datos y aquí se ha seleccionado el proveedor OLEDB de SQL Server, a continuación se pulsa siguiente.

En la figura 3.20 se observa el cuadro de dialogo de configuración del proveedor OLEDB de SQL Server con los parámetros básicos: nombre del servidor de datos, método de autentificación, nombre de usuario, clave y nombre de la base de datos; que son los mismos que en el caso del controlador de ODBC con la diferencia de que aquí no se necesita conocer muchos parámetros complicados como la encriptación, estándares de ANSI, configuraciones regionales, traducción de caracteres, etc.

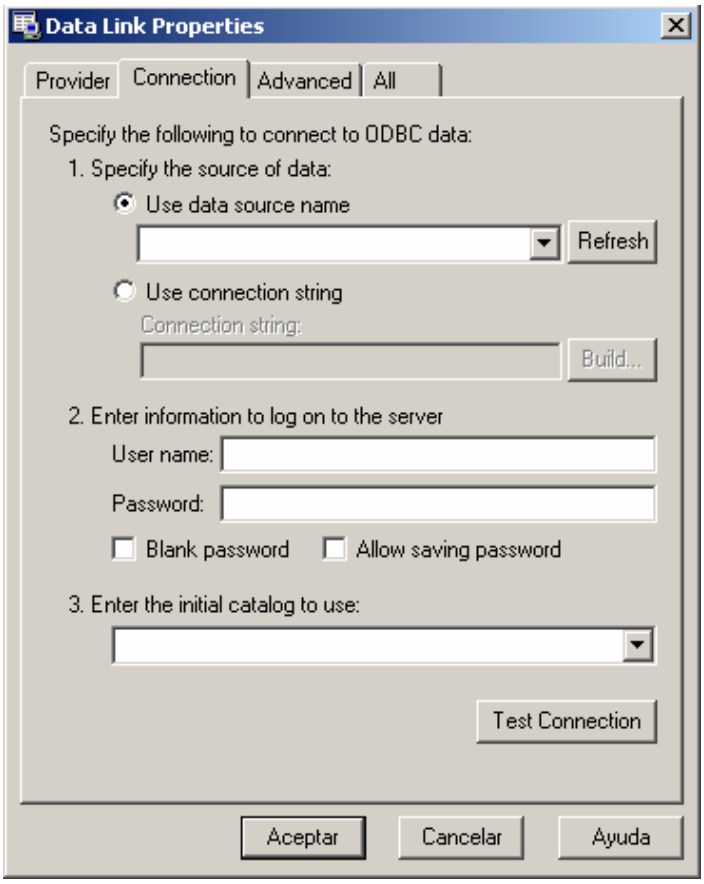

Figura 3.18 – Enlace de datos sin configurar, predeterminado OLEDB para ODBC

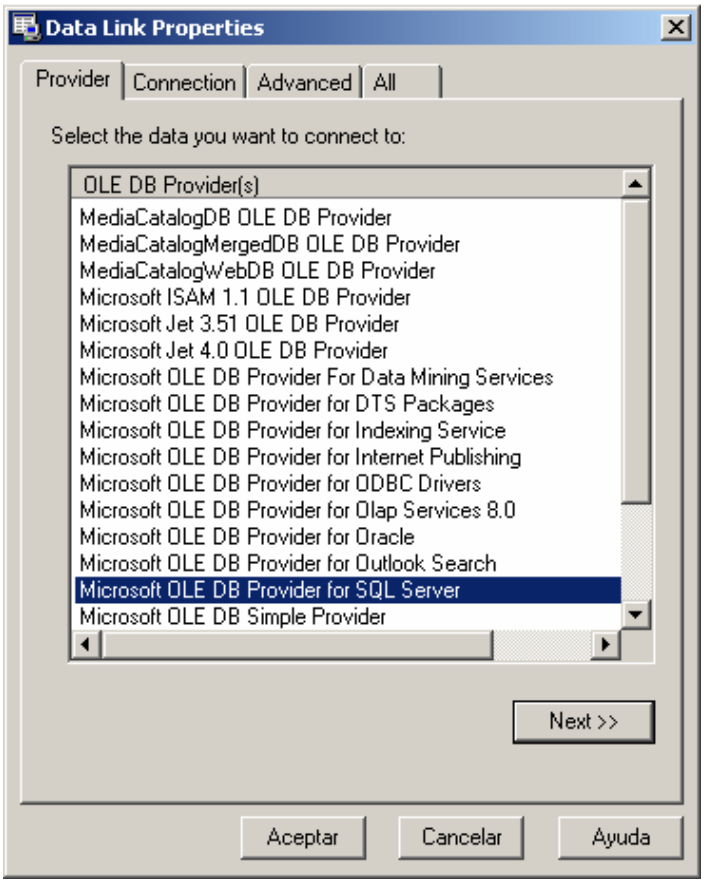

Figura 3.19 – Selección del proveedor de OLEDB para SQL Server

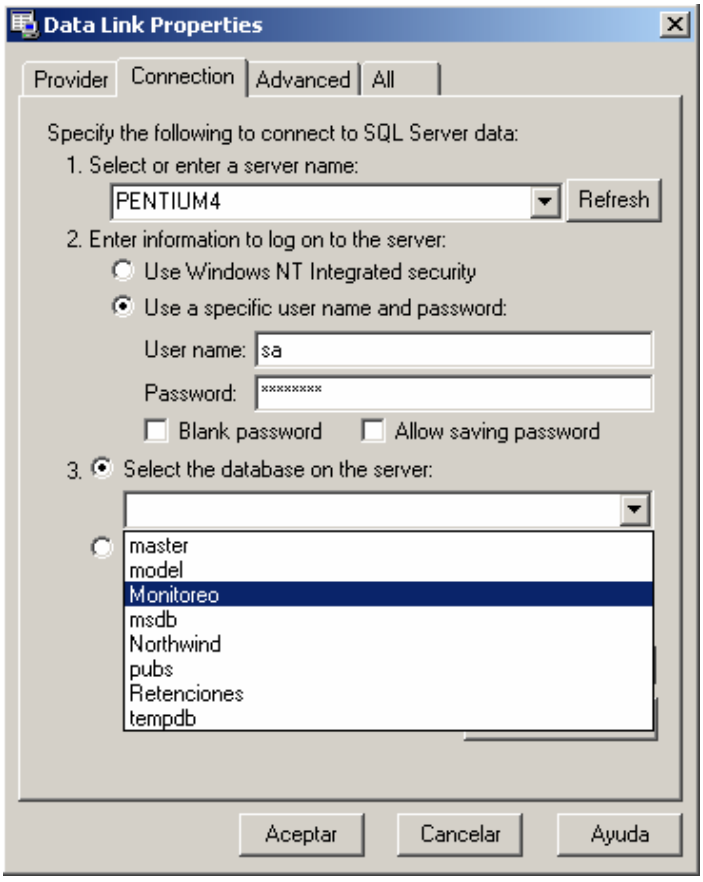

Figura 3.20 – Configuración del proveedor OLEDB para SQL Server

Después de la selección de la base de datos, se puede probar la conexión con la misma mediante el botón 'Test Connection'. En la figura 3.21 se puede observar la respuesta del proveedor de SQL Server indicando una conexión exitosa, que a la vez utiliza un cuadro de dialogo genérico para su respuesta.

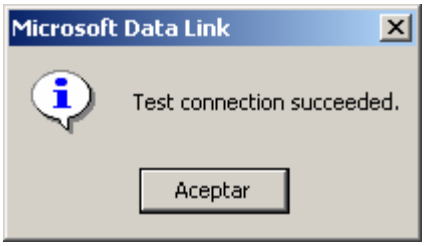

Figura 3.21 – Respuesta del proveedor OLEDB de conexión exitosa

A continuación, se realizará el proceso utilizando el proveedor de OLEDB de Oracle. En la figura 3.22 se puede ver la selección del proveedor de Oracle, luego se pulsa siguiente.

En la figura 3.23 se observa el cuadro de dialogo de configuración del proveedor OLEDB de Oracle con los parámetros básicos: nombre de la instancia de la base de datos, nombre de usuario y clave; que son los mismos que en el caso del controlador de ODBC con la diferencia de que aquí no se necesita conocer muchos parámetros complicados como el comportamiento de la aplicación y de Oracle, manejo de errores, estándares de ANSI, migración de SQL Server, opciones de traducción, etc.

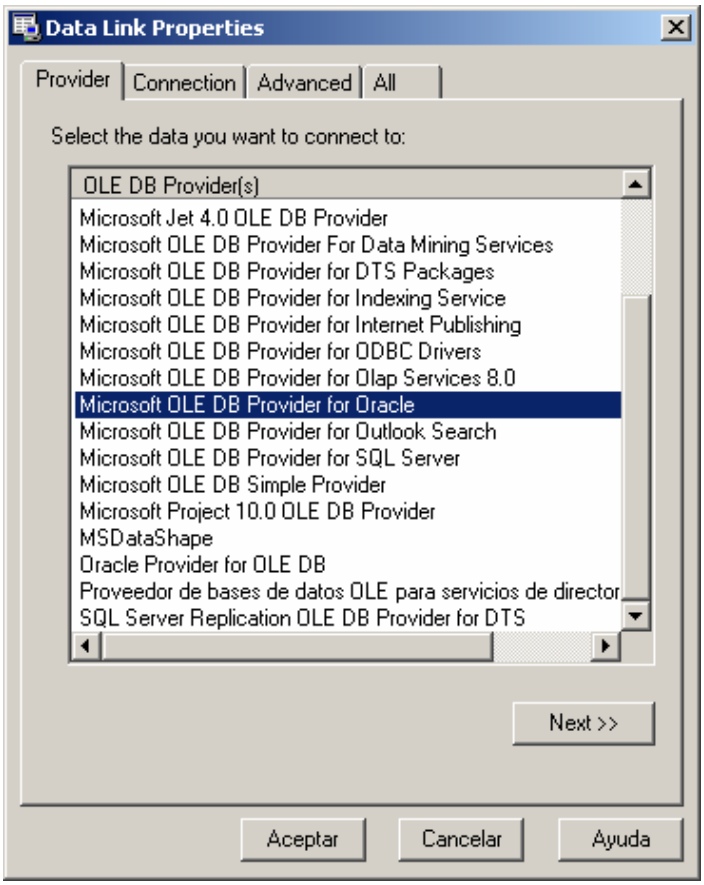

Figura 3.22 – Selección del proveedor de OLEDB para Oracle

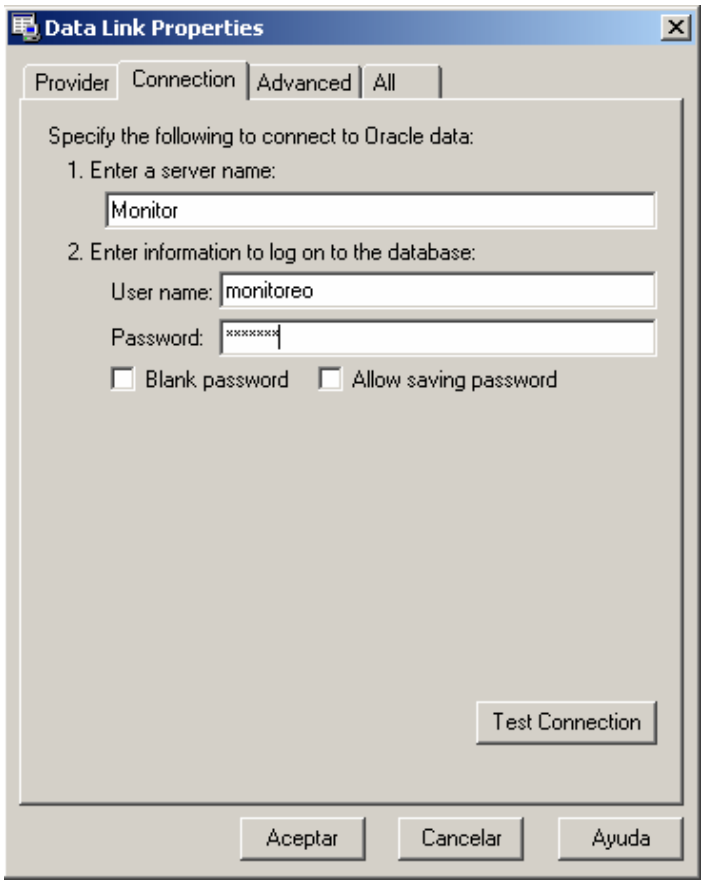

Figura 3.23 – Configuración del proveedor OLEDB para Oracle

En el proveedor de OLEDB para Oracle no se necesita saber el nombre del servidor de base de datos porque el servicio TNS Listener propio de Oracle se encarga de esta localización, lo que se necesita saber es el nombre de la instancia de la base de datos. Se puede probar la conexión con la misma mediante el botón 'Test Connection'. En la figura 3.24 se puede observar la respuesta del proveedor de Oracle indicando una conexión exitosa, que a la vez utiliza un cuadro de dialogo genérico para su respuesta.

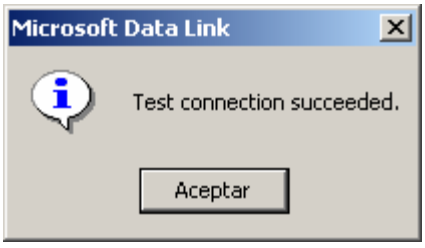

Figura 3.24 – Respuesta del proveedor OLEDB de conexión exitosa

Finalmente, se realizará el proceso utilizando el proveedor de OLEDB para controladores de ODBC. En la figura 3.25 se puede ver la selección del proveedor de OLEDB para ODBC, luego se pulsa siguiente.

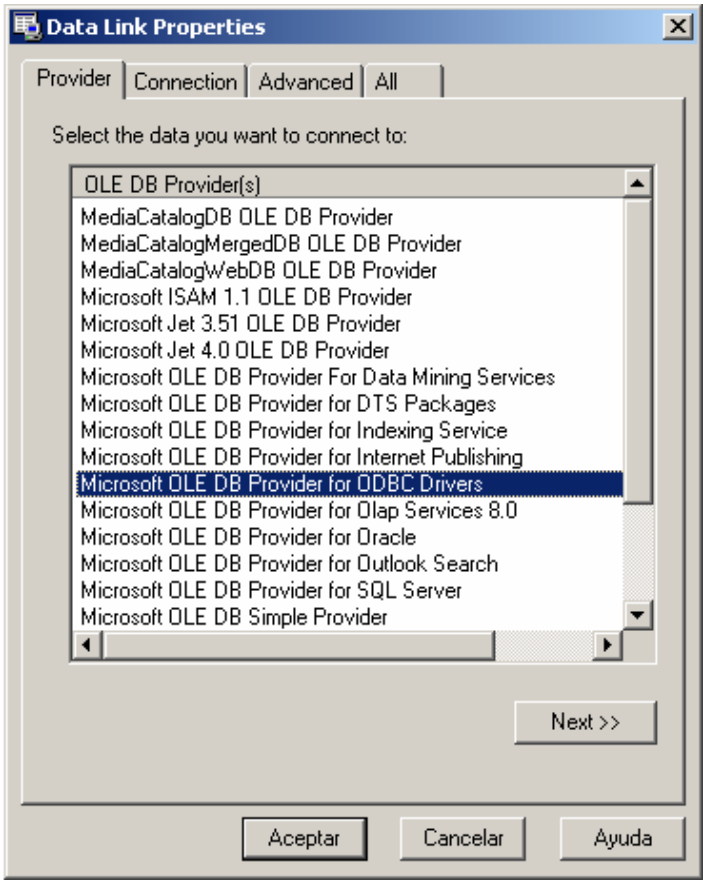

Figura 3.25 – Selección del proveedor de OLEDB para ODBC

En la figura 3.26 se observa el cuadro de dialogo de configuración del proveedor OLEDB para controladores de ODBC donde se debe seleccionar el DSN (Data Source Name), nombre de usuario, clave y seleccionar el nombre de la base de datos en el caso de SQL Server; aquí el proveedor el resto de parámetros complejos de la configuración del DSN.

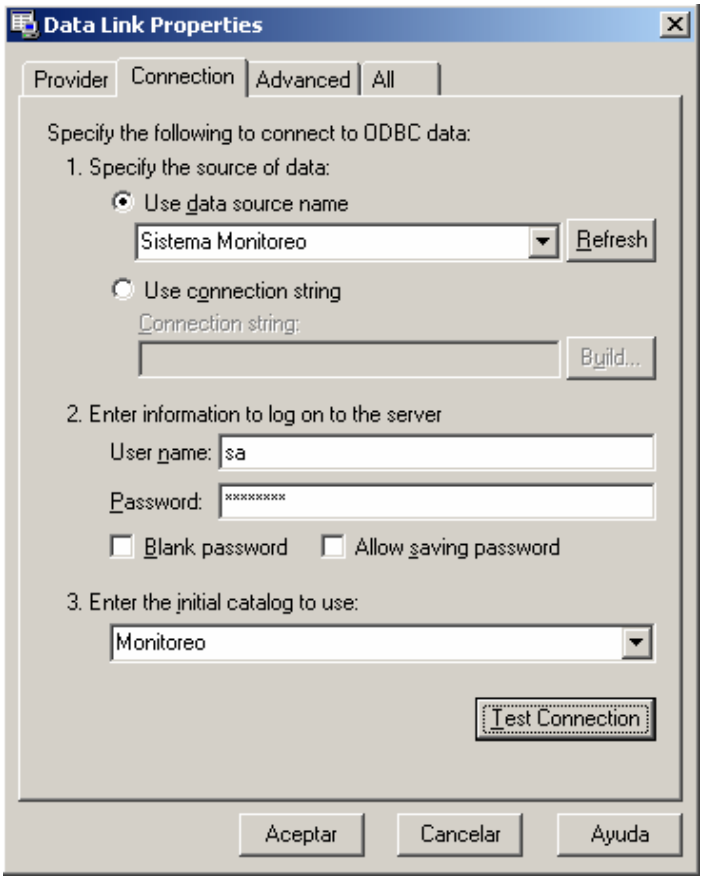

Figura 3.26 – Configuración del proveedor OLEDB para ODBC

Después de la selección de la base de datos, se puede probar la conexión con la misma mediante el botón ‘Test Connection’. En la figura 3.27 se puede observar la respuesta del proveedor de OLEDB para ODBC indicando una conexión exitosa, que a la vez utiliza un cuadro de dialogo genérico para su respuesta.

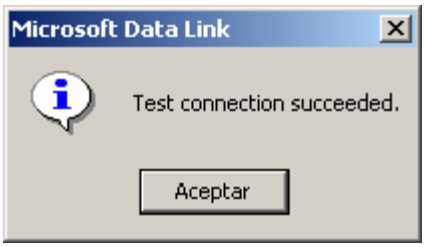

Figura 3.27 – Respuesta del proveedor OLEDB de conexión exitosa

Sólo en el caso de que exista un error, se obtendrá un mensaje en el idioma predeterminado de la base de datos y con algunas referencias que indicarán a que sistema de base de datos se está conectando.

3.2. ESTRUCTURAS DE ALMACENAMIENTO Y MANIPULACIÓN ESTÁNDAR DE LA INFORMACIÓN

Para realizar la manipulación de la información y las estructuras que contienen la misma en los DBMS se hará uso del lenguaje SQL (Structured Query Language) estándar. El lenguaje SQL está divido en dos partes que son:

- DDL (Data Definition Language)
- DML (Data Manipulation Language)

3.2.1. Lenguaje de Definición de Datos (DDL).- es el que está encargado de la definición y administración de las estructuras de datos que almacenarán la información. Existen pocas instrucciones muy sencillas para la ejecución de estas tareas; el análisis se centrará sobre las instrucciones de definición de tablas debido a que son las más usadas y las únicas que se utilizarán en el presente proyecto, siendo las siguientes:

- Create table
- Alter table
- Drop table

3.2.1.1. Create table.- sirve para crear una nueva tabla.

Sintáxis

```
  CREATE TABLE
[ nombre_base_datos.[ propietario ] . | propietario. ] nombre_tabla
( { < definición_columna >
    | nombre_columna AS expresión_columna_calculada
    | < restricción_tabla > ::= [ CONSTRAINT nombre_restricción ] }
          | [ { PRIMARY KEY | UNIQUE } [ ,...n ]
)

[ ON { grupo_filas | DEFAULT } ]
[ TEXTIMAGE_ON { grupo_filas | DEFAULT } ]

< definición_columna > ::= { nombre_columna tipo_dato }
  [ COLLATE < nombre_comparación > ]
  [ [ DEFAULT expresión_constante ]
    | [ IDENTITY [ ( semilla , incremento ) [ NOT FOR REPLICATION ] ] ]
  ]
  [ ROWGUIDCOL]
  [ < restricción_columna > ] [ ...n ]

< restricción_columna > ::= [ CONSTRAINT nombre_restricción ]
  { [ NULL | NOT NULL ]
    | [ { PRIMARY KEY | UNIQUE }
      [ CLUSTERED | NONCLUSTERED ]
      [ WITH FILLFACTOR = factor_relleno ]
      [ON {grupo_filas | DEFAULT} ] ]
    ]
    | [ [ FOREIGN KEY ]
      REFERENCES referencia_tabla[ ( referencia_columna) ]
      [ ON DELETE { CASCADE | NO ACTION } ]
      [ ON UPDATE { CASCADE | NO ACTION } ]
      [ NOT FOR REPLICATION ]
```

```
    ]
    | CHECK [ NOT FOR REPLICATION ]
    ( expresión_lógica)
  }
   < restricción_tabla > ::= [ CONSTRAINT nombre_restricción ]
  { [ { PRIMARY KEY | UNIQUE }
    [ CLUSTERED | NONCLUSTERED ]
    { ( columna [ ASC | DESC ] [ ,...n ] ) }
    [ WITH FILLFACTOR = factor_relleno ]
    [ ON { grupo_filas | DEFAULT } ]
  ]
  | FOREIGN KEY
    [ ( columna [ ,...n ] ) ]
    REFERENCES referencia_tabla[ ( referencia_columna[ ,...n ] ) ]
    [ ON DELETE { CASCADE | NO ACTION } ]
    [ ON UPDATE { CASCADE | NO ACTION } ]
    [ NOT FOR REPLICATION ]
  | CHECK [ NOT FOR REPLICATION ]
    ( condiciones_búsqueda)
  }
```

3.2.1.2. Alter table.- sirve para modificar la definición de una tabla por alteración, adición o eliminación de columnas y restricciones; o por deshabilitamiento o habilitamiento de restricciones o disparadores.

Sintáxis

```
    ALTER TABLE tabla
{ [ ALTER COLUMN nombre_columna
  { nuevo_tipo_dato [ ( precisión  [ , escala ] ) ]
```

```
    [ COLLATE < nombre_comparación > ]
    [ NULL | NOT NULL ]
    | {ADD | DROP } ROWGUIDCOL }
  ]
  | ADD
    { [ < definición_columna > ]
    | nombre_columna AS expresión_columna_calculada
    } [ ,...n ]
  | [ WITH CHECK | WITH NOCHECK ] ADD
    { < restricción_tabla > } [ ,...n ]
  | DROP
    { [ CONSTRAINT ] nombre_restricción
      | COLUMN columna } [ ,...n ]
  | { CHECK | NOCHECK } CONSTRAINT
    { ALL | nombre_restricción [ ,...n ] }
  | { ENABLE | DISABLE } TRIGGER
    { ALL | nombre_disparador[ ,...n ] }
}
   < definición_columna > ::=
  { nombre_columna tipo_dato }
  [ [ DEFAULT expresión_constante ] [ WITH VALUES ]
  | [ IDENTITY [ (semilla , incremento ) [ NOT FOR REPLICATION ] ] ]
    ]
  [ ROWGUIDCOL ]
  [ COLLATE < nombre_comparación > ]
  [ < restricción_columna > ] [ ...n ]
   < restricción_columna > ::=
  [ CONSTRAINT nombre_restricción ]
  { [ NULL | NOT NULL ]
    | [ { PRIMARY KEY | UNIQUE }
      [ CLUSTERED | NONCLUSTERED ]
```

```
        [ WITH FILLFACTOR = factor_relleno ]
        [ ON { grupo_filas | DEFAULT } ]
        ]
    | [ [ FOREIGN KEY ]
        REFERENCES referencia_tabla[ ( referencia_columna) ]
        [ ON DELETE { CASCADE | NO ACTION } ]
        [ ON UPDATE { CASCADE | NO ACTION } ]
        [ NOT FOR REPLICATION ]
        ]
    | CHECK [ NOT FOR REPLICATION ]
        ( expresión_lógica)
}
  < restricción_tabla > ::=
[ CONSTRAINT nombre_restricción ]
{ [ { PRIMARY KEY | UNIQUE }
    [ CLUSTERED | NONCLUSTERED ]
    { ( columna [ ,...n ] ) }
    [ WITH FILLFACTOR = factor_relleno ]
    [ ON {grupo_filas | DEFAULT } ]
    ]
    |   FOREIGN KEY
        [ ( columna [ ,...n ] ) ]
        REFERENCES referencia_tabla[ ( referencia_columna[ ,...n ] ) ]
        [ ON DELETE { CASCADE | NO ACTION } ]
        [ ON UPDATE { CASCADE | NO ACTION } ]
        [ NOT FOR REPLICATION ]
    | DEFAULT expresión_constante
        [ FOR columna ] [ WITH VALUES ]
    |   CHECK [ NOT FOR REPLICATION ]
        ( condiciones_búsqueda)
}
```

3.2.1.3. Drop table.- sirve para eliminar una tabla y todos los datos, índices, disparadores, restricciones y especificaciones de permisos para esa tabla. Cualquier vista o procedimiento almacenado que referencia la tabla eliminada debe ser eliminada explícitamente usando la declaración DROP VIEW o DROP PROCEDURE.

Sintáxis

DROP TABLE *nombre_tabla*

3.2.2. Lenguaje de Manipulación de Datos (DML).- es el que está encargado de la manipulación de los datos como seleccionar, insertar, eliminar o actualizar desde las estructuras de datos creadas con el DDL. Existen pocas instrucciones muy sencillas para la ejecución de estas tareas, son las más usadas y son las siguientes:

- Select
- Insert
- Delete
- Update
- Truncate table

3.2.2.1. Select.- sirve para recuperar filas desde la base de datos y permite la selección de una o más filas o columnas de una o más tablas. La sintaxis completa de la expresión Select es compleja, pero las cláusulas principales pueden ser resumidas como:

SELECT *listas_selección*
[INTO *nueva_tabla*]
FROM *tabla_fuente*
[WHERE *condición_búsqueda*]

[GROUP BY *expresión_agrupar_por*]
[HAVING *condición_búsqueda*]
[ORDER BY *expresión_orden*[ASC | DESC]]

El operador de unión puede ser usado entre consultas para combinar sus resultados en un sólo conjunto de resultado.

Sintáxis

SELECT expresión ::=
< expresión_consulta >
[ORDER BY { *expresión_orden_por*| *posición_columna*[ASC | DESC] }
[**,**...*n*]]
[COMPUTE
{ { AVG | COUNT | MAX | MIN | SUM } **(** *expresión* **)** } [**,**...*n*]
[BY *expresión* [**,**...*n*]]
]
[FOR { BROWSE | XML { RAW | AUTO | EXPLICIT }
[**,** XMLDATA]
[**,** ELEMENTS]
[**,** BINARY base64]
}
]
[OPTION (< indicio_consulta> [**,**...*n*])]
< expresión consulta > ::=
{ < especificación consulta> | (< expresión consulta>) }
[UNION [ALL] < especificación consulta| (< expresión consulta>) [...*n*]]
< especificación consulta> ::=
SELECT [ALL | DISTINCT]
[{ TOP *entero* | TOP *entero* PERCENT } [WITH TIES]]
< listas_selección>

[INTO *nueva_tabla*]
[FROM { < tabla_fuente> } [**,**...*n*]]
[WHERE < condición_búsqueda>]
[GROUP BY [ALL] *expresión_agrupar_por*[**,**...*n*]
  [WITH { CUBE | ROLLUP }]
]
[HAVING < condición_búsqueda>]

3.2.2.2. Insert.- sirve para adicionar una nueva fila a una tabla o una vista.

Sintáxis

INSERT [INTO]
{ *nombre_tabla* WITH **(** < indicio_tabla_limitado> [...*n*] **)**
  | *nombre_vista*
  | *función_filas_limitada*
}

{ [**(** *lista_columnas***)**]
  { VALUES
    ({ DEFAULT | NULL | *expresión* } [**,**...*n*] **)**
    | *tabla_derivada*
    | *ejecutar_expresión*
  }
}
| DEFAULT VALUES
< indicio_tabla_limitado> ::=
{ FASTFIRSTROW
  | HOLDLOCK
  | PAGLOCK

```
    | READCOMMITTED
    | REPEATABLEREAD
    | ROWLOCK
    | SERIALIZABLE
    | TABLOCK
    | TABLOCKX
    | UPDLOCK
}
```

3.2.2.3. Delete.- sirve para eliminar filas de una tabla.

Sintáxis

```
DELETE
[ FROM ]
    { nombre_tabla WITH ( < indicio_tabla_limitado> [ ...n ] )
     | nombre_vista
     | función_filas_limitada
    }

    [ FROM { < tabla_fuente> } [ ,...n ] ]
      [ WHERE
    { < condición_búsqueda>
    | { [ CURRENT OF
          { { [ GLOBAL ] nombre_cursor}
             | nombre_variable_cursor
          }
       ] }
    }
```

]
[OPTION **(** < indicio_consulta> [**,**...*n*] **)**]
< tabla_fuente> ::=
nombre_tabla [[AS] *alias_tabla*] [WITH **(** < indicio_tabla> [**,**...*n*] **)**]
| *nombre_vista*[[AS] *alias_tabla*]
| *función_filas*[[AS] *alias_tabla*]
| *tabla_derivada*[AS] *alias_tabla*[(*alias_columna*[**,**...*n*])]
| < tabla_agregada>
< tabla_agregada> ::=
< tabla_fuente> < tipo_unión> < tabla_fuente> ON < condición_búsqueda>
| < tabla_fuente> CROSS JOIN < tabla_fuente>
| < tabla_agregada>
< tipo_unión> ::=
[INNER | { { LEFT | RIGHT | FULL } [OUTER] }]
[< indicio_unión>]
JOIN
< indicio_tabla_limitado> ::=
{ FASTFIRSTROW
| HOLDLOCK
| PAGLOCK
| READCOMMITTED
| REPEATABLEREAD
| ROWLOCK
| SERIALIZABLE
| TABLOCK
| TABLOCKX
| UPDLOCK
}
< indicio_tabla> ::=
{ INDEX **(** *valor_índice*[**,**...*n*] **)**
| FASTFIRSTROW

| HOLDLOCK
| NOLOCK
| PAGLOCK
| READCOMMITTED
| READPAST
| READUNCOMMITTED
| REPEATABLEREAD
| ROWLOCK
| SERIALIZABLE
| TABLOCK
| TABLOCKX
| UPDLOCK
}
< indicio_consulta> ::=
{ { HASH | ORDER } GROUP
| { CONCAT | HASH | MERGE } UNION
| FAST *número_filas*
| FORCE ORDER
| MAXDOP
| ROBUST PLAN
| KEEP PLAN
}

3.2.2.4. Update.- sirve para cambiar datos existentes en una tabla.

Sintáxis

UPDATE
{
nombre_tabla WITH (< indicio_tabla_limitado> [...*n*])

```
    | nombre_vista
    | función_filas_limitada
    }
    SET
    { nombre_columna = { expresión | DEFAULT | NULL }
    | @variable = expresión
    | @variable = columna = expresión } [ ,...n ]

{ { [ FROM { < tabla_fuente> } [ ,...n ] ]

    [ WHERE
       < condición_búsqueda> ] }
    |
    [ WHERE CURRENT OF
    { { [ GLOBAL ] nombre_cursor} | nombre_variable_cursor}
    ] }
    [ OPTION ( < indicio_consulta> [ ,...n ] ) ]
  < tabla_fuente> ::=
nombre_tabla [ [ AS ] alias_tabla] [ WITH ( < indicio_tabla> [ ,...n ] ) ]
| nombre_vista[ [ AS ] alias_tabla]
| función_filas[ [ AS ] alias_tabla]
| tabla_derivada[ AS ] alias_tabla[ ( alias_columna[ ,...n ] ) ]
| < tabla_agregada>
  < tabla_agregada> ::=
< tabla_fuente> < tipo_unión> < tabla_fuente> ON < condición_búsqueda>
| < tabla_fuente> CROSS JOIN < tabla_fuente>
| < tabla_agregada>
  < tipo_unión> ::=
[ INNER | { { LEFT | RIGHT | FULL } [ OUTER ] } ]
[ < indicio_unión> ]
JOIN
```

< indicio_tabla_limitado> ::=
{ FASTFIRSTROW
| HOLDLOCK
| PAGLOCK
| READCOMMITTED
| REPEATABLEREAD
| ROWLOCK
| SERIALIZABLE
| TABLOCK
| TABLOCKX
| UPDLOCK
}
< indicio_tabla> ::=
{ INDEX **(** *valor_índice*[**,**...*n*] **)**
| FASTFIRSTROW
| HOLDLOCK
| NOLOCK
| PAGLOCK
| READCOMMITTED
| READPAST
| READUNCOMMITTED
| REPEATABLEREAD
| ROWLOCK
| SERIALIZABLE
| TABLOCK
| TABLOCKX
| UPDLOCK
}
< indicio_consulta> ::=
{ { HASH | ORDER } GROUP
| { CONCAT | HASH | MERGE } UNION

| {LOOP | MERGE | HASH } JOIN
| FAST *número_filas*
| FORCE ORDER
| MAXDOP
| ROBUST PLAN
| KEEP PLAN
}

3.2.2.5. Truncate table.- sirve para eliminar todas las filas de una tabla sin registrar las eliminaciones individuales de las filas. Esta instrucción no forma parte del conjunto de SQL 92 DML, sin embargo es muy útil cuando se desea eliminar filas sin dejar rastros.

Sintáxis

TRUNCATE TABLE *nombre*

Cuando se ejecutan las operaciones de manipulación de datos existe un parámetro que es la condición de búsqueda. Es una combinación de uno o más predicados usando los operadores lógicos AND, OR y NOT; para de esta manera crear condiciones de filtrado muy completas y complejas.

Sintáxis
< condición_búsqueda> ::=
{ [NOT] < predicado > | (< condición_búsqueda>) }
[{ AND | OR } [NOT] { < predicado > | (< condición_búsqueda>) }]
} [**,**...*n*]
< predicado > ::=
{ *expresión* { = | < > | ! = | > | > = | ! > | < | < = | ! < } *expresión*
| *expresión_cadena*[NOT] LIKE *expresión_cadena*
[ESCAPE '*caracter_escape*']

| *expresión* [NOT] BETWEEN *expresión* AND *expresión*
| *expresión* IS [NOT] NULL
| CONTAINS
({ *columna* | ***** } **,** '< contiene_condición_búsqueda>' **)**
| FREETEXT **(** { *columna* | ***** } **,** '*cadena_texto_libre*' **)**
| *expresión* [NOT] IN **(** *subconsulta* | *expresión* [**,**...*n*] **)**
| *expresión* { = | < > | ! = | > | > = | ! > | < | < = | ! < }
{ ALL | SOME | ANY} **(** *subconsulta* **)**
| EXISTS **(** *subconsulta* **)**
}

3.3. CREACIÓN DE ESTRUCTURAS DE ALMACENAMIENTO EN LOS SISTEMAS DE BASES DE DATOS

Para el diseño y creación de las estructuras de almacenamiento en los DBMS, se utilizará la ayuda de un programa que está dentro de la categoría de los CASE (Computer Aided Systems Engineering) de la compañía Sybase, Power Designer 9.0. Este paquete permite realizar el diagrama lógico, físico, de objetos, procesos de negocios y de capas para Internet con la generación de los respectivos scripts de código básico. Como se mencionó anteriormente en el capítulo 2, se utilizará el lenguaje SQL 92 que es el estándar de ANSI nivel 2 para garantizar una total compatibilidad con los DBMS que desee utilizar el cliente que implante el Sistema de Monitoreo.

3.3.1. Modelo Conceptual de Datos

Se realizará el modelo conceptual de datos porque éste es un estándar independiente del tipo de dato y ayuda a tener una buena abstracción de los objetos de la realidad que se desea modelar. Para esto, se crea un modelo nuevo desde el menú de archivo y se selecciona un modelo conceptual de datos, figura 3.28.

En la figura 3.29 se puede ver la paleta de herramientas para la creación y edición de modelos conceptuales de datos donde se tiene los objetos más importantes como entidades, paquetes, relaciones, herencias, asociaciones, enlaces de asociaciones, enlaces de dependencias y otras figuras cosméticas.

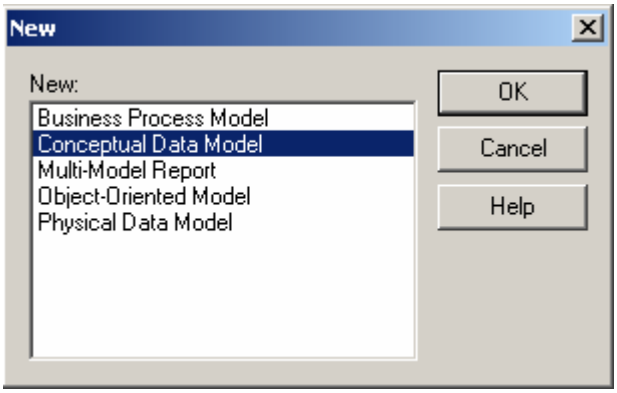

Figura 3.28 – Selección del nuevo Modelo Conceptual de Datos

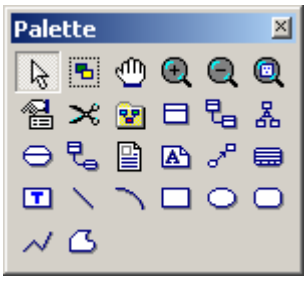

Figura 3.29 – Paleta de Herramientas para crear modelos conceptuales

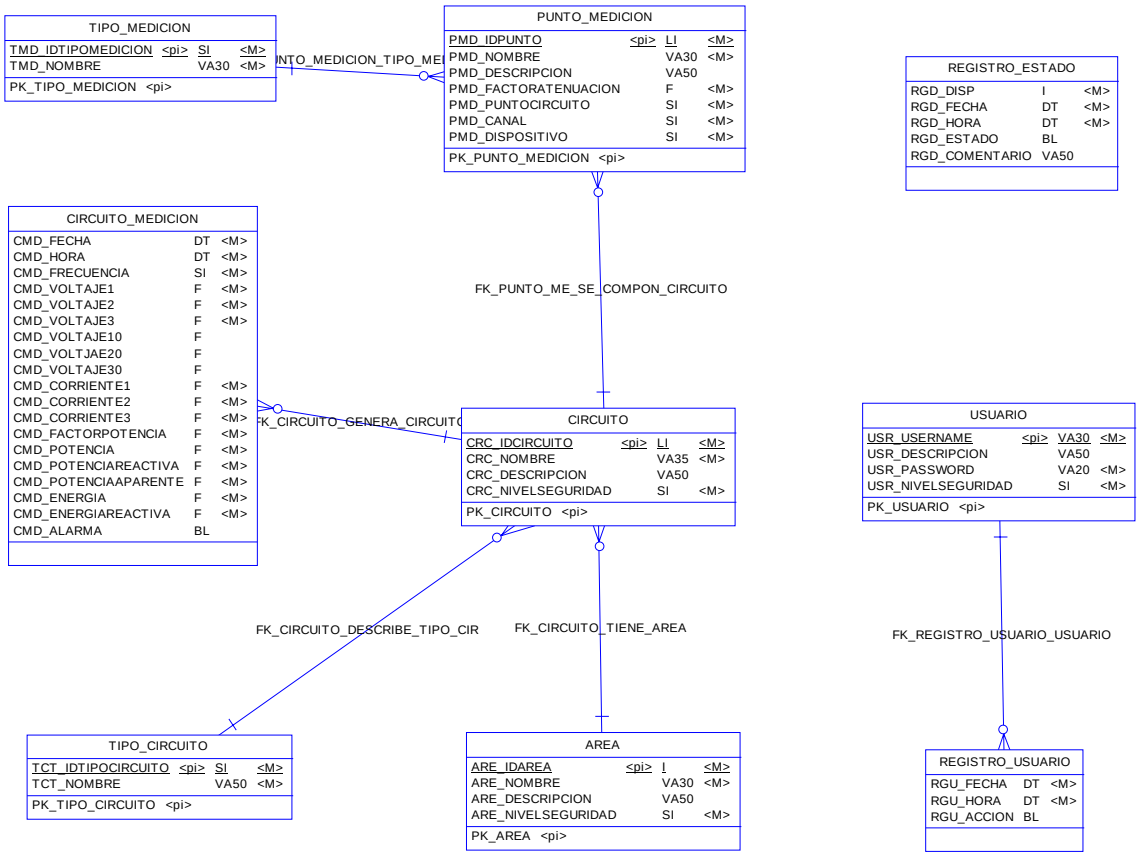

Figura 3.30 – Modelo Conceptual de Datos

En la figura 3.30 se tiene el modelo conceptual de datos del Sistema de Monitoreo de Energía Eléctrica, donde se puede ver que se han abstraído los siguientes objetos:

- Usuario (administrador, operador avanzado, operador)
- Tipo Circuito (trifásico 3 hilos, trifásico 4 hilos, trifásico 3 hilos balanceado)
- Area (bodegas, producción 1, producción 2, embalaje y almacenamiento, etc)
- Circuito (caldero 1, caldero 2, empacadoras, inyección de CO_2, lavado, etc)
- Punto Medición (parámetros de configuración de la tarjeta DAQ)
- Circuito Medición (parámetros eléctricos de las mediciones de cada circuito)
- Registro (usuario, fecha y hora de entradas y salidas del sistema)
- Tipo Medición (voltaje, corriente)

3.3.2. Modelo Físico de Datos

Una vez finalizado el modelo conceptual se genera el modelo físico de datos, se escoge la opción ‘Generar Modelo Físico de Datos’ del menú de herramientas y; aparece el cuadro de dialogo de opciones de la figura 3.31, donde se escoge el DBMS a utilizar y algunas opciones adicionales. En este punto es donde empiezan a independizarse los modelos físicos, pudiendo tener tantos archivos de modelos físicos como tipos y versiones de sistemas de bases de datos existen.

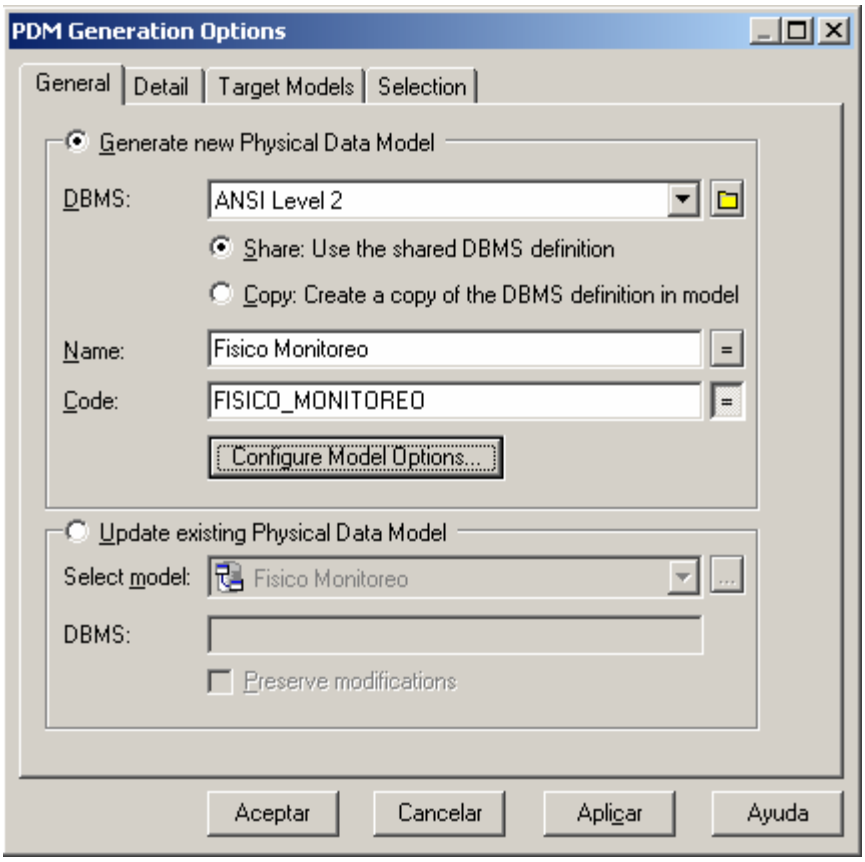

Figura 3.31 – Opciones de generación de Modelo Físico de Datos

En la figura 3.32 se puede ver la paleta de herramientas para la creación y edición de modelos físicos de datos donde se tiene los objetos más importantes como paquetes, tablas, vistas, referencias, enlaces de dependencia y otras figuras cosméticas.

Figura 3.32 – Paleta de Herramientas para crear modelos físicos

En la figura 3.33 se tiene el modelo físico de datos del Sistema de Monitoreo de Energía Eléctrica, donde se puede ver que los atributos de cada objeto ya tienen un tipo de dato y además las relaciones de llaves primarias y llaves foráneas en cada objeto.

Luego se procede a la obtención del script del sistema de base de datos, para esto se selecciona ‘Generar Base de Datos’ del menú Base de Datos. En la figura 3.34 se muestra el cuadro de dialogo de las opciones para tablas, vistas, índices, llaves y base de datos; para la creación del script y/o base de datos en forma directa mediante una conexión de ODBC. Se recomienda crear sólo el script, luego utilizar la herramienta del DBMS para realizar la ejecución del mismo debido a varias opciones que van implementando las nuevas versiones y Power Designer no las contempla.

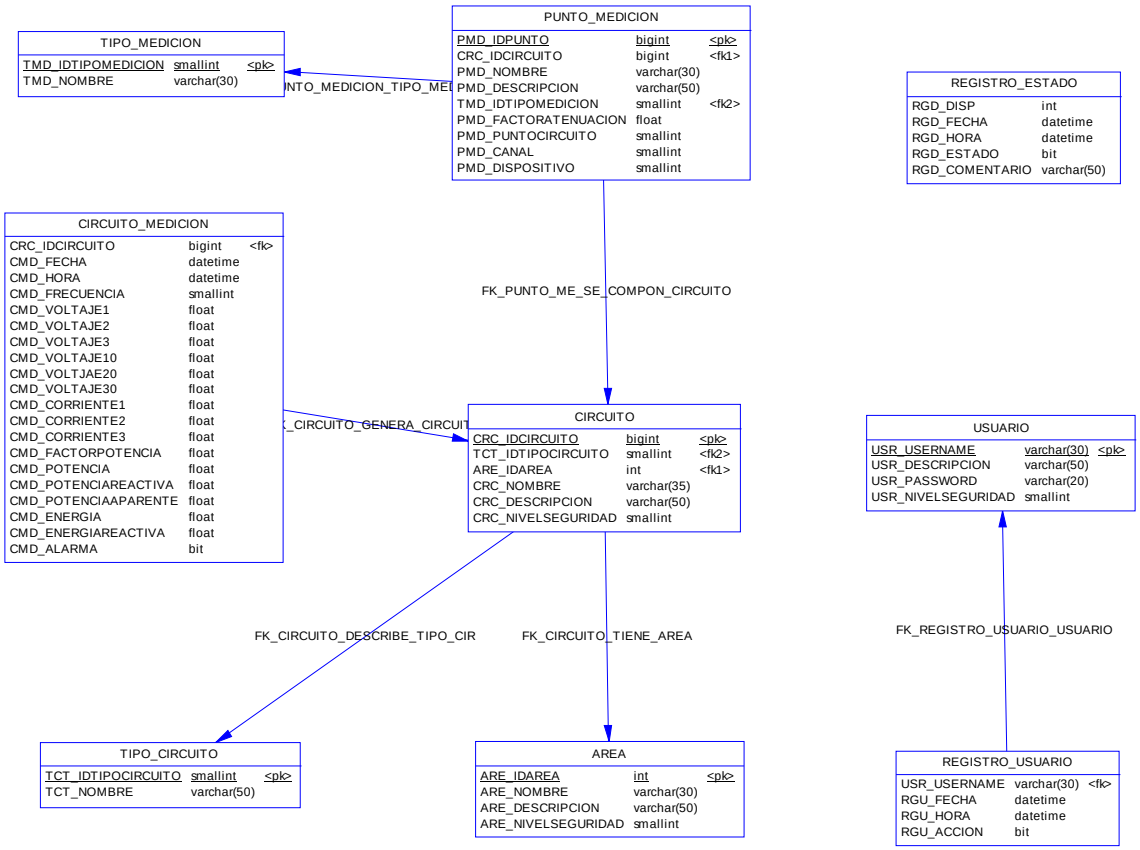

Figura 3.33 – Modelo Físico de Datos terminado

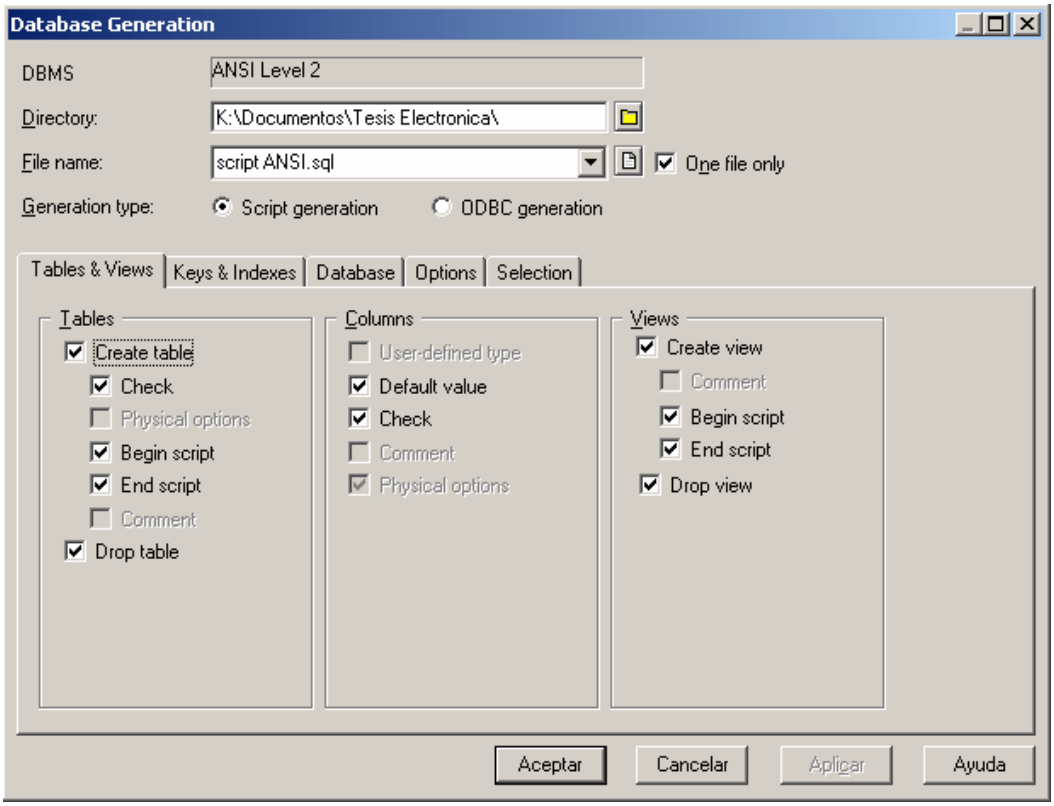

Figura 3.34 – Opciones para la creación del script y la base de datos

En la figura 3.35 se ve el resultado de las operaciones de creación del script y del DBMS si se ha utilizado esa opción. Al pulsar el botón 'Edit' de este cuadro de dialogo se obtiene el script que se mostrará a continuación.

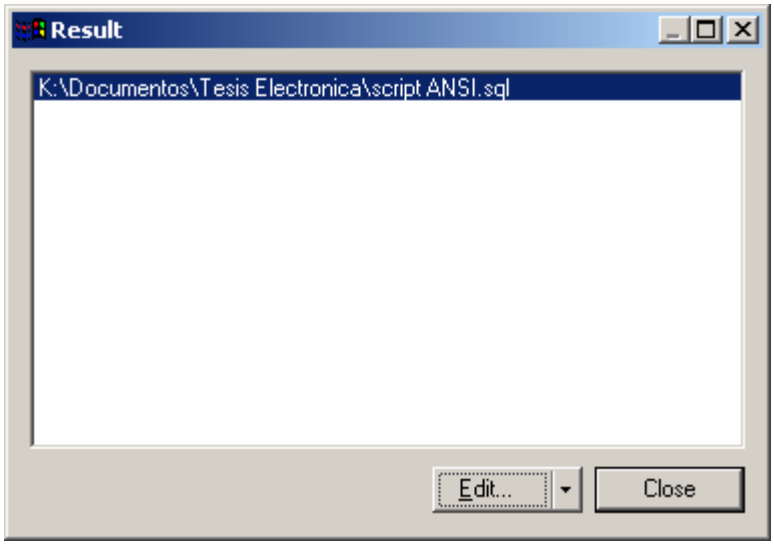

Figura 3.35 – Resultado de creación del script para el DBMS

3.3.3. Script de Tablas, Índices y Relaciones

```
--==============================================================
-- DBMS name:      ANSI Level 2
-- Created on:     08/10/2004 01:57:44 AM
--==============================================================


drop index DESCRIBE_FK;

drop index TIENE_FK;

drop index GENERA_FK;

drop index SE_COMPONE_FK;

drop table CIRCUITO_MEDICION cascade;

drop table PUNTO_MEDICION cascade;

drop table CIRCUITO cascade;

drop table REGISTRO_USUARIO cascade;

drop table USUARIO cascade;

drop table TIPO_MEDICION cascade;

drop table TIPO_CIRCUITO cascade;

drop table REGISTRO_ESTADO cascade;

drop table AREA cascade;

--==============================================================
-- Table: AREA
--==============================================================
```

```
create table AREA (
ARE_IDAREA           INTEGER              not null,
ARE_NOMBRE           VARCHAR(30)          not null,
ARE_DESCRIPCION      VARCHAR(50),
ARE_NIVELSEGURIDAD   SMALLINT             not null,
primary key (ARE_IDAREA)
);

--==============================================================
-- Table: REGISTRO_ESTADO
--==============================================================
create table REGISTRO_ESTADO (
RGD_DISP             INTEGER              not null,
RGD_FECHA            DATE                 not null,
RGD_HORA             DATE                 not null,
RGD_ESTADO           smallint,
RGD_COMENTARIO       VARCHAR(50)
);

--==============================================================
-- Table: TIPO_CIRCUITO
--==============================================================
create table TIPO_CIRCUITO (
TCT_IDTIPOCIRCUITO   SMALLINT             not null,
TCT_NOMBRE           VARCHAR(50)          not null,
primary key (TCT_IDTIPOCIRCUITO)
);

--==============================================================
-- Table: TIPO_MEDICION
--==============================================================
create table TIPO_MEDICION (
TMD_IDTIPOMEDICION   SMALLINT             not null,
TMD_NOMBRE           VARCHAR(30)          not null,
primary key (TMD_IDTIPOMEDICION)
);

--==============================================================
```

```
-- Table: USUARIO
--==============================================================
create table USUARIO (
USR_USERNAME         VARCHAR(30)          not null,
USR_DESCRIPCION      VARCHAR(50),
USR_PASSWORD         VARCHAR(20)          not null,
USR_NIVELSEGURIDAD   SMALLINT             not null,
primary key (USR_USERNAME)
);

--==============================================================
-- Table: REGISTRO_USUARIO
--==============================================================
create table REGISTRO_USUARIO (
USR_USERNAME         VARCHAR(30)          not null,
RGU_FECHA            DATE                 not null,
RGU_HORA             DATE                 not null,
RGU_ACCION           smallint,
foreign key (USR_USERNAME)
      references USUARIO (USR_USERNAME)
);

--==============================================================
-- Table: CIRCUITO
--==============================================================
create table CIRCUITO (
CRC_IDCIRCUITO       INTEGER              not null,
TCT_IDTIPOCIRCUITO   SMALLINT             not null,
ARE_IDAREA           INTEGER              not null,
CRC_NOMBRE           VARCHAR(35)          not null,
CRC_DESCRIPCION      VARCHAR(50),
CRC_NIVELSEGURIDAD   SMALLINT             not null,
primary key (CRC_IDCIRCUITO),
foreign key (ARE_IDAREA)
      references AREA (ARE_IDAREA),
foreign key (TCT_IDTIPOCIRCUITO)
      references TIPO_CIRCUITO (TCT_IDTIPOCIRCUITO)
);
```

```
--==============================================================
-- Index: DESCRIBE_FK
--==============================================================
create  index DESCRIBE_FK on CIRCUITO (
TCT_IDTIPOCIRCUITO ASC
);

--==============================================================
-- Index: TIENE_FK
--==============================================================
create  index TIENE_FK on CIRCUITO (
ARE_IDAREA ASC
);

--==============================================================
-- Table: PUNTO_MEDICION
--==============================================================
create table PUNTO_MEDICION (
PMD_IDPUNTO          INTEGER              not null,
CRC_IDCIRCUITO       INTEGER              not null,
PMD_NOMBRE           VARCHAR(30)          not null,
PMD_DESCRIPCION      VARCHAR(50),
TMD_IDTIPOMEDICION   SMALLINT             not null,
PMD_FACTORATENUACION FLOAT                not null,
PMD_PUNTOCIRCUITO    SMALLINT             not null,
PMD_CANAL            SMALLINT             not null,
PMD_DISPOSITIVO      SMALLINT             not null,
primary key (PMD_IDPUNTO),
foreign key (CRC_IDCIRCUITO)
      references CIRCUITO (CRC_IDCIRCUITO),
foreign key (TMD_IDTIPOMEDICION)
      references TIPO_MEDICION (TMD_IDTIPOMEDICION)
);

--==============================================================
-- Index: SE_COMPONE_FK
--==============================================================
```

```
create  index SE_COMPONE_FK on PUNTO_MEDICION (
CRC_IDCIRCUITO ASC
);

--==============================================================
-- Table: CIRCUITO_MEDICION
--==============================================================
create table CIRCUITO_MEDICION (
CRC_IDCIRCUITO       INTEGER              not null,
CMD_FECHA            DATE                 not null,
CMD_HORA             DATE                 not null,
CMD_FRECUENCIA       SMALLINT             not null,
CMD_VOLTAJE1         FLOAT                not null,
CMD_VOLTAJE2         FLOAT                not null,
CMD_VOLTAJE3         FLOAT                not null,
CMD_VOLTAJE10        FLOAT,
CMD_VOLTJAE20        FLOAT,
CMD_VOLTAJE30        FLOAT,
CMD_CORRIENTE1       FLOAT                not null,
CMD_CORRIENTE2       FLOAT                not null,
CMD_CORRIENTE3       FLOAT                not null,
CMD_FACTORPOTENCIA   FLOAT                not null,
CMD_POTENCIA         FLOAT                not null,
CMD_POTENCIAREACTIVA FLOAT                not null,
CMD_POTENCIAAPARENTE FLOAT                not null,
CMD_ENERGIA          FLOAT                not null,
CMD_ENERGIAREACTIVA  FLOAT                not null,
CMD_ALARMA           smallint,
foreign key (CRC_IDCIRCUITO)
      references CIRCUITO (CRC_IDCIRCUITO)
);

--==============================================================
-- Index: GENERA_FK
--==============================================================
create  index GENERA_FK on CIRCUITO_MEDICION (
CRC_IDCIRCUITO ASC
);
```

En el script se puede ver la utilización de las instrucciones SQL antes descritas y algunas adicionales como; Create Index y Drop Index que sirven para relacionar las tablas mediante sus llaves primarias.

3.3.4. Diccionario de Datos

El diccionario de datos sirve para tener una visión más amplia de la información que almacenará cada campo dentro de las tablas, las relaciones entre las mismas; además se puede conocer también el tipo de dato, su código que debe ser único para fácil ubicación, a que tabla pertenece, si es una llave primaria o foránea y si es un dato obligatorio u opcional.

Para obtener estos reportes se selecciona ‘Reportes’ del menú Modelo en cada unos de los modelos de datos, pues estos nos aportarán información diferente. A continuación se listarán algunas tablas importantes que conforman el diseño del Sistema de Monitoreo de Energía Eléctrica.

Name	Code	Parent Table	Child Table
tiene	TIENE	Area	Circuito
genera	GENERA	Circuito	Circuito Medicion
se compone	SE_COMPONE	Circuito	Punto Medicion
describe	DESCRIBE	Tipo Circuito	Circuito
Describe	DESCRIBE	Tipo Medicion	Punto Medicion
genera entradas	GENERA_ENTRADAS	Usuario	Registro

Tabla 3.1 – Lista de Referencias entre Tablas

Name	Code
Usuario	USUARIO
Punto Medicion	PUNTO_MEDICION
Area	AREA
Circuito Medicion	CIRCUITO_MEDICION
Tipo Circuito	TIPO_CIRCUITO
Circuito	CIRCUITO
Registro	REGISTRO
Tipo Medicion	TIPO_MEDICION

Tabla 3.2 –Lista de Tablas

Name	Code
ID Area	ARE_IDAREA
Nombre Area	ARE_NOMBRE
Descripcion Area	ARE_DESCRIPCION
Nivel Seguridad Area	ARE_NIVELSEGURIDAD
ID Circuito	CRC_IDCIRCUITO
ID Area	ARE_IDAREA
ID Tipo Circuito	TCT_IDTIPOCIRCUITO
Nombre Circuito	CRC_NOMBRE
Descripción Circuito	CRC_DESCRIPCION
Nivel Seguridad Circuito	CRC_NIVELSEGURIDAD
ID Circuito	CRC_IDCIRCUITO
Fecha Circuito	CMD_FECHA
Hora Circuito	CMD_HORA
Frecuencia	CMD_FRECUENCIA
Voltaje 1	CMD_VOLTAJE1
Voltaje 2	CMD_VOLTAJE2
Voltaje 3	CMD_VOLTAJE3

Voltaje 10	CMD_VOLTAJE10
Voltaje 20	CMD_VOLTJAE20
Voltaje 30	CMD_VOLTAJE30
Corriente 1	CMD_CORRIENTE1
Corriente 2	CMD_CORRIENTE2
Corriente 3	CMD_CORRIENTE3
Factor Potencia	CMD_FACTORPOTENCIA
Potencia	CMD_POTENCIA
Potencia Reactiva	CMD_POTENCIAREACTIVA
Potencia Aparente	CMD_POTENCIAAPARENTE
Energia	CMD_ENERGIA
Energia Reactiva	CMD_ENERGIAREACTIVA
Alarma Circuito	CMD_ALARMA
ID Punto Medicion	PMD_IDPUNTO
ID Circuito	CRC_IDCIRCUITO
ID Tipo Medicion	TMD_IDTIPOMEDICION
Nombre Punto	PMD_NOMBRE
Descripción Punto	PMD_DESCRIPCION
Factor Atenuación	PMD_FACTORATENUACION
Punto Circuito	PMD_PUNTOCIRCUITO
Canal	PMD_CANAL
Dispositivo	PMD_DISPOSITIVO
Username	USR_USERNAME
Fecha Registro	REG_FECHA
Hora Registro	REG_HORA
Tipo Accion	REG_ACCION
ID Tipo Circuito	TCT_IDTIPOCIRCUITO
Nombre Tipo Circuito	TCT_NOMBRE
ID Tipo Medicion	TMD_IDTIPOMEDICION
Nombre Tipo Medicion	TMD_NOMBRE

Username	USR_USERNAME
Descripción Usuario	USR_DESCRIPCION
Password	USR_PASSWORD
Nivel Seguridad Usuario	USR_NIVELSEGURIDAD

Tabla 3.3 –Lista de Columnas de las Tablas

Name	Code	Unique	Cluster	Primary	Foreign Key	Alternate Key	Table
AREA_PK	AREA_PK	TRUE	FALSE	TRUE	FALSE	FALSE	Area
CIRCUITO_PK	CIRCUITO_PK	TRUE	FALSE	TRUE	FALSE	FALSE	Circuito
TIENE_FK	TIENE_FK	FALSE	FALSE	FALSE	TRUE	FALSE	Circuito
DESCRIBE_FK	DESCRIBE_FK	FALSE	FALSE	FALSE	TRUE	FALSE	Circuito
GENERA_FK	GENERA_FK	FALSE	FALSE	FALSE	TRUE	FALSE	Circuito Medicion
PUNTO_MEDICION_PK	PUNTO_MEDICION_PK	TRUE	FALSE	TRUE	FALSE	FALSE	Punto Medicion
SE_COMPONE_FK	SE_COMPONE_FK	FALSE	FALSE	FALSE	TRUE	FALSE	Punto Medicion
CLASIFICA_FK	CLASIFICA_FK	FALSE	FALSE	FALSE	TRUE	FALSE	Punto Medicion
GENERA_INFO_FK	GENERA_INFO_FK	FALSE	FALSE	FALSE	TRUE	FALSE	Registro
TIPO_CIRCUITO_PK	TIPO_CIRCUITO_PK	TRUE	FALSE	TRUE	FALSE	FALSE	Tipo Circuito
TIPO_MEDICION_PK	TIPO_MEDICION_PK	TRUE	FALSE	TRUE	FALSE	FALSE	Tipo Circuito
USUARIO_PK	USUARIO_PK	TRUE	FALSE	TRUE	FALSE	FALSE	Usuario

Tabla 3.4 - Lista de Índices

Name	Code	Description Text	Data Type	Length
Descripcion	ARE_DESCRIPCION	Descripción del área donde se encuentran los circuitos trifásicos.	VA50	50
ID Area	ARE_IDAREA	Código único para identificar el área.	I	
Nivel Seguridad	ARE_NIVELSEGURIDAD	Valor de nivel de seguridad que posee el área.	SI	
Nombre	ARE_NOMBRE	Nombre del área donde se encuentran los circuitos.	VA30	30
Alarma	CMD_ALARMA	Indica si se debe registrar una alarma cuando este valor se salga de un rango establecido.	BL	
Corriente A	CMD_CORRIENTE1	Valor de la corriente de línea A del circuito trifásico.	F	
Corriente B	CMD_CORRIENTE2	Valor de la corriente de línea B del circuito trifásico.	F	
Corriente C	CMD_CORRIENTE3	Valor de la corriente de línea C del circuito trifásico.	F	
Energia	CMD_ENERGIA	Valor de la energía acumulada en cada circuito.	LF	
Energia Reactiva	CMD_ENERGIAREACTIVA	Valor de la energía reactiva acumulada en cada circuito.	LF	
Factor Potencia	CMD_FACTORPOTENCIA	Valor del factor de potencia del circuito medido.	F	
Fecha Medicion	CMD_FECHA	Fecha en la que ha sido realizada la medición.	D	
Frecuencia	CMD_FRECUENCIA	Frecuencia de las ondas de voltaje y corriente en el circuito medido.	SI	
Hora Medicion	CMD_HORA	Hora a la que ha sido realizada la medición.	T	
Nivel Seguridad	CMD_NIVELSEGURIDAD	Valor de nivel de seguridad que posee el circuito trifásico.	SI	

Potencia	CMD_POTENCIA	Valor de potencia en cada circuito.	LF	
Potencia Aparente	CMD_POTENCIAAPARENTE	Valor de potencia aparente en cada circuito.	LF	
Potencia Reactiva	CMD_POTENCIAREACTIVA	Valor de potencia reactiva en cada circuito.	LF	
Voltaje AB	CMD_VOLTAJE1	Valor del voltaje de línea AB del circuito trifásico.	F	
Voltaje AN	CMD_VOLTAJE10	Valor del voltaje de línea a neutro A del circuito trifásico.	F	
Voltaje BC	CMD_VOLTAJE2	Valor del voltaje de línea BC del circuito trifásico.	F	
Voltaje CA	CMD_VOLTAJE3	Valor del voltaje de línea CA del circuito trifásico.	F	
Voltaje CN	CMD_VOLTAJE30	Valor del voltaje de línea a neutro C del circuito trifásico.	F	
Voltaje BN	CMD_VOLTJAE20	Valor del voltaje de línea a neutro B del circuito trifásico.	F	
Descripcion	CRC_DESCRIPCION	Descripción del circuito trifásico de carga.	VA50	50
ID Circuito	CRC_IDCIRCUITO	Código único para identificar el circuito trifásico.	LI	
Nombre	CRC_NOMBRE	Nombre del circuito trifásico.	VA35	35
Canal	PMD_CANAL	Indica el número del canal del dispositivo de adquisición de datos que utiliza LabVIEW en los instrumentos virtuales.	SI	
Descripcion	PMD_DESCRIPCION	Descripción del punto de medición que se configura.	VA50	50
Dispositivo	PMD_DISPOSITIVO	Indica el índice del dispositivo de adquisición de datos que utiliza LabVIEW para los instrumentos virtuales.	SI	
Factor Atenuacion	PMD_FACTORATENUACION	Factor de atenuación que ha sido utilizado durante la medición de los datos.	F	

ID Punto Medicion	PMD_IDPUNTO	Código único para identificar el punto de medición que pertenece a cada circuito.	LI	
Nombre	PMD_NOMBRE	Nombre del punto de medición.	VA30	30
Punto Circuito	PMD_PUNTOCIRCUITO	Número de punto que ocupa en el circuito.	SI	
Tipo Medicion	PMD_TIPOMEDICION	Tipo de medición que realiza (voltaje, corriente)	SI	
Accion	REG_ACCION	Accion que realiza el usuario de entrada o salida del sistema.	BL	
Fecha Registro	REG_FECHA	Fecha del registro de entrada o salida del sistema.	DT	
Hora Registro	REG_HORA	Hora del registro de entrada o salida del sistema.	DT	
ID Tipo Circuito	TCT_IDTIPOCIRCUITO	Código único que identifica el tipo de circuito trifásico.	SI	
Nombre	TCT_NOMBRE	Nombre del tipo de circuito trifásico.	VA50	50
ID Tipo Medicion	TMD_IDTIPOMEDICION	Código único que identifica el tipo de medicion (voltaje, corriente).	SI	
Nombre	TMD_NOMBRE	Nombre del tipo de medicion.	VA30	30
Descripcion	USR_DESCRIPCION	Descripción del tipo de usuario.	VA50	50
Nivel Seguridad	USR_NIVELSEGURIDAD	Valor de nivel de seguridad que posee el usuario.	SI	
Password	USR_PASWWORD	Contraseña del usuario para verificar su identidad.	VA20	20
Username	USR_USERNAME	Nombre de usuario que ingresa a la aplicación.	VA30	30

Tabla 3.5 - Lista de Ítems de Datos

3.4. SERVIDOR Y CLIENTE DE DATASOCKET

DataSocket es una tecnología de programación de Internet creada por National Instruments. Simplifica el intercambio de datos en tiempo real entre diferentes aplicaciones en una computadora o entre computadoras conectadas a través de una red. Aunque una variedad de tecnologías diferentes existen en la actualidad para compartir datos entre aplicaciones incluyendo TCP/IP, DDE y otros; la mayoría de estas herramientas no están diseñadas para un intercambio de datos en tiempo real. DataSocket, sin embargo implementa una herramienta de programación fácil de usar y de alto rendimiento que está diseñada específicamente para compartir y publicar datos en tiempo real en aplicaciones de medición y automatización. Con DataSocket se puede pasar eficientemente datos sobre Internet y responder a múltiples usuarios sin la complejidad de la programación de bajo nivel de TCP. DataSocket hace que el compartir datos científicos y de ingeniería entre aplicaciones de software y entre computadoras conectadas en red tan fácil como leer y escribir datos en archivos.

La mayoría de herramientas de desarrollo populares para medición y automatización incorporan DataSocket. Los programadores que usan Visual C++, Visual Basic, LabWindows/CVI y LabVIEW usan DataSocket y, esto se está haciendo cada vez más popular como una tecnología para transmitir datos entre aplicaciones de medición y automatización. Debido a que DataSocket está construido en las tecnologías industriales estándares TCP/IP y ActiveX/COM, la adopción de la misma continuará siendo relativamente rápida.

DataSocket reduce substancialmente la cantidad de tiempo que toma para desarrollar aplicaciones que comparte datos a través de Internet. DataSocket está optimizado para brindar velocidad y para compartir datos científicos y de ingeniería. Por lo tanto, facilita el desarrollo de aplicaciones de medición distribuida y automatización que han sido históricamente difíciles de desarrollar y lentas en términos del rendimiento completo del sistema.

DataSocket puede ser agregada fácilmente a aplicaciones existentes. La tecnología consiste de un servidor de DataSocket que puede residir en cualquier computadora basada en Windows en la red y una API de DataSocket que es usada para enviar y recibir datos a través

de la red. Todas las herramientas de desarrollo de National Instruments incluyen DataSocket (LabVIEW y Measurement Studio), puede ser también descargado del sitio Web de National Instruments.

3.4.1. Servidor DataSocket

Un servidor de DataSocket es un archivo ejecutable que habilita el intercambio de datos entre múltiples lectores de DataSocket y escritores de DataSocket. El servidor de DataSocket acepta y almacena información de fuentes de datos y la transmite a otros destinos de datos. Cuando se ejecuta un servidor de DataSocket en una computadora, se hacen los datos accesibles a lectores y escritores de DataSocket en la misma computadora u otras computadoras conectadas a través de una red TCP, como el Internet.

Para ejecutar el servidor de DataSocket se sigue los pasos a continuación: Inicio, Programas, National Instruments, DataSocket, DataSocket Server**.** Cuando el servidor de DataSocket está ejecutándose, se observa el icono de DataSocket en la bandeja de sistema de Windows indicando que el servidor está levantado. Para verificar el estado se da un clic derecho en el icono de DataSocket y se selecciona ‘Show DataSocket Server’ del menú rápido.

La configuración predeterminada del servidor de DataSocket funciona para muchas aplicaciones de Internet, pero se puede necesitar el cambiar la configuración para algunas aplicaciones. Para modificar la configuración del servidor de DataSocket se debe usar el Administrador del Servidor de DataSocket. Los cambios realizados en la configuración del servidor de DataSocket utilizando el Administrador toman efecto la próxima vez que se ejecute el servidor de DataSocket, por lo tanto se debe bajar y reiniciar el servidor para que los cambios entren en funcionamiento en ese momento.

Para detener el servidor de DataSocket se siguen los pasos a continuación: Server, Shutdown DataSocket Server en la pantalla del servidor. Si el servidor de DataSocket está

oculto, dar un clic derecho en el icono del servidor de DataSocket de la bandeja de sistema de Windows y seleccionar Shutdown DataSocket Server de menú rápido.

3.4.1.1. Ítems de Datos

Un ítem de datos en un servidor de DataSocket representa una medición del mundo real. Los lectores de DataSocket pueden suscribirse los ítems de datos en un servidor de DataSocket, y los escritores de DataSocket pueden crear y publicar o actualizar ítems de datos en un servidor de DataSocket.

Existen dos tipos de ítems de datos: ítems de datos predefinidos e ítems de datos creados dinámicamente.

- Los ítems de datos predefinidos siempre existen el servidor de DataSocket, y estos siempre tienen un valor inicial asociado. Debido a que el servidor de DataSocket nunca libera los ítems de datos predefinidos, sus valores existen aún cuando ningún cliente de DataSocket está conectado a los mismos. Los ítems de datos predefinidos pueden tener grupos de acceso de lectura y escritura especiales de tal manera que se puede permitir que diferentes computadoras accedan a diferentes ítems de datos predefinidos.
- Los ítems de datos creados dinámicamente solo existen el tiempo que exista al menos un cliente de DataSocket conectado para leer o escribir el valor del item de datos. Cuando no quedan conexiones a un ítem de datos creado dinámicamente, el servidor de DataSocket libera el ítem de datos y su valor.

En forma predeterminada, solo los programas que se ejecutan en la misma computadora como el servidor de DataSocket pueden crear ítems de datos o publicar a los ítems de datos, mientras que las aplicaciones en la misma computadora o en otras computadoras pueden leer ítems de datos. Se debe usar el Administrador del Servidor de DataSocket para especificar cuales computadoras pueden crear ítems de datos, publicar a los ítems de datos y suscribirse a los ítems de datos.

3.4.1.2. Fuentes de Datos

Una fuente de datos es una localización a la cual se desea que un lector de DataSocket se conecte y recupere datos. Se especifica una localización de fuente de datos con un URL. Como los URLs que se usan en un explorador de Web, el localizador de fuentes de datos puede apuntar hacia muchos tipos diferentes de fuentes dependiendo del prefijo. El prefijo es llamado el esquema URL. DataSocket soporta varios esquemas:

- dstp: (Protocolo de Transferencia de DataSocket)
- http: (Protocolo de Transferencia de Hipertexto)
- ftp: (Protocolo de Transferencia de Archivo)
- opc: (OLE para Control de Procesos)
- fieldpoint:, logos:, Lookout:, (Protocolo de Comunicación para Módulos FieldPoint National Instruments, Procesos del Módulo DSC de LabVIEW, Procesos de National Instruments Lookout)
- file: (Servidores Locales de Archivos)

3.4.1.3. Destinos de Datos

Un destino de datos es una localización a la cual se desea que un escritor de DataSocket se conecte y escriba datos. Se especifica un destino de datos con un URL. Como los URLs que se usan en un explorador de Web, el localizador de destinos de datos puede apuntar hacia muchos tipos diferentes de fuentes dependiendo del prefijo. El prefijo es llamado el esquema URL. DataSocket soporta varios esquemas:

- dstp: (Protocolo de Transferencia de DataSocket)
- ftp: (Protocolo de Transferencia de Archivo)
- opc: (OLE para Control de Procesos)

- fieldpoint:, logos:, Lookout:, (Protocolo de Comunicación para Módulos FieldPoint National Instruments, Procesos del Módulo DSC de LabVIEW, Procesos de National Instruments Lookout)
- file: (Servidores Locales de Archivos)

3.4.2. Protocolos de DataSocket

DataSocket es una tecnología que simplifica el intercambio de datos entre una aplicación y otras aplicaciones, archivos, servidores FTP y servidores Web. Este provee una API común para un número de diferentes de protocolos de comunicación. DSTP es un protocolo de la capa de aplicación para la transferencia de datos de mediciones desde y hacia un servidor DSTP llamado un servidor de DataSocket. DSTP está implementado sobre TCP, y por lo tanto provee comunicación orientada a la conexión entre el servidor y el cliente. En otras palabras, los clientes mantienen una sesión durante la comunicación con el servidor. Durante la sesión el servidor mantiene rastro de la información de la conexión del cliente. Los clientes que proveen los datos de mediciones a un servidor de DataSocket son referidos como publicadores o escritores. Los clientes que consumen los datos de mediciones proveídos por los publicadores son referidos como subscriptores o lectores. En la figura 3.36 se puede ver el ejemplo de un publicador LabVIEW y dos subscriptores también en LabVIEW, sin embargo esto implicaría costos de licenciamiento.

Figura 3.36 – Ejemplo de Transmisión de Datos mediante DataSocket con LabVIEW

DataSocket, una tecnología y un grupo de herramientas, facilitan el intercambio de datos e información entre una aplicación y un número de diferentes fuentes de datos y destinos de datos. Esas fuentes y destinos incluyen archivos, servidores HTTP/FTP, servidores de OLE para Control de Procesos y servidores de DataSocket de National Instruments para publicar datos en tiempo real entre aplicaciones. A menudo, esas fuentes y destinos están localizados en una computadora diferente. Se puede especificar fuentes y destinos de DataSocket (conexiones) usando URLs (localizadores de recursos uniformes) que se adhieran al familiar modelo URL.

DataSocket usa un formato de datos mejorado para intercambiar datos de mediciones, así como también los atributos de los datos. Los atributos de los datos pueden incluir información tal como una tasa de adquisición, nombre del operador de pruebas, estampa de tiempo y calidad de los datos.

Aunque se puede usar funciones de entrada/salida de archivos de propósito general, funciones TCP/IP y peticiones FTP/HTTP para transferir datos entre diferentes aplicaciones, aplicaciones y archivos, y diferentes computadoras se debe escribir una cantidad significante de código de programa para lograr esto. DataSocket simplifica enormemente esta tarea proveyendo una API unificada para estos protocolos de comunicaciones de bajo nivel. La transferencia de datos entre computadoras con DataSocket es tan simple como usar un explorador para leer páginas Web en el Internet.

Un sistema que está participando en un intercambio de datos DSTP usualmente consiste de tres componentes – el servidor de DataSocket, un publicador y subscriptores. Un publicador adquiere datos desde un dispositivo de adquisición de datos local o remoto y los envía al servidor. El servidor debería estar localizado en la misma computadora (conocido como el localhost) o remotamente en el Internet. Los subscriptores quienes tienen un interés en los datos publicados pueden subscribirse para recibir los datos desde el servidor. Aplicaciones complejas pueden requerir más de un publicador o más de un servidor.

Tal como está implementado actualmente, por defecto DSTP transfiere los datos más recientes en el servidor a los subscriptores de los datos de mediciones. En otras palabras, los subscriptores no están garantizados de recibir cada instancia de los datos que han sido publicados en el servidor. Si el publicador está escribiendo datos al servidor a una tasa más alta que el servidor está enviando datos a sus subscriptores, los subscriptores podrían no obtener los datos publicados previamente. Esto podría parecer ser un defecto de DSTP; sin embargo, el valor más reciente de los datos de medición es usualmente el de más interés para la mayoría de las aplicaciones de medición. El principio de trabajo es exactamente el mismo como un voltímetro digital (DVM). El DVM muestra el valor de la diferencia de voltaje entre dos puntos en ese instante. Transmitiendo los datos más recientes, se puede usar DSTP en aplicaciones de medición de alto rendimiento. Sin embargo, la pérdida de datos puede minimizarse o aún eliminarse mediante almacenamiento temporal de datos en el cliente y/o el servidor. DSTP ahora ofrece ambos, almacenamiento temporal de datos del lado del cliente y del lado del servidor. El almacenamiento temporal del lado del cliente puede ser configurado a través de opciones de URL, mientras que el almacenamiento temporal del lado del servidor puede ser configurado a través del cuadro de dialogo de Diagnósticos del Servidor de DataSocket en el servidor.

Típicamente, los usuarios interactuarán con DSTP sólo usando la Interfase de Programación de Aplicaciones DataSocket (API) dentro de una aplicación de software de National Instruments. Para establecer una conexión a un servidor de DataSocket usando la API de DataSocket, el usuario solo escribe un valor a un URL. Detrás de las escenas, el cliente de DataSocket va a través de varios pasos para establecer una conexión DSTP. Primero, para establecer una sesión al servidor, el cliente envía un mensaje de ‘petición de entrada al sistema’ al servidor. La petición también incluye el número de versión del DSTP para la sesión. Segundo, si el servidor acepta la conexión, este envía de regreso una señal de realización de la conexión que confirma el número de versión del DSTP. Tercero, el cliente a su vez envía una petición para conectarse a un URL particular en el servidor. Los URLs que representan un dato de medición particular en el servidor de DataSocket son referidos como ítems de DataSocket. El servidor de DataSocket usa la misma conexión TCP para todos los

ítems de DataSocket que existen dentro del mismo espacio de proceso. En la figura 3.37 se puede visualizar este proceso de conexión del cliente.

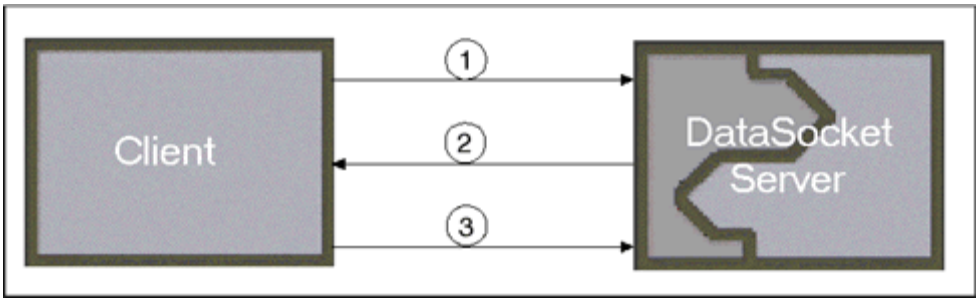

Figura 3.37 – Forma de Establecer una Conexión con el Servidor de DataSocket

El Protocolo de Transferencia de DataSocket (DSTP) está basado en TCP/IP, a través del cual los datos son pasados entre los clientes DataSocket – conocidas como las aplicaciones del lector y escritor de DataSocket – usando el servidor de DataSocket.

Un localizador uniforme de recursos identifica la localización de los datos de medición en un servidor. Un URL DSTP indica el nombre del servidor de DataSocket y la ruta de acceso para un item de datos particular. Publicadores y subscriptores para el dato de medición deben referirse al item de datos por el mismo URL.

Se puede conectar a un servidor de DataSocket desde aplicaciones de lector y escritor de DataSocket usando el esquema URL DSTP, como se demuestra en los siguientes URLs de ejemplo.

Este URL se conecta al item de datos llamado ‘wave’ en el servidor de DataSocket ejecutándose en la computadora local:

dstp://localhost/wave

Este URL se conecta al item de datos llamado ‘wave’ en un servidor de DataSocket ejecutándose en una computadora de red llamada ‘lab’:

dstp://lab/wave

Para transferir datos al servidor, un cliente envía un mensaje cuya cabecera indica al servidor para escribir los datos encapsulados en el mensaje como el nuevo valor para los datos. El mensaje también incluye el URL identificando los datos en el servidor. Para pedir datos del servidor, un cliente envía un mensaje que pide el valor más reciente de los datos. El mensaje también incluye el URL de los datos pedidos del servidor. Los clientes pueden recibir datos del servidor vía peticiones explícitas de valores o vía una conexión de actualización automática programada. El servidor envía las actualizaciones de datos a los subscriptores programados para actualizar automáticamente tan pronto como el servidor recibe valores nuevos del publicador. En otras palabras, el servidor envía actualizaciones de datos mediante difusiones múltiples para un URL particular a sus subscriptores.

3.5. APLICACIÓN DEL SERVIDOR

3.5.1. Esquema General

Para visualizar de una forma clara y adecuada los procesos que realizará el servidor en el sistema de monitoreo, se realizará un diagrama de bloques de la aplicación mostrando el flujo de los datos desde su medición hasta la visualización, que se muestran en las figuras 3.38 y 3.39.

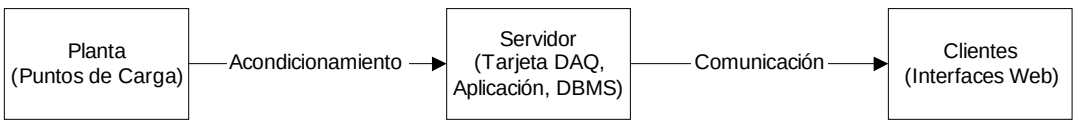

Figura 3.38 – Diagrama de Bloques General del Sistema de Monitoreo

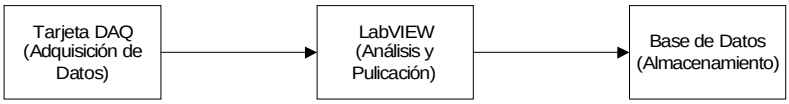

Figura 3.39 – Diagrama de Bloques del Sistema en el Servidor

En la figura 3.40 se muestra un diagrama de flujo de datos que indica en detalle el funcionamiento de la aplicación del servidor cuando se ingresa por primera vez al sistema y se

realiza la configuración de conexión a la DBMS. En la figura 3.41 se muestra el diagrama de flujo de datos que indica en detalle el funcionamiento de la aplicación del servidor cuando se ingresa al sistema desde la segunda vez en adelante. Luego la aplicación continúa normalmente en forma cíclica para realizar la adquisición, análisis, publicación y almacenamiento de los datos.

Como pantalla de control principal se tiene un instrumento virtual que se llama ‘Monitoreo’ y es el que se encarga de la coordinación de toda la aplicación mediante programación lógica y llamadas a los otros VI’s. En las figuras 3.40 y 3.41 se ha explicado de forma lógica el flujo de datos y funcionamiento de la aplicación.

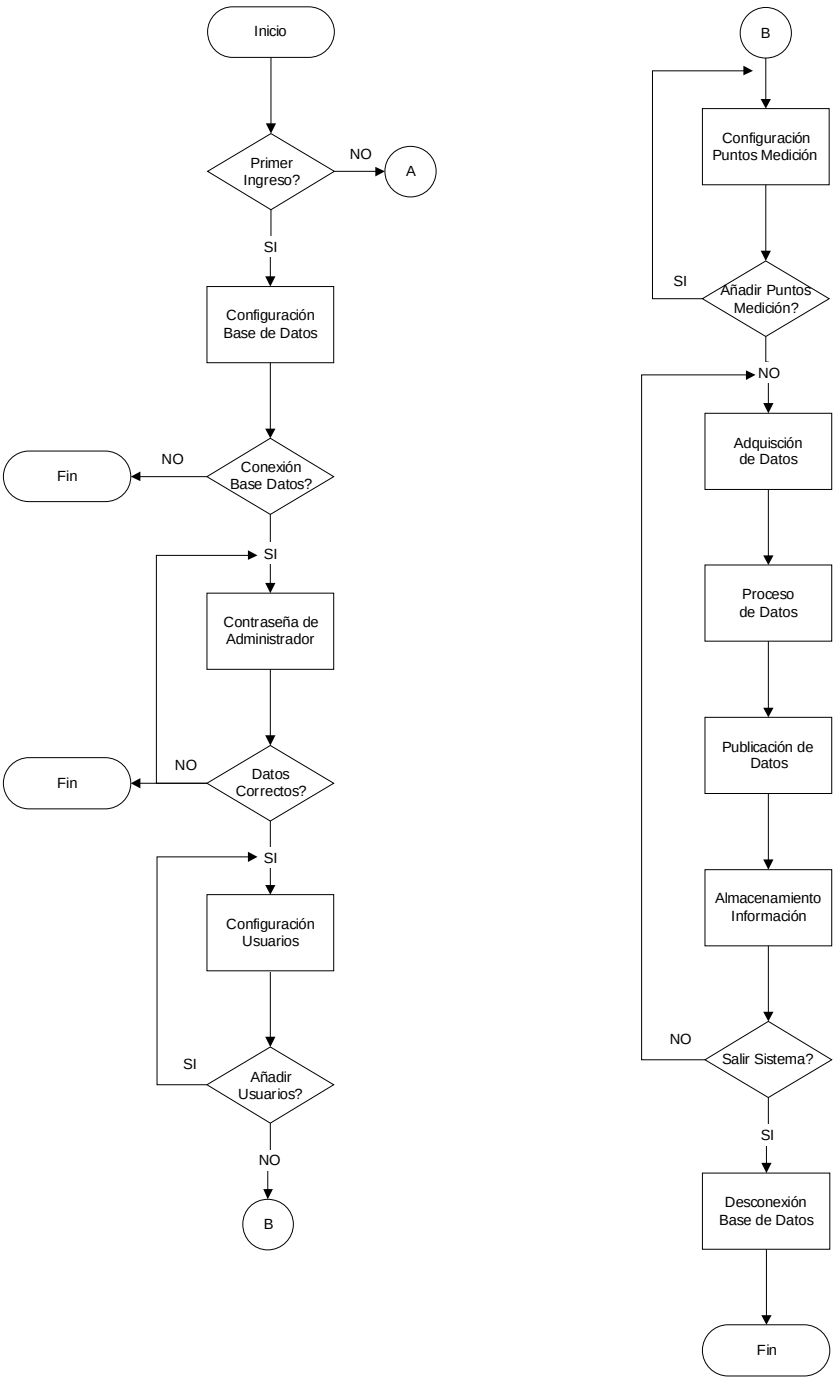

Figura 3.40 – Diagrama de Flujo de la Aplicación del Servidor (Primer Ingreso)

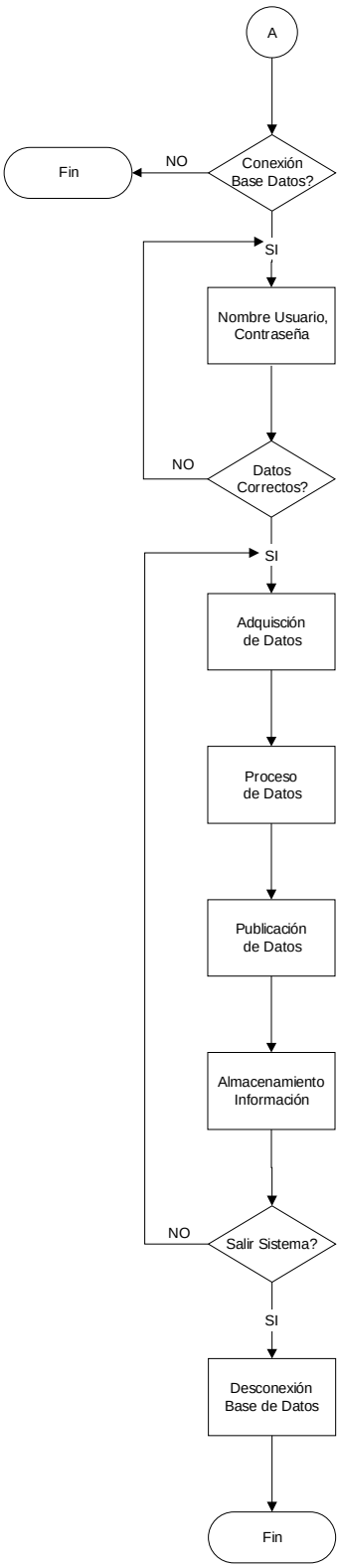

Figura 3.41 – Diagrama de Flujo de la Aplicación del Servidor (Segundo y Posterior Ingresos)

3.5.2. Programa Principal de Configuración y Seguridad

Los diagramas de flujo de las figuras 3.40 y 3.41 se traducen a un diagrama de programación en LabVIEW que se muestra en la figura 3.42.

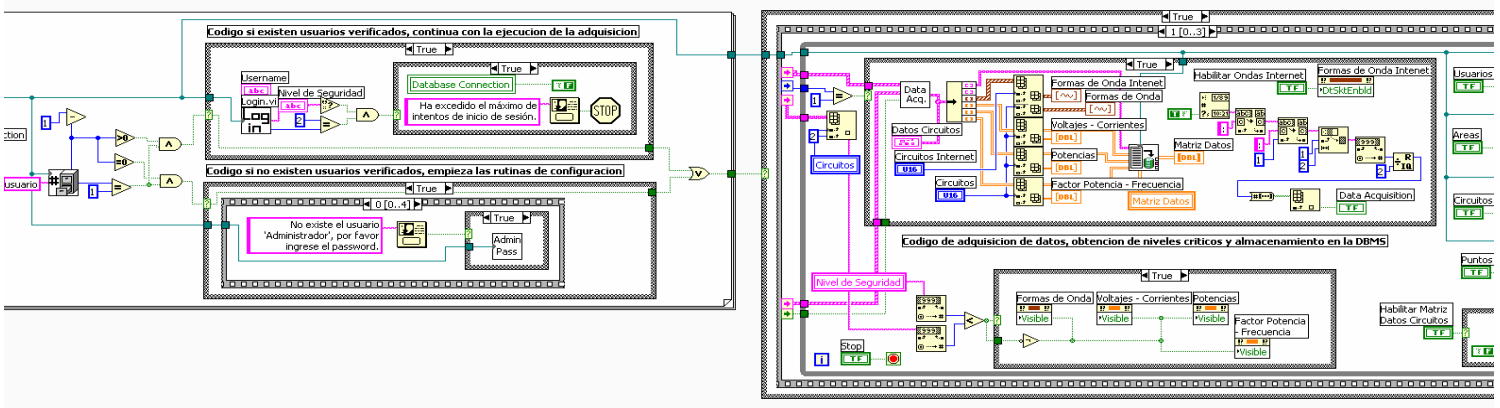

Figura 3.42 – Programación del VI Principal de la Aplicación de Monitoreo

Primero se realiza la conexión a la DBMS porque de otro modo la aplicación no puede verificar los niveles de permisos de los usuarios ni almacenar la información de los puntos de monitoreo, este VI no tiene entradas y; como salidas tiene un número de referencia de automatización ActiveX válido que posee la conexión a la DBMS, el estado de la conexión y el código de error para verificar si se ha realizado correctamente la misma. En la figura 3.43 se puede ver el VI con sus salidas.

Figura 3.43 – VI de Conexión a la DBMS

Al iniciar por primera vez la aplicación se deben configurar los parámetros del controlador o proveedor de conexión, nombre del servidor, nombre de la instancia y/o nombre de la base de datos; en la figura 3.44 se visualiza la pantalla frontal de ingreso de los parámetros de configuración donde se elige primero el método de acceso a la DBMS, es decir, mediante controladores de ODBC o proveedores de OLEDB.

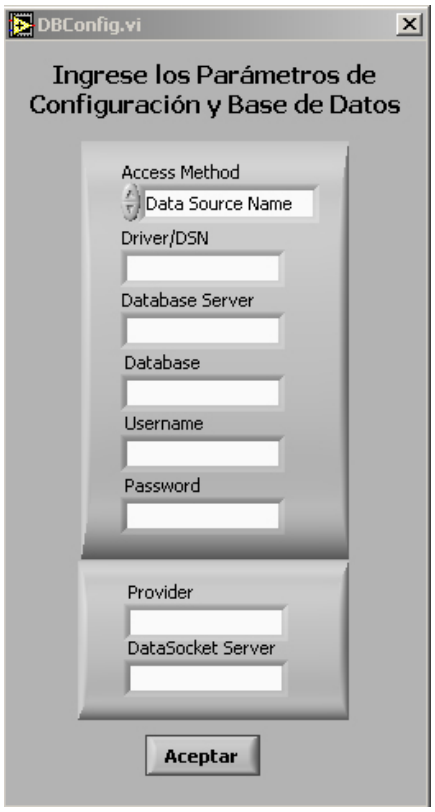

Figura 3.44 – Pantalla para Configuración de los Parámetros de Conexión a la DBMS

3.5.2.1. Funciones de Manipulación del Registro de Windows

Desde la figura 3.45 se mostrará la programación que se ha realizado para leer la información de los parámetros de configuración y conectarse a la DBMS. Aquí se realiza una lectura del registro de Windows para buscar la llave 'Sistema Monitoreo' que está en el grupo 'SOFTWARE'; el instrumento que realiza la lectura devuelve un 0 si no existieron errores y todos los parámetros han sido recuperados correctamente, caso contrario devuelve el número del error junto a su explicación. En la figura 3.46 se puede ver que se maneja el error -603 que implica la inexistencia de la llave en el registro de Windows, aquí se encuentra el código que realiza la petición de parámetros al usuario y luego los almacena en el registro para ejecuciones posteriores de la aplicación.

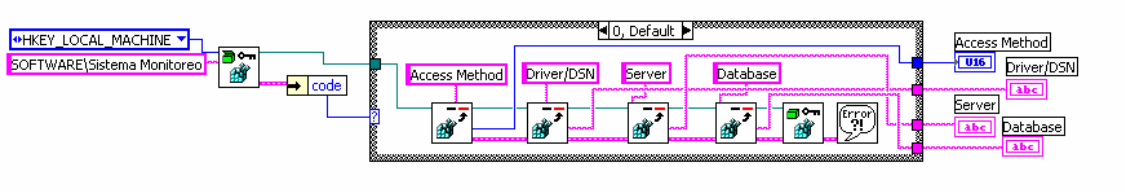

Figura 3.45 – Lectura del Registro de ActiveX para Obtener los Parámetros de Conexión

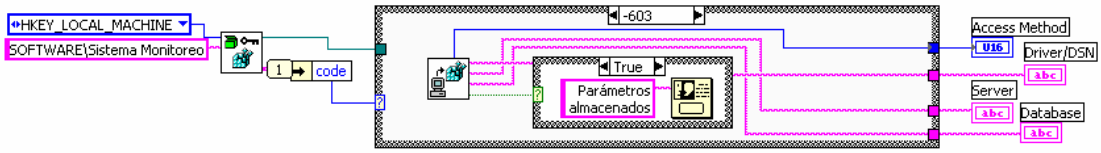

Figura 3.46 – Lectura del Registro de ActiveX, Manejo del Error de Inexistencia

En la figura 3.47 se puede ver una rutina creada para que la pantalla de petición de parámetros se comporte como orientada a eventos, es decir, esperará el tiempo que el usuario necesite para llenar los valores de los parámetros y pulse el botón de Aceptar, en este momento el código se encargará de verificar que no sean valores nulos para garantizar en algo de la validez en la configuración. En la figura 3.48 se puede ver el código que realiza la escritura de los valores de los parámetros en el registro de Windows y verifica la misma, de otra manera terminará la aplicación.

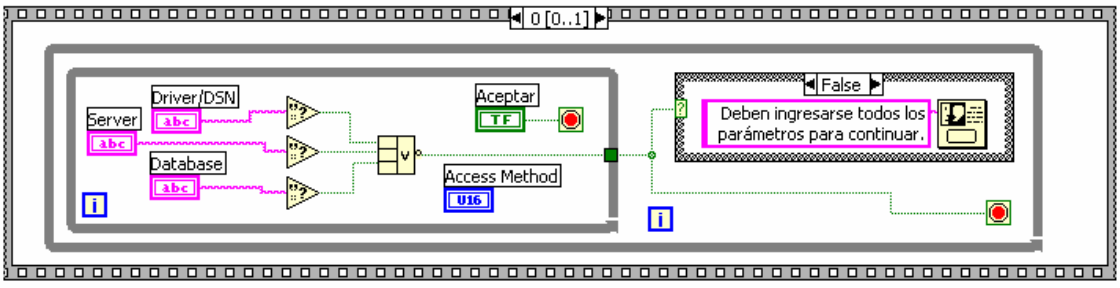

Figura 3.47 – Petición y Verificación de Parámetros de Configuración al Usuario

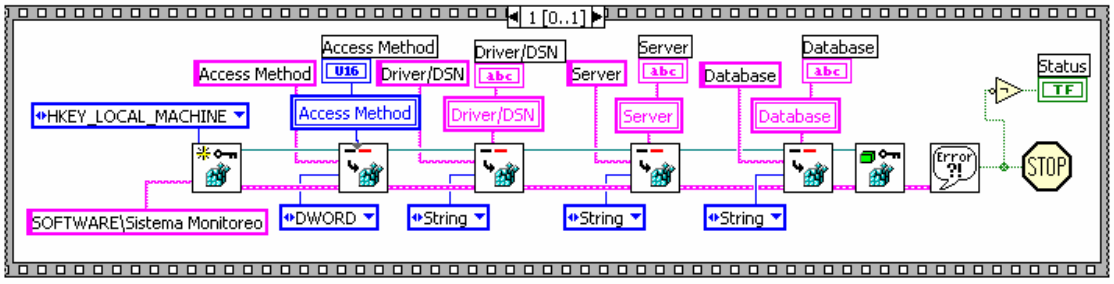

Figura 3.48 – Escritura y Verificación de Parámetros en el Registro de Windows

3.5.2.2. Código de Conexión al Sistema de Bases de Datos

En la figura 3.49 se muestra el código para la obtención de la cadena de conexión utilizando los parámetros del registro de Windows, luego se utiliza un objeto de automatización de ActiveX y se selecciona la clase ‘ADODB Connection’. Se utiliza el método ‘Open’ con algunos parámetros como la cadena de conexión y finalmente se utiliza la propiedad ‘State’ para verificar si la conexión a la DBMS está abierta. Si por alguna razón la conexión a la DBMS ha fallado se emitirá un mensaje indicando el error y terminará la aplicación por falta de conexión.

Si existe una conexión exitosa con la DBMS la aplicación continua su ejecución y se pedirá el ingreso de una contraseña para el administrador como se muestra en la figura 3.50. En la figura 3.51 se muestra el cuadro de dialogo donde se debe realizar el ingreso del password del usuario ‘Administrador’, que no puede ser blanco y por lo menos debe contener 6 caracteres. En la figura 3.52 se visualiza el mensaje que emite la aplicación indicando que el usuario ‘Administrador’ se ha insertado correctamente y se continuará el proceso de configuración de la misma.

Para realizar las operaciones de búsqueda e inserción del usuario ‘Administrador’ se utiliza el lenguaje SQL y las mismas están implementadas en el LabVIEW mediante instrumentos virtuales personalizados. Se han desarrollado las funciones básicas para manipulación de datos, que serán descritas a continuación; además se han realizado algunas

modificaciones para obtener una mejor funcionalidad y rendimiento de acuerdo a las necesidades de la aplicación.

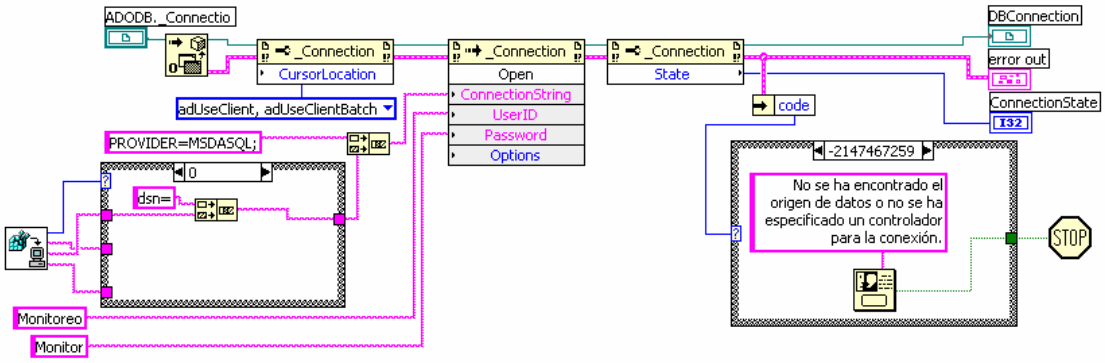

Figura 3.49 – Obtención de Cadena de Conexión, Conexión a la DBMS

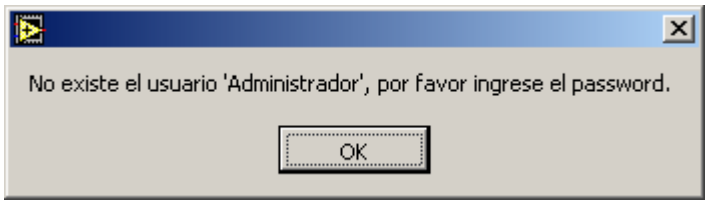

Figura 3.50 – Mensaje que indica que el usuario 'Administrador' no existe e ingrese el Password

Figura 3.51 – Cuadro de Dialogo de Ingreso y Confirmación del Password del Administrador

Figura 3.52 – Mensaje que indica la inserción correcta del usuario 'Administrador'

3.5.2.3. Funciones de Manipulación de Datos usando Structured Query Language

3.5.2.3.1. Conteo de Registros

La función de conteo de registros en una tabla o conjunto de registros es muy sencilla pero muy útil. En la figura 3.53 se indica el icono con las entradas del VI de conteo de registros que son el número de referencia ActiveX de la conexión activa, la tabla y las condiciones de filtrado y; las salidas que son el número de registros, el estado de la conexión y el error para verificar un conteo apropiado. Luego en la figura 3.54 se muestra la programación de esta función.

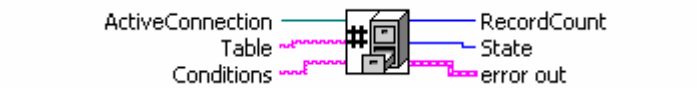

Figura 3.53 – VI para Conteo de Registros de una Tabla o Conjunto de Registros

Para el conteo de registros se construye la sentencia de SQL mediante Select y los parámetros ingresados por la aplicación; luego se utiliza el método Open de la clase ADODB Recordset. Los parámetros de este método son una referencia de automatización de ActiveX de ADODB Recordset, la sentencia de SQL, la referencia de la conexión activa de la DBMS y los tipos de cursores del recordset. Luego se utiliza las propiedades de conteo de registros y estado del recordset para verificar que cuando devuelva 0 en el contador no sea porque no se pudo abrir el objeto del recordset por algún error. Finalmente obtenidos los valores se cierra el recordset y el objeto de automatización.

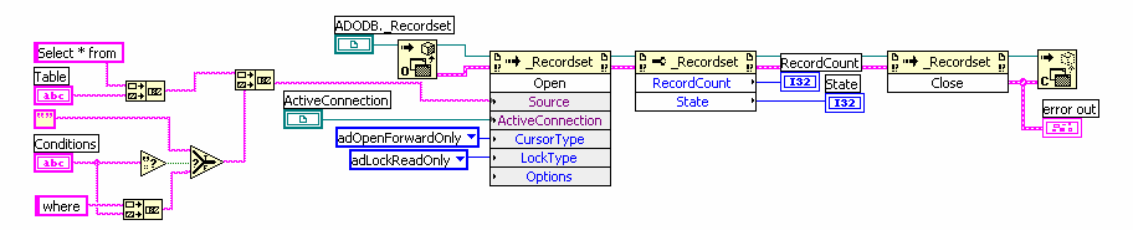

Figura 3.54 – Código del VI para Conteo de Registros

3.5.2.3.2. Selección de Registros y Columnas

Uno de los VI más importantes es el de selección de campos y registros de una tabla o conjunto de tablas combinadas. En la figura 3.55 se puede ver el icono del VI de selección de registros con sus entradas y salidas. Ya que éste es un VI complejo tiene varias entradas que permiten construir una sentencia SQL de selección muy completa y de la misma manera tiene varias salidas que indican el número de campos y registros, el estado del recordset, los nombres de los campos y por supuesto los datos recuperados en una matriz bidimensional de cadenas de caracteres.

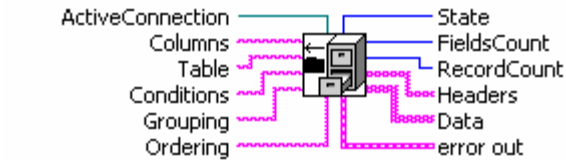

Figura 3.55 – VI para Selección de Columnas y Registros

En la figura 3.56 se puede ver la programación del VI de selección de registros. Primero se construye la sentencia de SQL de selección mediante los parámetros ingresados por la aplicación, luego se utiliza el método Open de la clase ADODB Recordset. Los parámetros de este método son una referencia de automatización de ActiveX de ADODB Recordset, la sentencia de SQL, la referencia de la conexión activa de la DBMS y los tipos de cursores del recordset. Luego se utiliza las propiedades de conteo de registros y estado del recordset para verificar que cuando devuelva 0 en el contador no sea porque no se pudo abrir el objeto del recordset por algún error.

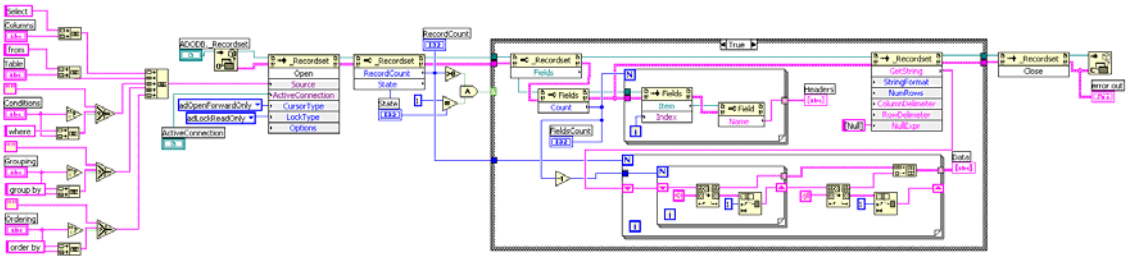

Figura 3.56 – Código del VI para Selección de Columnas y Registros

Verificadas las condiciones de existencia de registros mayor que 0 y la apertura exitosa del recordset, se procede a recuperar los nombres de los campos en una matriz unidimensional y los datos en una matriz bidimensional de cadenas de caracteres mediante un doble lazo For. Finalmente obtenidos los valores se cierra el recordset y el objeto de automatización.

3.5.2.3.3. Inserción de Registros

Para realizar la inserción de datos se ha implementado un VI que utiliza la instrucción 'Insert' de SQL y es también uno de los que más se utilizará, pues este realizará el almacenamiento de datos de los puntos monitoreados. En la figura 3.57 se puede ver el icono del VI de inserción con sus entradas y salidas. Como entradas se tiene la tabla, los campos donde se inserta y sus valores; como salidas se tiene el número de registros insertados y el código de error para verificar una correcta inserción.

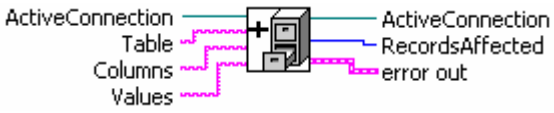

Figura 3.57 – VI para Inserción de Registros

En la figura 3.58 se muestra la programación del VI para inserción de registros. Primero se construye la sentencia de inserción de SQL con los parámetros enviados por la aplicación. Luego se utiliza un método de la clase ADODB Connection que tiene como único parámetro la sentencia de SQL y devuelve como resultado un objeto recordset que no se usa porque solo

se necesita saber la cantidad de registros que fueron insertados, valor que también devuelve como resultado del método.

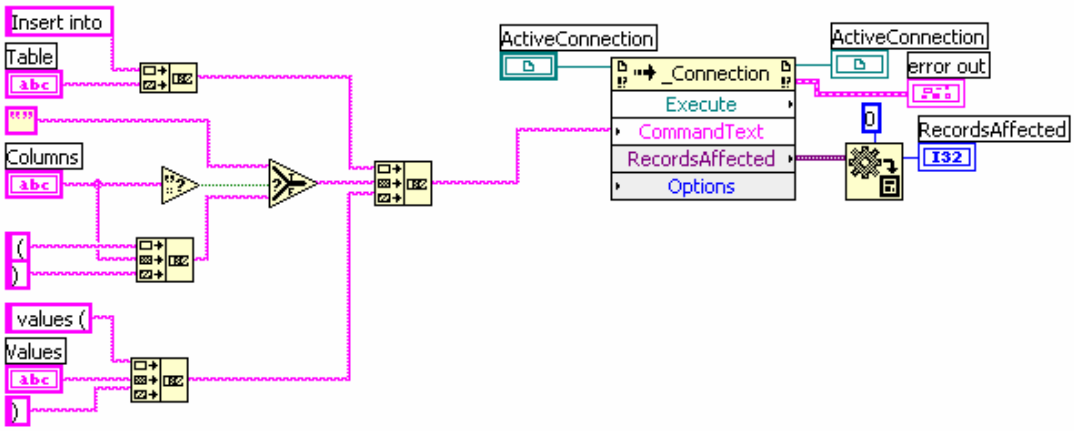

Figura 3.58 – Código del VI para Inserción de Registros

3.5.2.3.4. Inserción de Registros en Matriz

Debido a que los datos que se obtienen del monitoreo de los diversos circuitos en una planta son masivos y LabVIEW los adquiere en forma de matriz, se ha creado un VI que hace uso del VI de inserción de registros anterior para que se más fácil de usarlo dentro de la aplicación. En la figura 3.59 se puede ver el icono del VI que se llama 'Insert Matrix' con sus entradas y salidas.

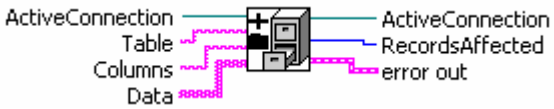

Figura 3.59 – VI para Inserción de Matriz de Registros

Como entradas se tiene la tabla, los campos en donde se insertarán los datos y la matriz de datos en forma de cadenas de caracteres; como salidas se tiene el número de registros insertados correctamente y el código de error para verificar la adecuada inserción de los datos.

En la figura 3.60 se muestra la programación del VI de inserción de matriz de registros. Primero se realiza una separación de la matriz en datos independientes, luego se utiliza un

doble lazo For con límites del tamaño de la matriz para crear cadenas de caracteres separadas por comas y cada elemento entre comillas simples para entregar al VI de inserción de registros. Al terminar el lazo For se tiene el número de registros insertados.

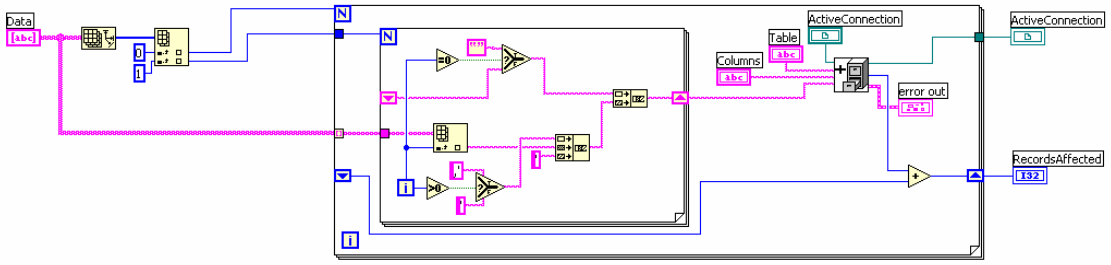

Figura 3.60 – Código del VI para Inserción de Matriz de Registros

3.5.2.3.5. Eliminación de Registros

En la configuración de usuarios y puntos de medición se puede necesitar eliminar algunos registros, por lo tanto se ha implementado un VI que realiza eliminación de registros. En la figura 3.61 se puede ver el VI de eliminación de registros con sus entradas y salidas. Como entradas tiene la tabla y las condiciones de eliminación de registros; como salidas tiene el número de registros eliminados y el código de error para verificar la correcta eliminación de los registros.

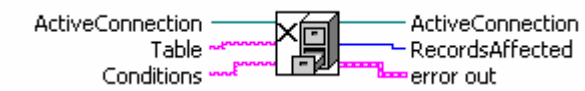

Figura 3.61 – VI para Eliminación de Registros

En la figura 3.62 se muestra la programación del VI para eliminación de registros. Primero se construye la sentencia de SQL para eliminación de registros con los parámetros enviados por la aplicación. Luego se utiliza un método de la clase ADODB Connection que tiene como único parámetro la sentencia de SQL y devuelve como resultado un objeto recordset que no se usa porque solo se necesita saber la cantidad de registros que fueron eliminados, valor que también devuelve como resultado del método.

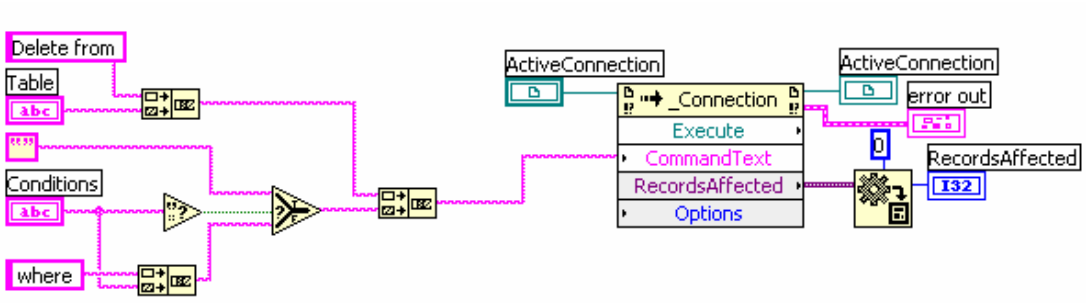

Figura 3.62 – Código del VI para Eliminación de Registros

3.5.2.3.6. Actualización de Registros

Asimismo, se puede tener la necesidad de realizar una actualización de los datos almacenados en la DBMS sobre los usuarios y la configuración de puntos de medición, por lo tanto se ha implementado un VI que realiza actualización de registros. En la figura 3.63 se puede ver el VI de actualización de registros con sus entradas y salidas. Como entradas tiene la tabla, los nuevos valores y las condiciones de actualización de registros; como salidas tiene el número de registros actualizados y el código error para verificar la correcta actualización de los registros.

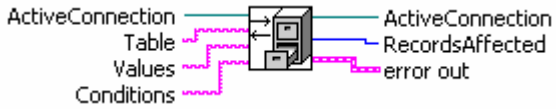

Figura 3.63 – VI para Actualización de Registros

En la figura 3.64 se muestra la programación del VI para actualización de registros. Primero se construye la sentencia de SQL para actualización de registros con los parámetros enviados por la aplicación. Luego se utiliza un método de la clase ADODB Connection que tiene como único parámetro la sentencia de SQL y devuelve como resultado un objeto recordset que no se usa porque solo se necesita saber la cantidad de registros que fueron actualizados, valor que también devuelve como resultado del método.

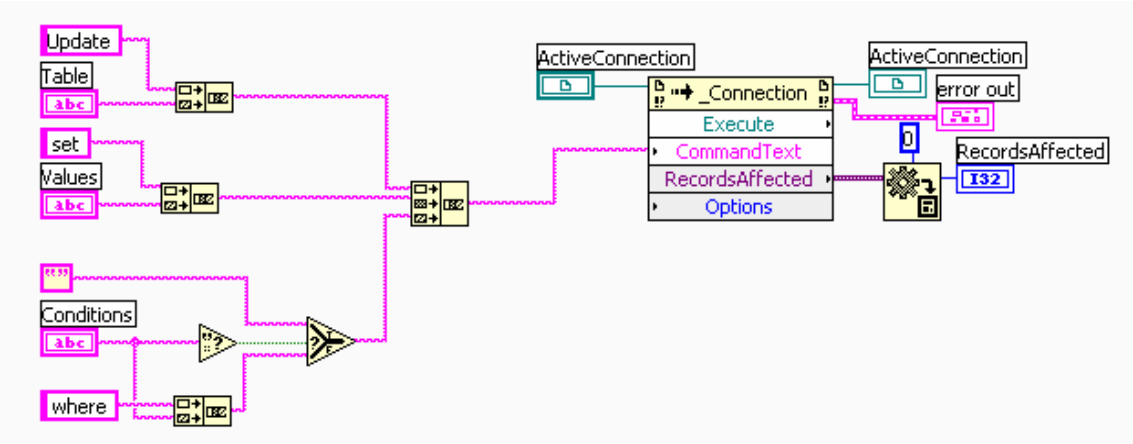

Figura 3.64 – Código del VI para Actualización de Registros

3.5.2.4. Pantallas de Configuración

3.5.2.4.1. Configuración de Usuarios

Una vez que se ha ingresado el password del usuario ‘Administrador’ continua la ejecución de la aplicación, entonces se procede con la configuración de los usuarios en una pantalla especialmente diseñada para el efecto. En la figura 3.65 se puede visualizar la pantalla de configuración de los usuarios donde se debe ingresar el nombre de usuario, la descripción, el password con su debida confirmación y el nivel de seguridad que se le asigna al mismo en valores de 1 a 10 correspondiente a usuario limitado y administrador respectivamente; a continuación se presiona el botón ‘Agregar’ y el usuario estará ya en la lista de los usuarios registrados. A la derecha se tiene una tabla donde se puede navegar en el registro de los usuarios, al dar clic sobre uno de ellos la información pasa a la pantalla de la izquierda y se puede realizar la eliminación del mismo. Finalmente se pulsa el botón ‘Continuar’ cuando ya se encuentran configurados todos los usuarios, para proseguir con la rutina de configuración inicial de la aplicación.

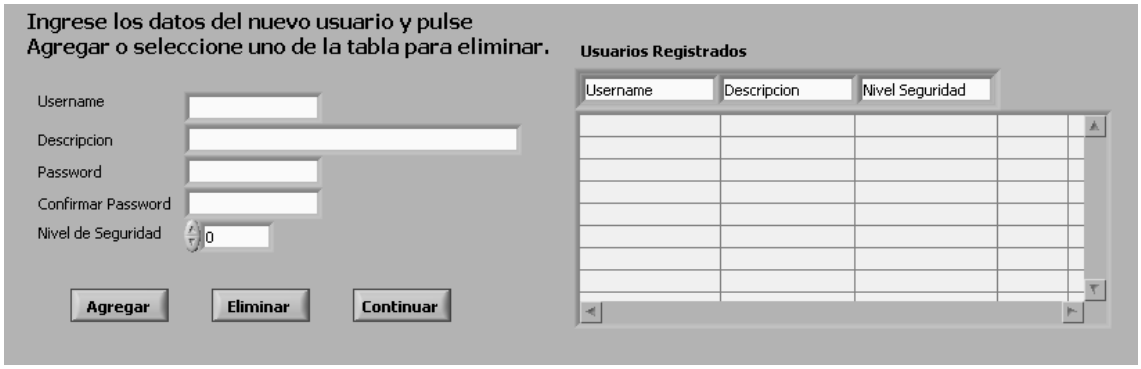

Figura 3.65 – Pantalla Inicial de Configuración de Usuarios

En la figura 3.66 se puede visualizar la pantalla de configuración de usuarios con algunos elementos ingresados.

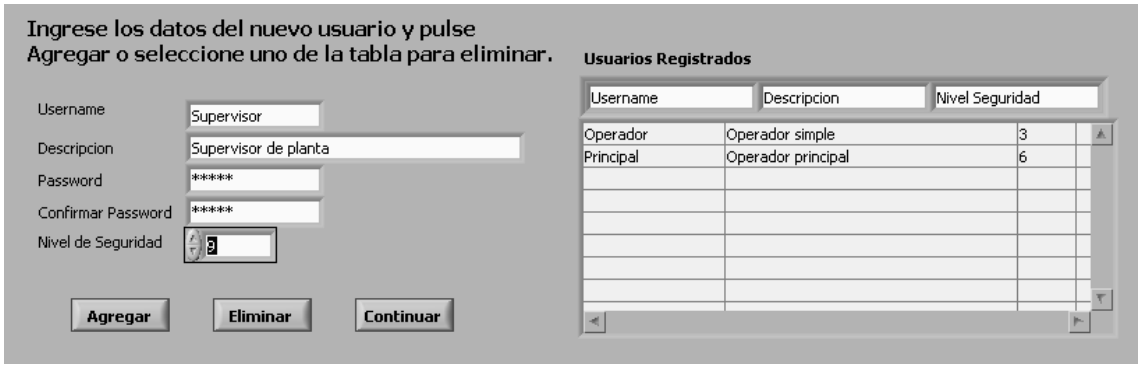

Figura 3.66 – Pantalla de Configuración de Usuarios con Registros

La programación para la configuración de usuarios se ha realizado de tal forma que responda como una pantalla orientada a los eventos de agregar un usuario, eliminar un usuario y continuar a la siguiente pantalla. En la figura 3.67 se puede visualizar la programación realizada para la pantalla de configuración de usuarios. Aquí se realiza primero una recuperación de todos los usuarios y los encabezados mediante una bandera de ejecución, para evitar que se esté realizando la misma consulta cada vez que se repite el código por estar dentro de una estructura while (mientras) para crear un comportamiento de eventos; luego se coloca la información en una tabla para que se pueda visualizar todos los usuarios registrados y se pueda realizar operaciones de inserción o eliminación. Cuando se pulsa el botón de ‘Eliminar’ se envía el nombre de usuario para eliminarlo y se coloca la bandera en verdadero para que se ejecute el código de recuperación de registros y se pueda visualizar el cambio

realizado. Al pulsar el botón de 'Agregar' primero se verifica que algunos campos no se encuentren en blanco debido a que son obligatorios, luego se construye la sentencia de SQL para la inserción de los datos y se realiza una limpieza de las cajas de texto de ingreso de información; finalmente se coloca la bandera en verdadero para que se ejecute el código de recuperación de registros y se pueda visualizar los registros insertados. Una vez terminada la configuración de los usuarios se pulsa el botón 'Continuar' y se pasa a la pantalla de configuración de áreas en las que se ha dividido la planta industrial.

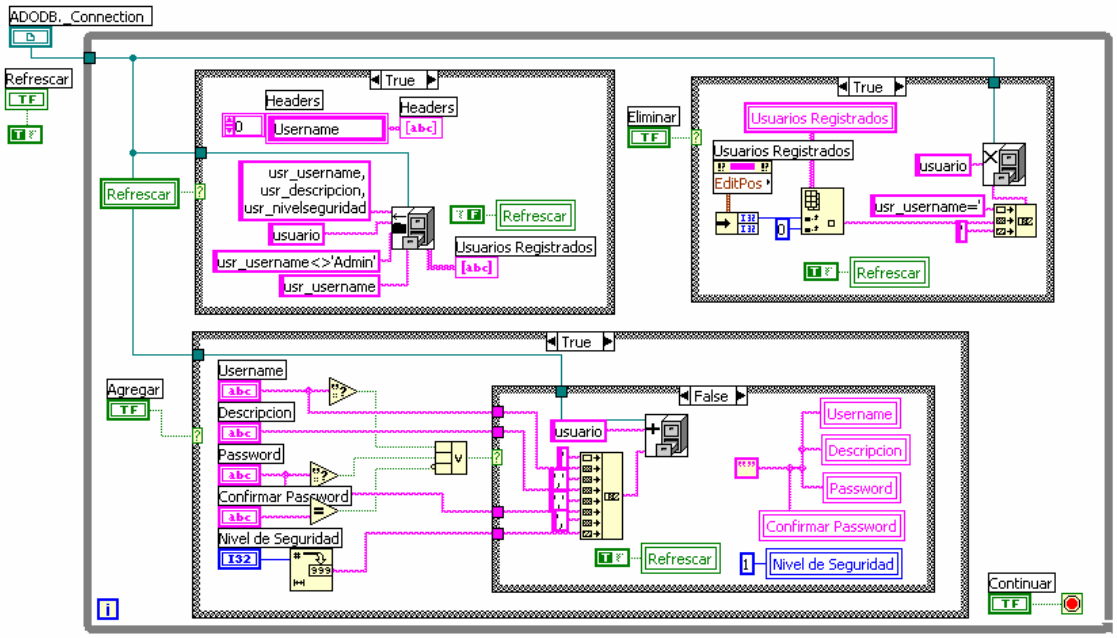

Figura 3.67 – Programación de Pantalla de Configuración de Usuarios

3.5.2.4.2. Configuración de Áreas de Planta

La programación de la pantalla de configuración de áreas se ha realizado también utilizando el mismo principio de la pantalla de configuración de usuarios para un comportamiento que responda a eventos. En la figura 3.68 se puede visualizar la pantalla de configuración de áreas que posee los campos de información, una tabla de datos y sus encabezados, los botones para agregar áreas, eliminar áreas y continuar el proceso de configuración.

Figura 3.68 – Pantalla Inicial de Configuración de Áreas

En la figura 3.69 se visualiza la pantalla de configuración de áreas con algunos registros ingresados.

Figura 3.69 – Pantalla de Configuración de Áreas con Registros

En la figura 3.70 se puede ver la programación realizada para la pantalla de configuración de áreas. Aquí se realiza primero una recuperación de todas las áreas y los encabezados mediante una bandera de ejecución, para evitar que se esté realizando la misma consulta cada vez que se repite el código por estar dentro de una estructura while (mientras) para crear un comportamiento de eventos; luego se coloca la información en una tabla para que se pueda visualizar todas las áreas registradas y se pueda realizar operaciones de inserción o eliminación. Cuando se pulsa el botón de ‘Eliminar’ se envía el código único de área para eliminarla y se coloca la bandera en verdadero para que se ejecute el código de recuperación de registros y se pueda visualizar el cambio realizado. Al pulsar el botón de ‘Agregar’ primero se verifica que algunos campos no se encuentren en blanco debido a que son obligatorios, luego se construye la sentencia de SQL para la inserción de los datos y se realiza una limpieza de las cajas de texto de ingreso de información; finalmente se coloca la bandera en verdadero para que se ejecute el código de recuperación de registros y se pueda visualizar los registros

insertados. Una vez terminada la configuración de las áreas se pulsa el botón ‘Continuar’ y se pasa a la pantalla de configuración de circuitos que se encuentran instalados en la planta industrial.

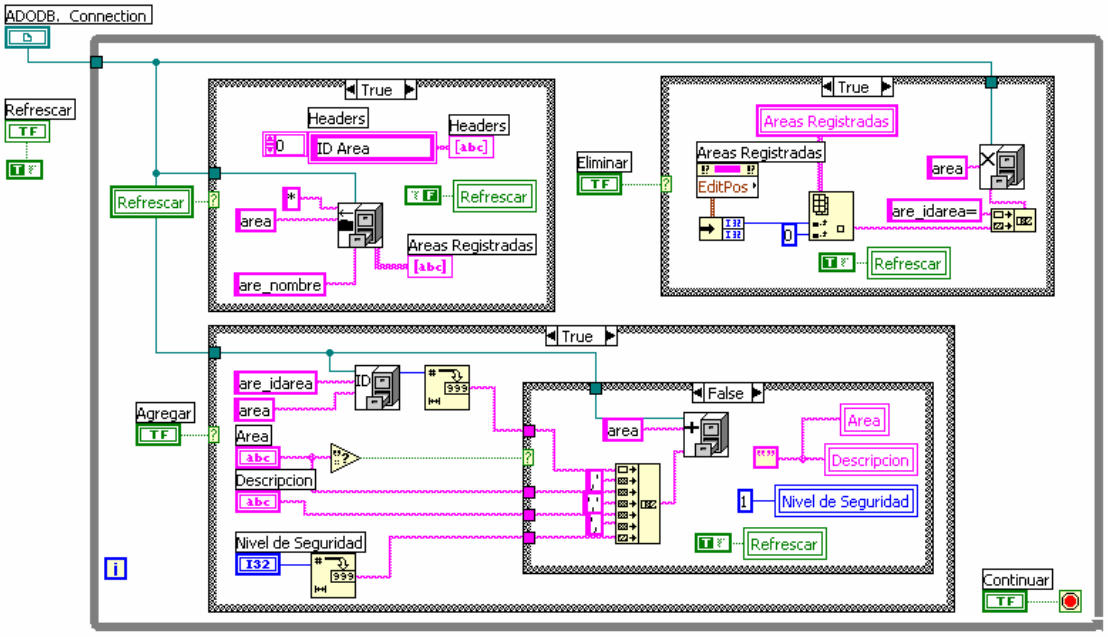

Figura 3.70 – Programación de Pantalla de Configuración de Áreas

3.5.2.4.3. Configuración de Circuitos de Área

La programación de la pantalla de configuración de circuitos se ha realizado también utilizando el mismo principio de la pantalla de configuración de usuarios para un comportamiento que responda a eventos. En la figura 3.71 se puede visualizar la pantalla de configuración de circuitos que posee los campos de información, una tabla de datos y sus encabezados, los botones para agregar circuitos, eliminar circuitos y continuar el proceso de configuración. Para facilitar el ingreso de la información se han colocado 2 combos desplegables que presentan el tipo de circuito que puede ser y el área a la que pertenece, para de esta forma limitar las opciones de entrada a los usuarios y así reducir el riesgo de que se cometan errores al ingresar los datos.

Figura 3.71 – Pantalla Inicial de Configuración de Circuitos

En la figura 3.72 se visualiza la pantalla de configuración de circuitos con algunos registros ingresados.

Figura 3.72 – Pantalla de Configuración de Circuitos con Registros

En la figura 3.73 se puede ver la programación realizada para la pantalla de configuración de circuitos. Aquí se realiza primero y una sola vez una recuperación de todas las áreas y tipos de circuitos que se cargarán en los combos de datos. A continuación, dentro del lazo se realiza una recuperación de los encabezados mediante una bandera de ejecución, para evitar que se esté realizando la misma consulta cada vez que se repite el código por estar dentro de una estructura while (mientras) para crear un comportamiento de eventos; luego se coloca la información en una tabla para que se pueda visualizar todos los circuitos registrados y se pueda realizar operaciones de inserción o eliminación. Cuando se pulsa el botón de ‘Eliminar’ se envía el código único de circuito para eliminarlo y se coloca la bandera en verdadero para que se ejecute el código de recuperación de registros y se pueda visualizar el cambio realizado. Al pulsar el botón de ‘Agregar’ primero se verifica que algunos campos no se encuentren en blanco debido a que son obligatorios, luego se construye la sentencia de SQL para la inserción de los datos y se realiza una limpieza de las cajas de texto de ingreso de información;

finalmente se coloca la bandera en verdadero para que se ejecute el código de recuperación de registros y se pueda visualizar los registros insertados. Una vez terminada la configuración de los circuitos se pulsa el botón ‘Continuar’ y se pasa a la pantalla de configuración de puntos de medición que posee cada circuito y que se encuentran instalados en la planta industrial.

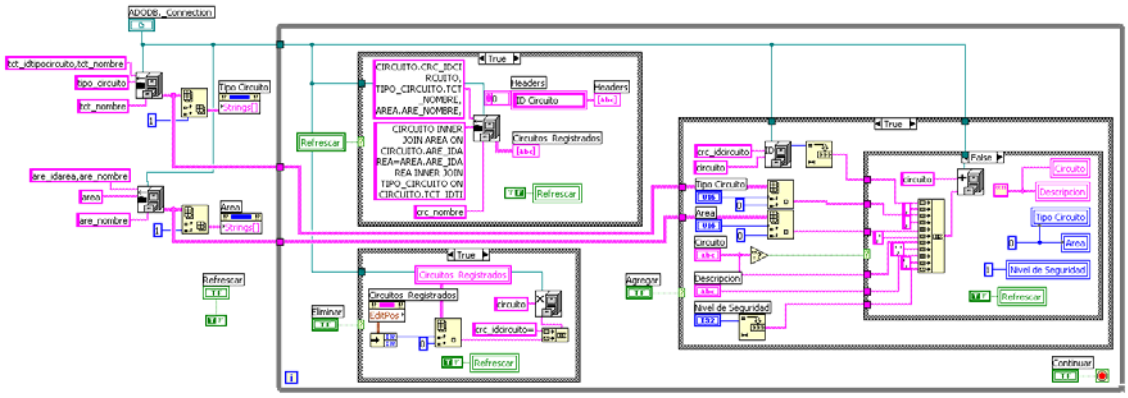

Figura 3.73 – Programación de Pantalla de Configuración de Circuitos

3.5.2.4.4. Configuración de Puntos de Medición

La programación de la pantalla de configuración de puntos de medición de circuitos se ha realizado también utilizando el mismo principio de la pantalla de configuración de usuarios para un comportamiento que responda a eventos. En la figura 3.74 se puede visualizar la pantalla de configuración de puntos de medición que posee los campos de información, una tabla de datos y sus encabezados, los botones para agregar puntos de medición, eliminar puntos de medición y continuar el proceso de configuración. Para facilitar el ingreso de la información se han colocado 2 combos desplegables que presentan el circuito al que pertenece y el tipo de medición que se realiza, para de esta forma limitar las opciones de entrada a los usuarios y así reducir el riesgo de que se cometan errores al ingresar los datos.

En la figura 3.75 se visualiza la pantalla de configuración de circuitos con algunos registros ingresados.

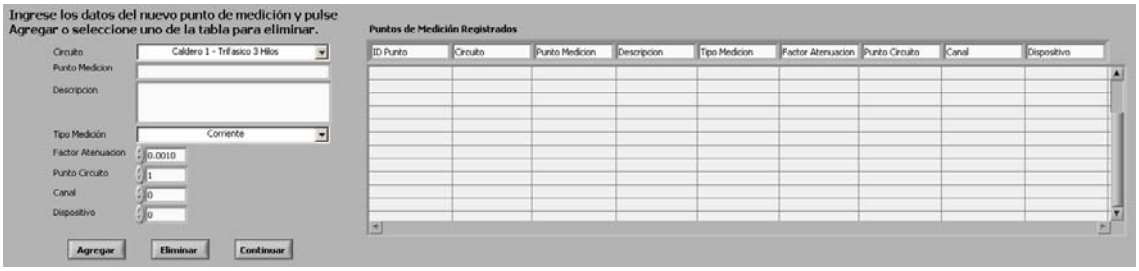

Figura 3.74 – Pantalla Inicial de Configuración de Puntos de Medición

Figura 3.75 – Pantalla de Configuración de Puntos de Medición con Registros

En la figura 3.76 se puede ver la programación realizada para la pantalla de configuración de puntos de medición. Aquí se realiza primero y una sola vez una recuperación de todos los circuitos y tipos de medición que se cargarán en los combos de datos. A continuación, dentro del lazo se realiza una recuperación de los encabezados mediante una bandera de ejecución, para evitar que se esté realizando la misma consulta cada vez que se repite el código por estar dentro de una estructura while (mientras) para crear un comportamiento de eventos; luego se coloca la información en una tabla para que se pueda visualizar todos los puntos de medición registrados y se pueda realizar operaciones de inserción o eliminación. Cuando se pulsa el botón de ‘Eliminar’ se envía el código único de punto de medición para eliminarlo y se coloca la bandera en verdadero para que se ejecute el código de recuperación de registros y se pueda visualizar el cambio realizado. Al pulsar el botón de ‘Agregar’ primero se verifica que algunos campos no se encuentren en blanco debido a que son obligatorios, luego se construye la sentencia de SQL para la inserción de los datos y se realiza una limpieza de las cajas de texto de ingreso de información; finalmente se coloca la bandera en verdadero para que se ejecute el código de recuperación de registros y se pueda visualizar los registros insertados. Una vez terminada la configuración de los puntos de medición se pulsa el botón ‘Continuar’ y se regresa a la pantalla principal, donde continúa el programa realizando la recuperación de

datos, conformación de los circuitos con los puntos y configuración de tarjetas DAQ para proceder al proceso de adquisición.

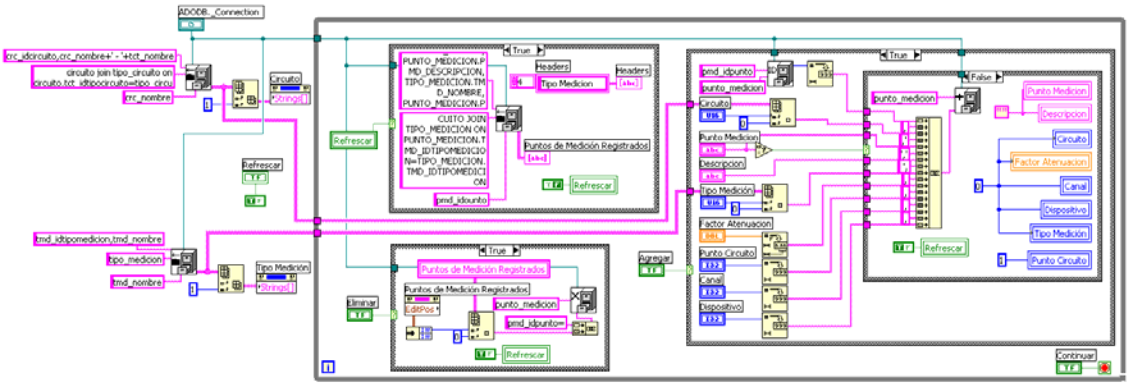

Figura 3.76 – Programación de Pantalla de Configuración de Puntos de Medición

3.5.2.5. Seguridad y Registro del Sistema

Cuando se ingresa a la aplicación en la segunda y posteriores veces se realiza el procedimiento de verificación de usuario para establecer si el mismo se encuentra registrado en el sistema y que nivel de permisos posee; esto se encuentra implementado en el instrumento virtual ‘Log In’ que tiene como entrada la referencia de ActiveX de la conexión a la DBMS; y como salidas el nombre de usuario, el nivel de seguridad y el número de intentos de ingreso al sistema. En la figura 3.77 se visualiza este VI.

Figura 3.77 – VI de Verificación de Usuarios para Ingreso al Sistema

La aplicación está diseñada de tal manera que se permite un máximo de tres intentos de ingreso para evitar que personal ajeno ingrese a la misma y cambie la información de todo lo concerniente a la planta industrial. Para esto se ha realizado una programación sencilla mostrada en la figura 3.78.

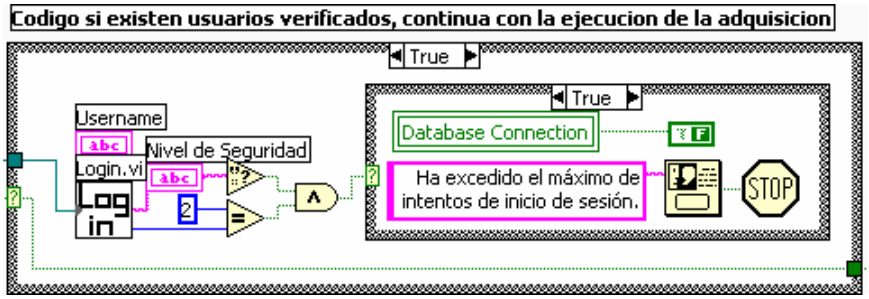

Figura 3.78 – Programación para Obtener los Tres Intentos Máximos

El VI de verificación de usuario tiene una pantalla frontal donde se ingresa el nombre de usuario y el password, la misma se puede visualizar en la figura 3.79.

Figura 3.79 – Pantalla del VI de Verificación de Usuarios

Dentro del VI de ingreso se realiza la tarea de verificar que los campos no sean blancos para que al pulsar 'Aceptar' no genere errores por falta de datos. En la figura 3.80 se visualiza la primera parte de la programación, donde primero se aplica un lazo while (mientras) que sirve para crear un comportamiento orientado a eventos; luego se crea la condición de búsqueda de la sentencia SQL del VI de selección y se pasa al siguiente cuadro. En la figura 3.81 se visualiza la segunda parte de la programación, aquí se usa la referencia de ActiveX de la entrada para el VI de selección. Luego se verifica que el estado de la selección haya sido uno, así como el número de registros recuperados, pues sólo debe existir un usuario con ese nombre y password dados.

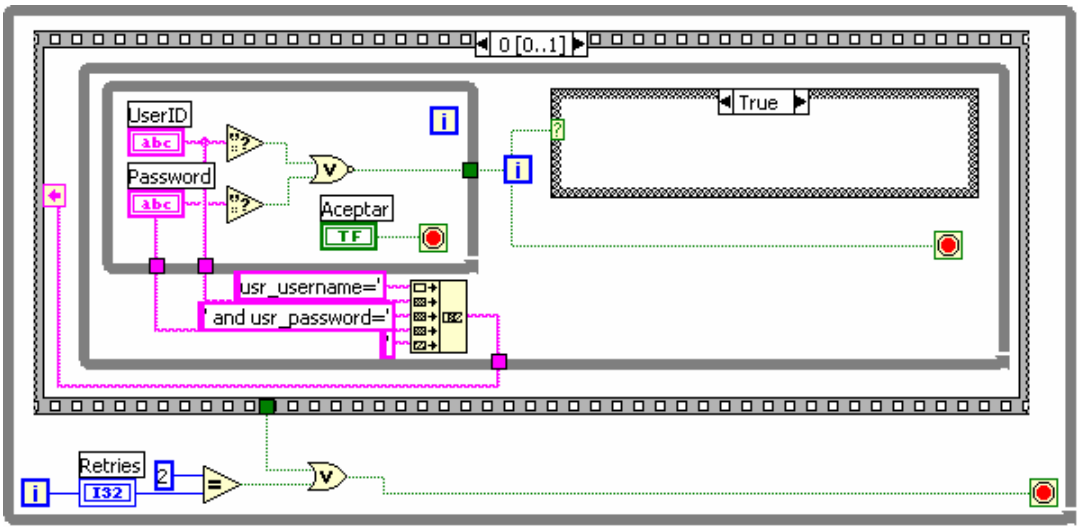

Figura 3.80 – Primera Pantalla de Programación del VI de Verificación de Usuarios

Si se cumplen las dos condiciones se recupera el nombre de usuario y el nivel de seguridad que posee, caso contrario emite un mensaje indicando que ese registro de usuario no se encuentra.

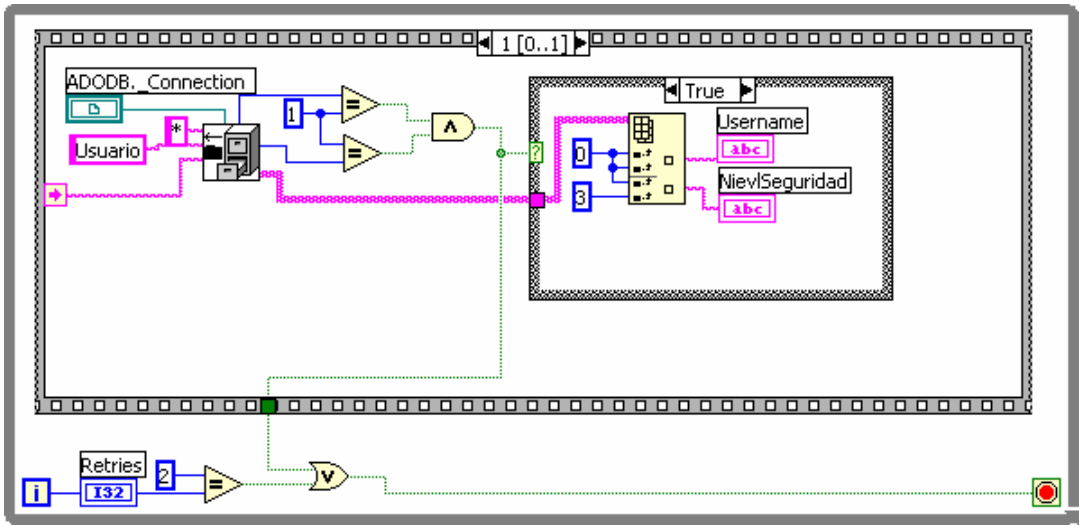

Figura 3.81 – Segunda Pantalla de Programación del VI de Verificación de Usuarios

3.5.2.5.1. Acceso al Cambio de Configuración, Niveles de Seguridad

Para realizar cambios en la configuración de la aplicación, se ha provisto cuatro botones en la pantalla principal que serán visibles sólo cuando el nivel de seguridad sea de 10. Estos

botones llamarán a las mismas pantallas de configuración antes vistas, de modo que el 'Administrador' se encuentre familiar con las interfases; en la figura 3.82 se visualiza los botones de configuración, el estado de la conexión a la DBMS, el nombre de usuario y el nivel de seguridad.

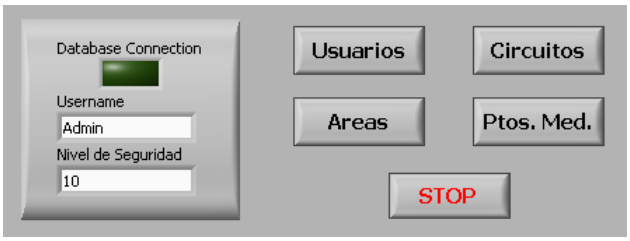

Figura 3.82 – Pantalla Principal con los Botones para Configuraciones

En la figura 3.83 se puede visualizar la programación realizada para los botones de configuración de la aplicación, donde se manipula la visibilidad de cada botón respecto al valor del nivel de seguridad y; cada botón llama al VI determinado cuando su valor es verdadero.

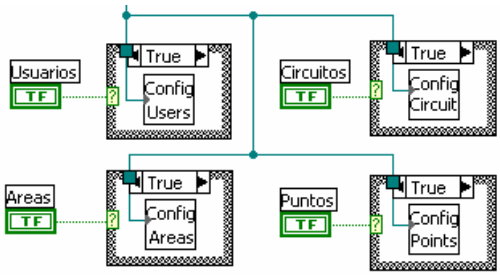

Figura 3.83 – Programación para Botones de Configuraciones

3.5.2.5.2. Registro de Trazas del Sistema

Este sistema de monitoreo cuenta con un esquema de seguridad básico que realiza trazas de las operaciones de inicio y finalización de la aplicación. Para lo cual se registra de forma transparente la fecha, hora, nombre de usuario y tipo de acción realizada; por lo tanto deben existir dos entradas de registro por cada ciclo completo de operación de la aplicación. Si no existen los dos registros en pares en la aplicación, es decir, entrada y salida de la misma;

quiere indicar que hubo una finalización anormal del sistema de monitoreo por razones diversas.

Para realizar las trazas del sistema se emplea el VI de inserción de registros explicado anteriormente. La fecha y hora se obtienen de la aplicación de monitoreo el momento de la inserción del registro y el nombre de usuario se almacena en una variable el momento de ingresar al sistema. El sencillo código se encuentra mostrado en la figura 3.84.

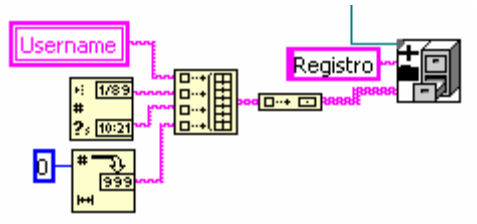

Figura 3.84 – Programación para Inserción de Registros de Traza

3.5.3. Programa de Adquisición y Almacenamiento de Datos

3.5.3.1. Carga de Datos de Circuitos

Luego de verificar el usuario y obtener su nivel de seguridad, la aplicación realiza la recuperación de los puntos de medición de los circuitos configurados en un formato agrupado y adecuado para manipular con facilidad los siguientes datos importantes:

- Código identificador del circuito
- Nombre del circuito
- Tipo de circuito (3 hilos, 3 hilos balanceado, 4 hilos)
- Arreglo de clusters de puntos de medición
  - ✓ Nombre del punto de medición
  - ✓ Tipo de medición (Corriente, Voltaje)
  - ✓ Factor de atenuación
  - ✓ Punto del circuito (1, 2, 3 según el tipo)

- ✓ Canal de tarjeta DAQ
- ✓ Número de tarjeta DAQ (puede haber varias instaladas)

Estos datos están agrupados en una estructura tipo cluster llamada ‘Cluster Circuito’. Al terminar la recuperación de datos se obtiene un arreglo de clusters de circuito. En la figura 3.85 se puede visualizar la pantalla frontal del VI de carga de circuitos que contiene el arreglo de clusters de circuito.

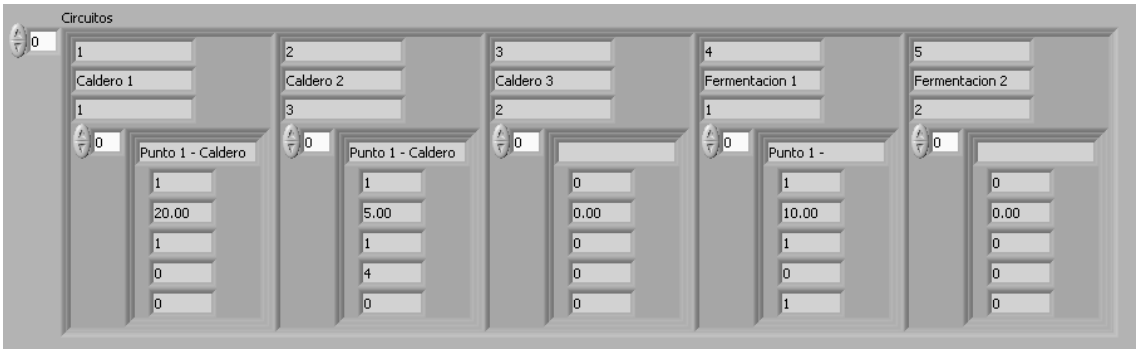

Figura 3.85 – Pantalla Frontal, Arreglo de Clusters de Circuito

Para la obtención de esta información se realiza una recuperación de todos los circuitos existentes en la DBMS, luego la información de los datos de puntos de medición de cada circuito para su posterior almacenamiento en clusters diseñados para el efecto.

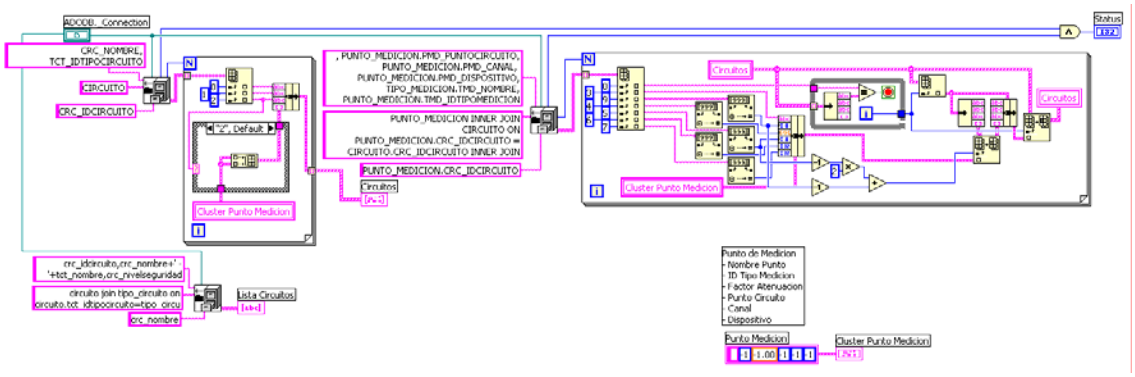

Figura 3.86 – Programación para Cargar Circuitos con Puntos de Medición

En la figura 3.86 se visualiza el código respectivo para obtener los circuitos y sus puntos de medición; así como también la obtención de cierta información adicional como el código

único del circuito, el nombre del circuito y el tipo del mismo para un cuadro de lista desplegable que se utiliza en la pantalla principal para elegir las formas de onda del circuito que se desea visualizar en el monitoreo.

3.5.3.2. Verificación de Tarjetas de Adquisición de Datos

Para realizar la verificación de las tarjetas DAQ se toma la información que proviene de cada uno de los puntos de medición configurados en una instancia específica de ejecución de la aplicación. En la figura 3.87 se puede visualizar que se tiene el mismo arreglo de clusters de Circuito previamente obtenido y en la parte inferior se tiene el valor para realizar una simulación cuando las tarjetas DAQ no están presentes teniendo el nivel de seguridad adecuado; así como también un arreglo de clusters con el número de tarjeta DAQ y un valor lógico indicador de presencia y configuración de la misma.

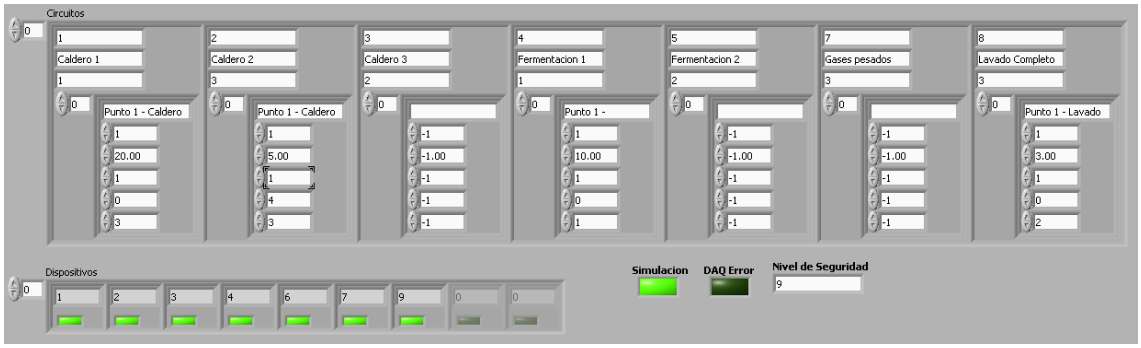

Figura 3.87 – Pantalla Frontal, Verificación de Tarjetas DAQ

En la programación de esta función primero se realiza un lazo a través de los circuitos y puntos de medición de cada uno para obtener un arreglo simplificado y ordenado de todas las tarjetas DAQ que se están utilizando. Los puntos de medición pueden ser 2, 4 ó 6 según sea el tipo de circuito. En la figura 3.88 se visualiza este doble lazo y la ordenación de la información.

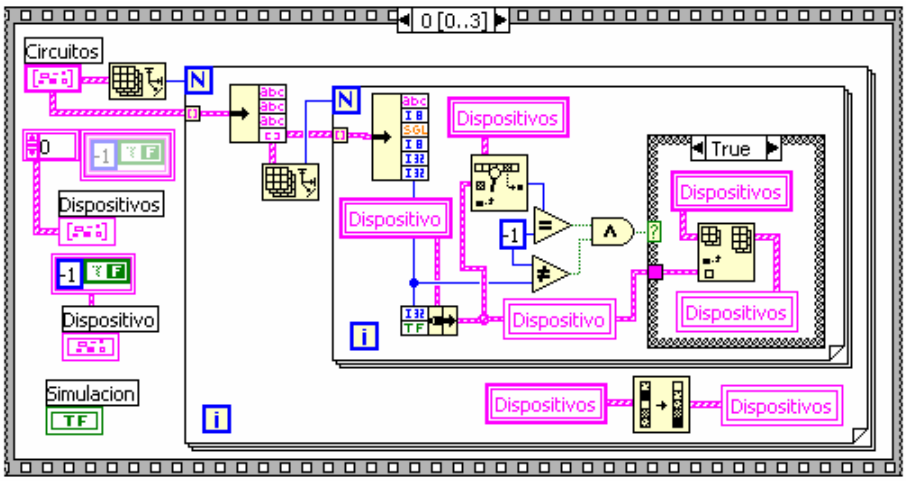

Figura 3.88 – Primera Parte de Programación de Verificación de Tarjetas DAQ

A continuación se realiza una verificación del Nivel de Seguridad para verificar si la aplicación puede funcionar en modo de simulación y se fija una bandera lógica para procesos posteriores, se visualiza en la figura 3.89.

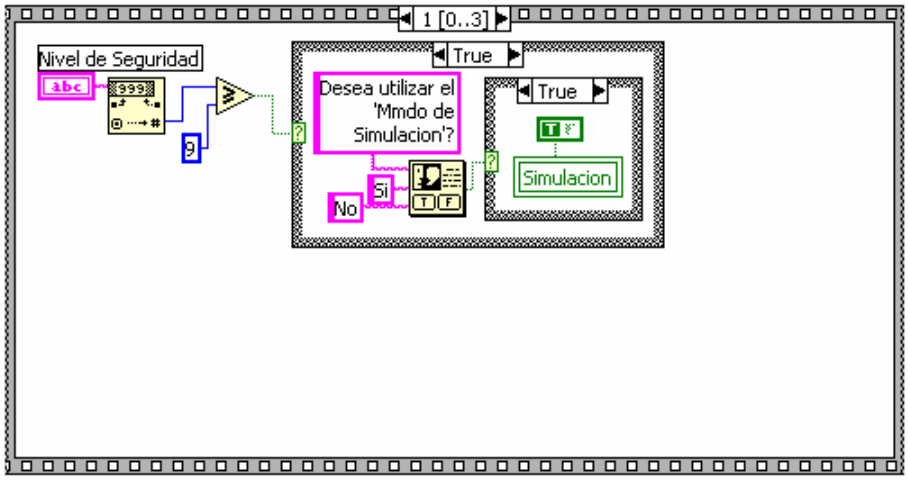

Figura 3.89 – Segunda Parte de Programación de Verificación de Tarjetas DAQ

Luego se realiza el proceso de configuración de las tarjetas DAQ mediante un recorrido por el arreglo de ‘Dispositivos’ obtenido anteriormente. Se utiliza el VI llamado ‘AI Group Config’ que toma el número de tarjeta DAQ y devuelve un código de error ‘0’ en caso de haber sido configurada correctamente; un error ‘-10401’ cuando la DAQ no está presente en el sistema, el driver no soporta el dispositivo, no es un producto de National Instruments o no ha sido configurada previamente mediante ‘Measurement & Automation Explorer’. En la figura

3.90 se puede ver que cuando sucede el error y la bandera de simulación está en falso se envía al usuario un cuadro de mensaje con el número de tarjeta DAQ, luego se activa la bandera 'DAQ Error' para proceder a la terminación de la aplicación.

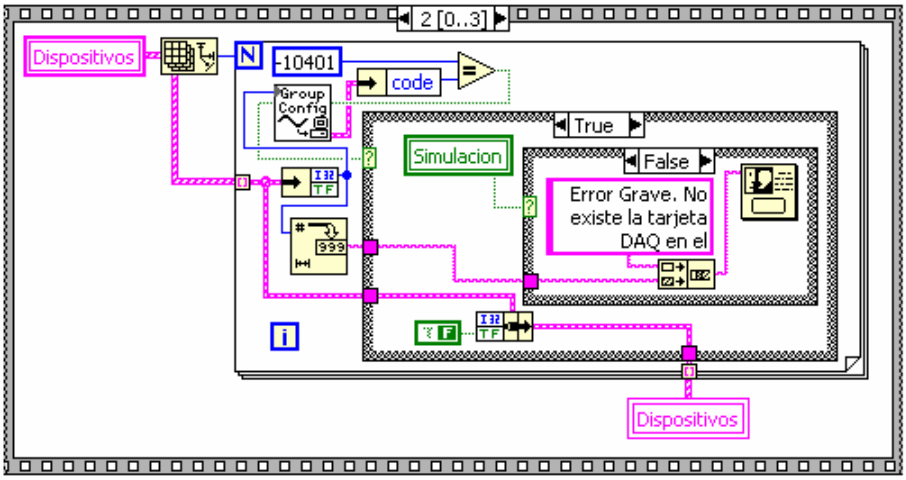

Figura 3.90 – Tercera Parte de Programación de Verificación de Tarjetas DAQ

Como salidas este VI tiene el arreglo de clusters de dispositivos, el valor de la bandera de simulación y la bandera de error de configuración de tarjeta DAQ que se utilizará posteriormente.

3.5.3.3. Adquisición de Datos

Este VI se ejecuta siempre y cuando la bandera de estado de recuperación de información de circuitos tenga el valor de 1, que es devuelto por el VI de 'Carga de Circuitos'. El VI de Adquisición de Datos tiene como entradas el arreglo de clusters de circuitos, el arreglo de clusters de tarjetas DAQ (dispositivos) y la bandera de simulación; como salidas tiene el cluster de datos adquiridos de los circuitos, que se compone de un arreglo de datos de formas de onda, valores RMS de corriente y voltaje, factor de potencia, frecuencia y los tres tipos de potencias. En la figura 3.91 se puede visualizar el icono del VI de adquisición de datos.

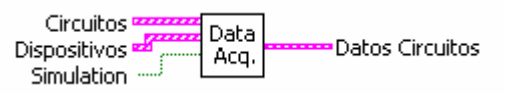

Figura 3.91 – VI de Adquisición de Datos

El VI de adquisición de datos obtiene los mismos en forma secuencial por código único de circuito y cíclicamente. Debido a que la aplicación también puede funcionar en modo de simulación para probar el correcto funcionamiento de la visualización y comunicación mediante DataSocket para el cliente Web, se utilizan datos aleatorios para generan los valores de impedancias de los circuitos. Luego se extrae información del cluster de circuito como el código único, el nombre, el tipo de circuito; y el cluster de puntos de medición porque este tiene toda la información con respecto a la atenuación que se aplicó antes de realizar la medición, el número de punto, el número de canal y dispositivo con el que se obtienen los datos. A continuación, mediante el tipo de circuito se realiza una selección debido a que existen circuitos trifásicos de 3 hilos balanceados, 3 hilos y 4 hilos. En la figura 3.92 se puede ver la programación del VI de adquisición de datos y el icono del VI de circuito trifásico de 3 hilos balanceado. Finalmente cada VI de circuito entrega los datos de formas de ondas para la visualización de monitoreo, valores RMS de voltaje y corriente, potencias, factor de potencia y frecuencia que son ensamblados en un nuevo cluster que se llama ‘Datos Circuitos’ y que es entregado como salida de este VI.

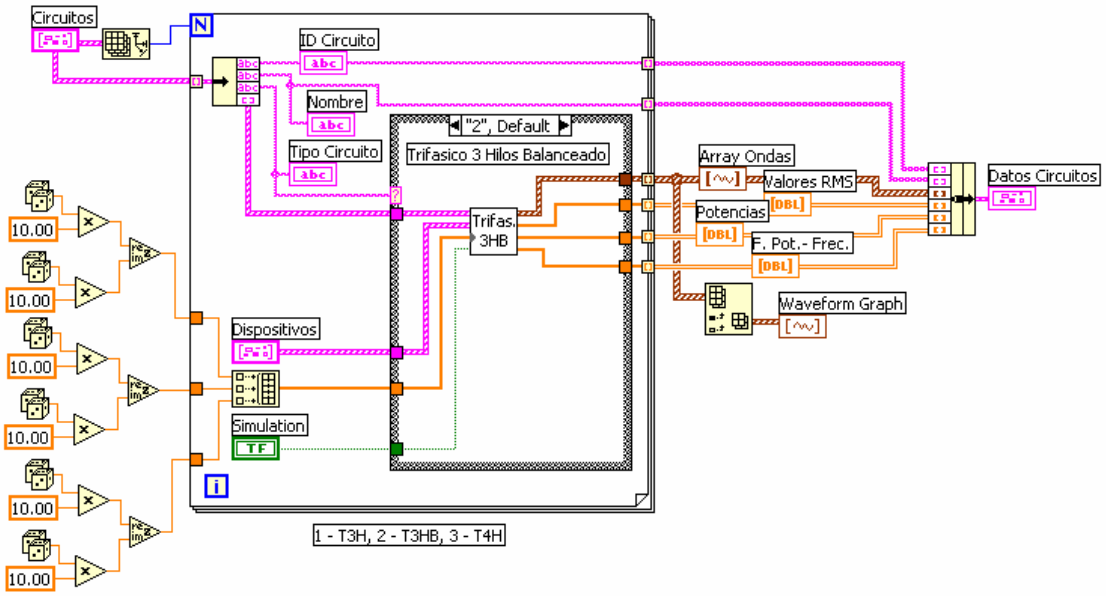

Figura 3.92 – Programación del VI de Adquisición de Datos

3.5.3.3.1. Análisis de Circuitos Trifásicos

Existen tres tipos de VI's de circuitos trifásicos que deben ser analizados en forma individual debido a que tienen diferentes características y forma de calcular los parámetros faltantes. En la figura 3.93 se puede visualizar la programación del circuito trifásico balanceado de tres hilos que puede reducirse al equivalente monofásico para su medición ó simulación.

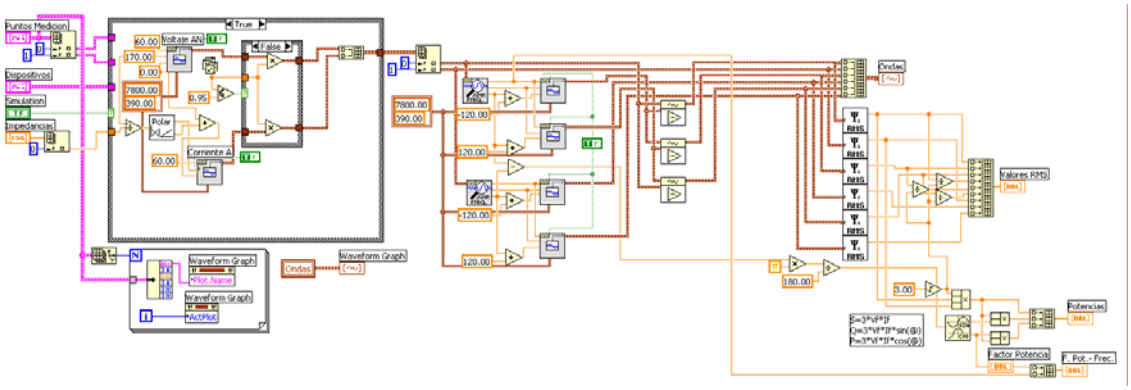

Figura 3.93 – Programación del VI de Circuito Trifásico Balanceado de 3 Hilos en Simulación

En la figura 3.94 se visualiza el mismo VI pero con la programación en modo de adquisición de datos de las tarjetas DAQ. Aquí se realiza una recuperación de parámetros de los clusters de cada punto de medición; necesarios para realizar la adquisición de datos como el número de tarjeta DAQ, el canal que se usa y el factor de atenuación utilizado.

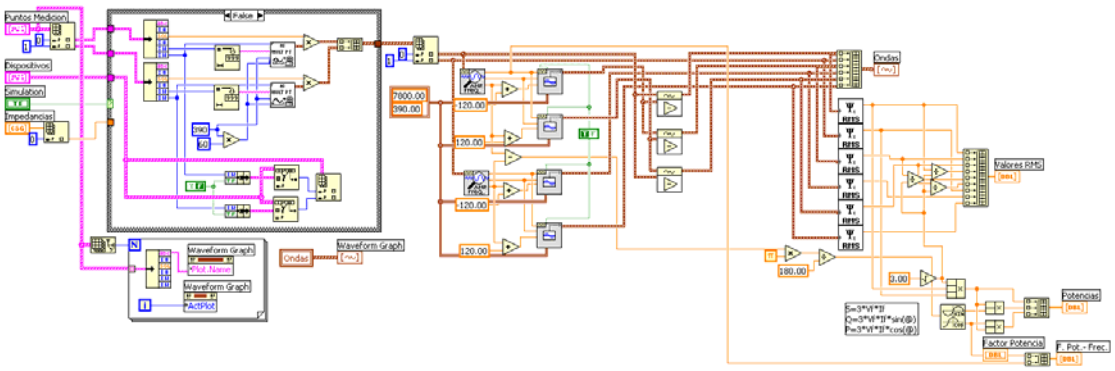

Figura 3.94 – Programación del VI de Circuito Trifásico Balanceado de 3 Hilos en Adquisición

En la figura 3.95 se visualiza la programación para la simulación del circuito trifásico de 4 hilos. Se nota que existe más programación en la parte de adquisición ó simulación de datos, pero más sencilla en lo referente a cálculos a realizarse para obtener los parámetros faltantes, pues en este tipo de circuito no se necesita hacer cálculos para corrientes ni voltajes. Sin embargo, si se necesitan hacer un poco más de cálculos para la obtención de potencias y factor de potencia total del circuito.

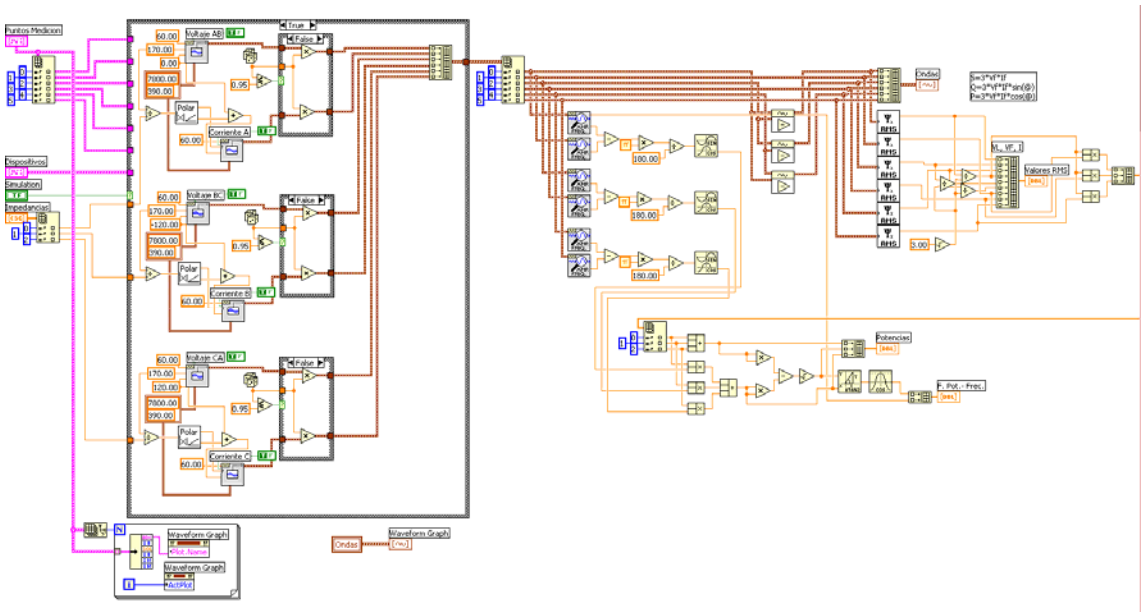

Figura 3.95 – Programación del VI de Circuito Trifásico de 4 Hilos en Simulación

En la figura 3.96 se visualiza la programación para el circuito trifásico de cuatro hilos en modo de adquisición de datos. Se puede ver que se realizan los mismos cálculos para obtener los voltajes de línea, sin embargo, las corrientes se las obtienen directamente de la medición o simulación. Este tipo de circuito puede tener cargas desbalanceadas, por lo tanto se debe realizar mayores cálculos para obtener las potencias y el factor de potencia del circuito.

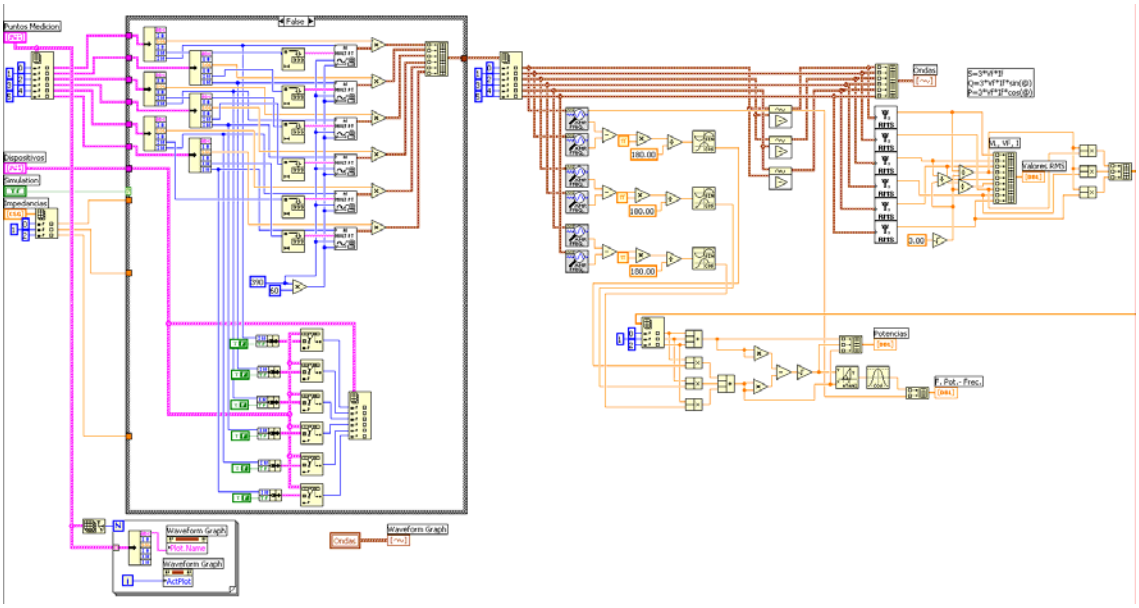

Figura 3.96 – Programación del VI de Circuito Trifásico de 4 Hilos en Adquisición

En la figura 3.97 se visualiza el panel frontal del VI de circuito trifásico de cuatro hilos. Aquí se muestra un arreglo de cluster de puntos de medición, el arreglo de cluster de dispositivos de adquisición, el arreglo de impedancias para la simulación, el indicador de bandera de simulación, los datos de las formas de onda, los datos de corrientes y voltajes RMS, potencias, factor de potencia total, frecuencia de las ondas y un graficador de formas de onda para voltajes de línea y corrientes.

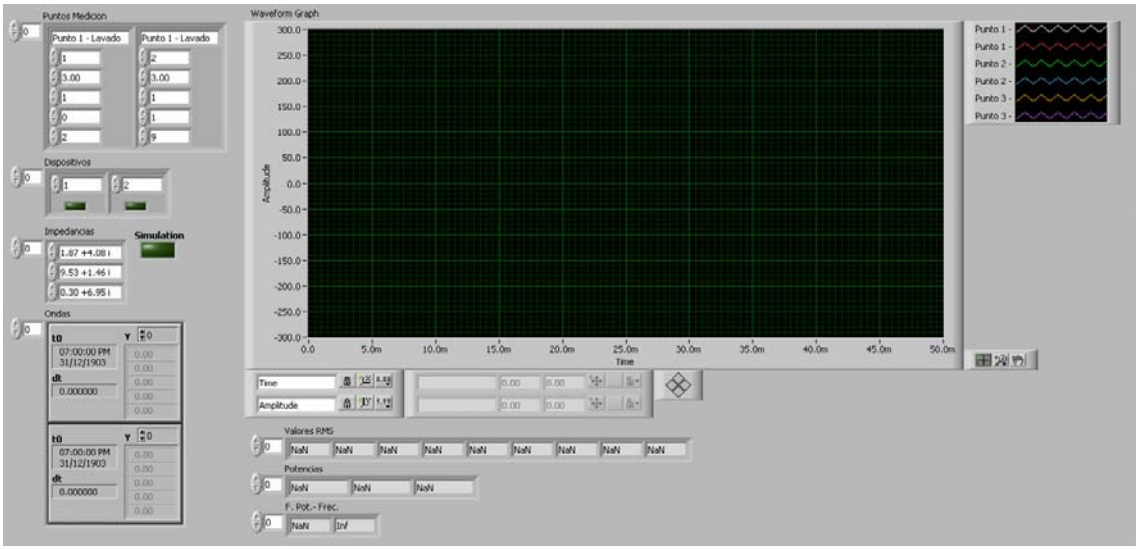

Figura 3.97 – Panel Frontal del VI de Circuito Trifásico de 4 Hilos en Adquisición

3.5.3.4. Datos Críticos y Almacenamiento de Información

Una vez obtenidos los datos desde el VI de adquisición de datos se desensambla el cluster de salida para poder tener la visualización de datos en forma ordenada en la pantalla principal de acuerdo al tipo de los mismos. En la figura 3.98 se visualiza la pantalla principal de la aplicación con cada uno de los indicadores y controles en forma ordenada y separada. Luego esta información se utiliza para el VI de obtención de datos críticos y almacenamiento en la DBMS. En la figura 3.99 se visualiza la programación completa del segundo cuadro, que se ejecuta en forma cíclica durante toda la aplicación. Aquí se puede ver que se hacen lecturas del tiempo al final de cada ciclo de adquisición de datos para verificar si se han cumplido 5 minutos para realizar el almacenamiento de información en la DBMS. También existen aquí controles de tipo booleano que llaman a los VI's de configuración y permite la visibilidad de ciertos indicadores como el graficador de formas de onda y de los valores de voltajes, corrientes, potencias y factor de potencia. Un control muy importante que se encuentra en este cuadro de programación es el que permite habilitar o deshabilitar la transmisión de datos al servidor de DataSocket para el cliente web.

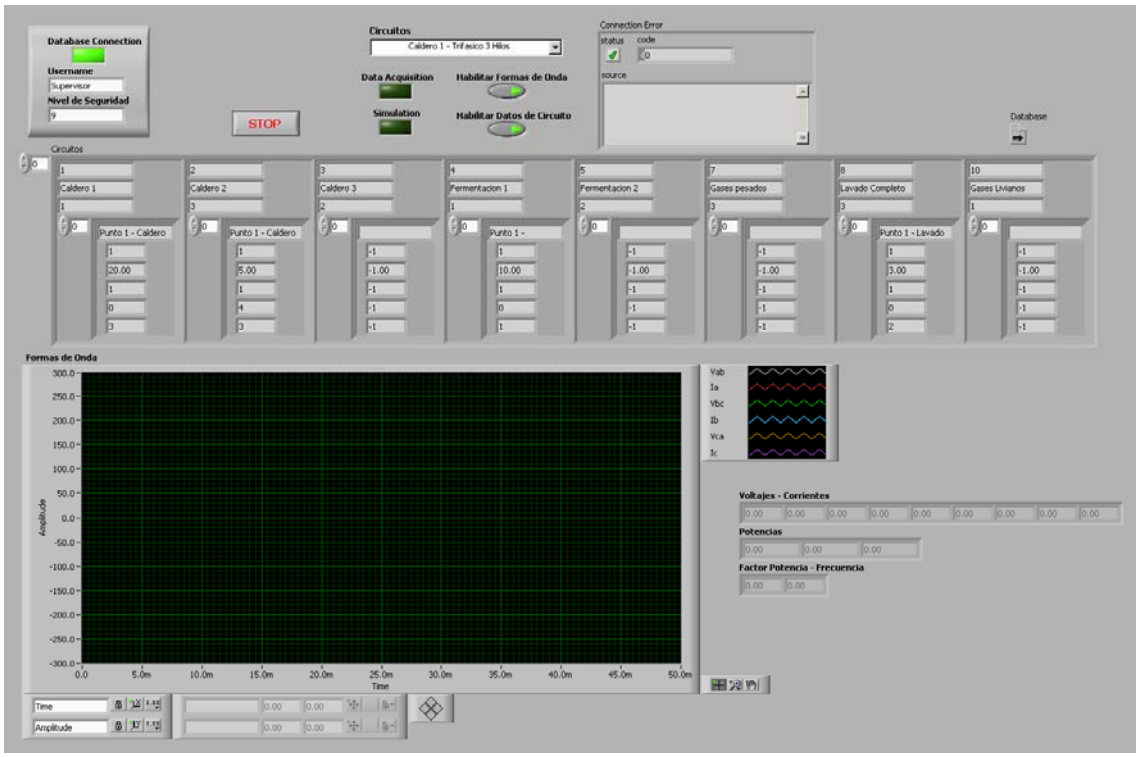

Figura 3.98 – Panel Principal de la Aplicación de Monitoreo

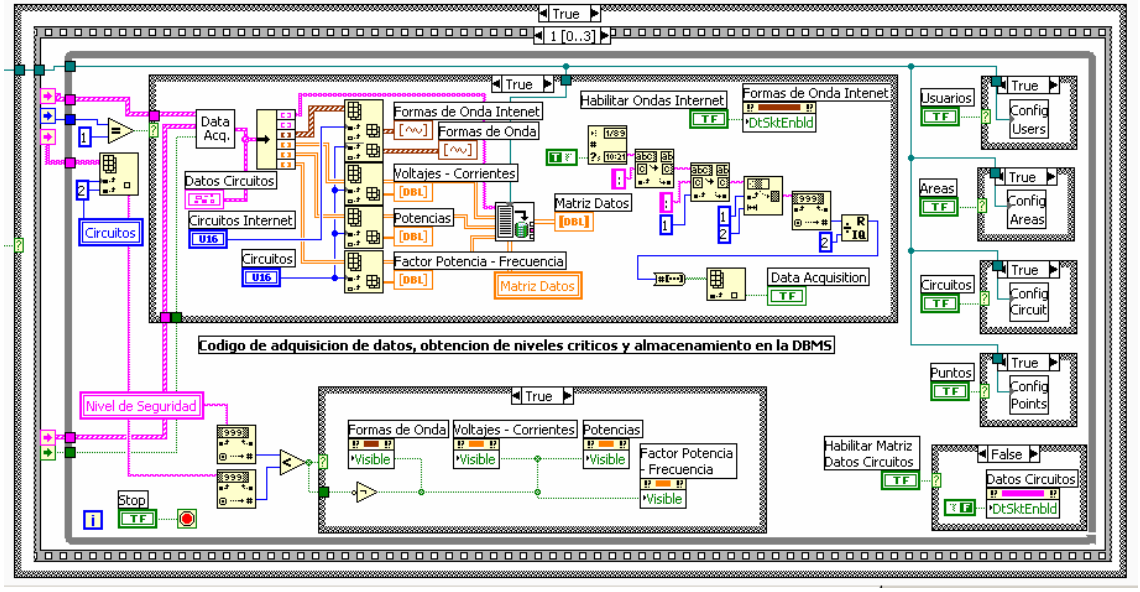

Figura 3.99 – Programación del Ciclo de Adquisición de Datos y Almacenamiento

3.5.3.4.1. Obtención de Datos Críticos

En el VI de almacenamiento de datos se desensambla nuevamente toda la información debido a que llega hecho un cluster de datos simples y no es posible usar la función para obtener máximos y mínimos. Por esta razón se deben construir nuevos arreglos para cada dato que ingresa y poder usar la función de máximos y mínimos de acuerdo al tipo de parámetro a almacenar. En la figura 3.100 se puede ver la programación donde se crean los arreglos y al final se obtienen los máximos y mínimos de los parámetros deseados; estos arreglos son salidas de este VI que se alimentarán nuevamente después del nuevo ciclo de adquisición de datos para tener un histórico de 5 minutos.

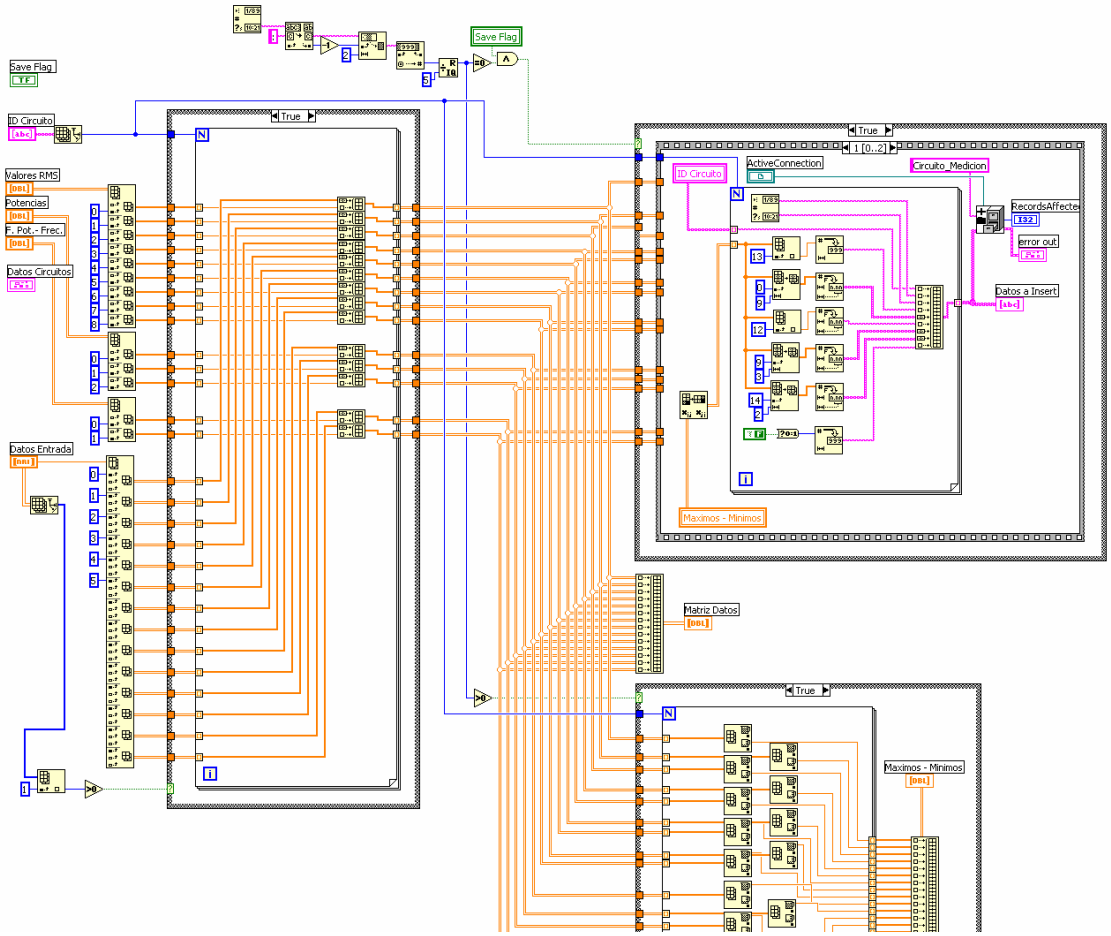

Figura 3.100 – Programación para Obtención de Máximos y Mínimos

3.5.3.4.2. Almacenamiento en la DBMS

Luego de obtener los máximos y mínimos en cada ciclo de 5 minutos se realizan los cálculos para obtener las energías y posteriormente poder almacenar los valores críticos de los parámetros en la DBMS, en la figura 3.101 se visualiza este proceso.

Para realizar el almacenamiento de los datos, primero se obtiene la fecha y hora del sistema, luego se debe realizar una conversión de datos numéricos a tipo string y; un reordenamiento de los mismos para que el VI de inserción de datos pueda funcionar de manera adecuada, en la figura 3.102 se puede apreciar este proceso.

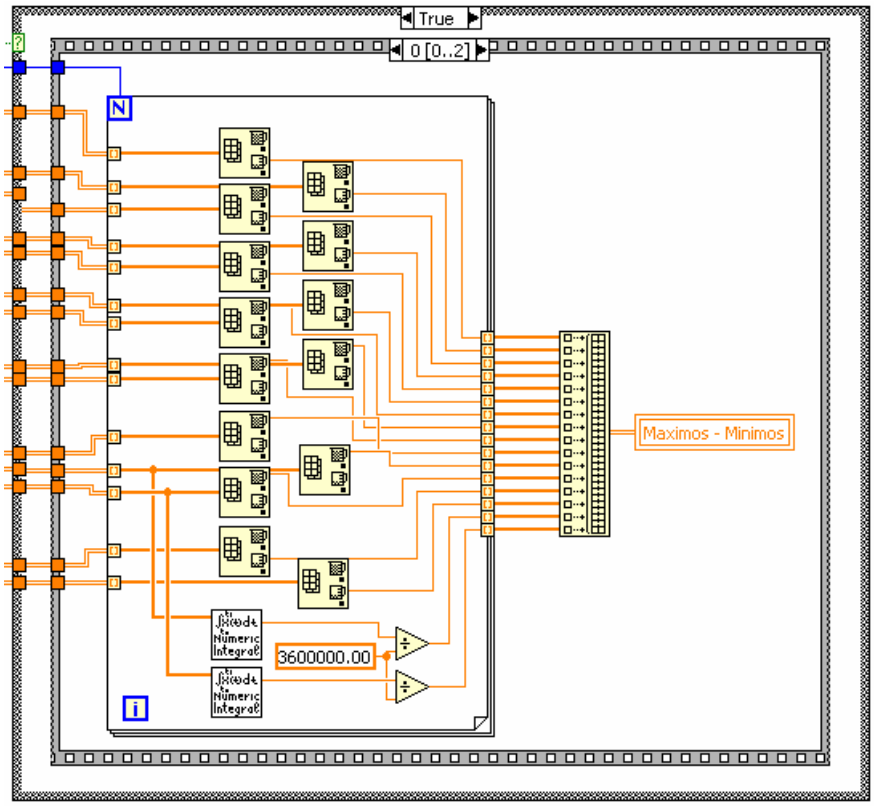

Figura 3.101 –Obtención de Máximos y Mínimos Finales, Cálculo de Energías

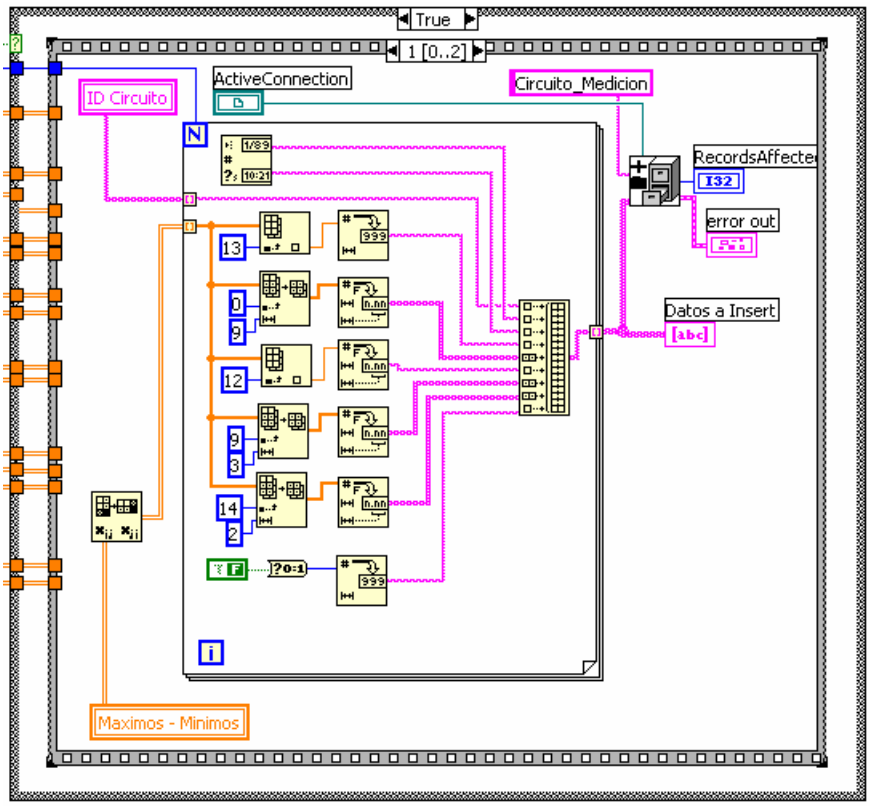

Figura 3.102 – Programación para Almacenamiento de Datos Críticos

Finalmente se procede a realizar una limpieza y redimensionamiento de los arreglos de datos históricos y de máximos-mínimos para el nuevo ciclo de adquisición de datos, terminando con la reinicialización de la bandera de grabación que evitará que entre en este caso nuevamente si no se cumplen las condiciones para realizar el almacenamiento de datos, este código se puede ver en la figura 3.103.

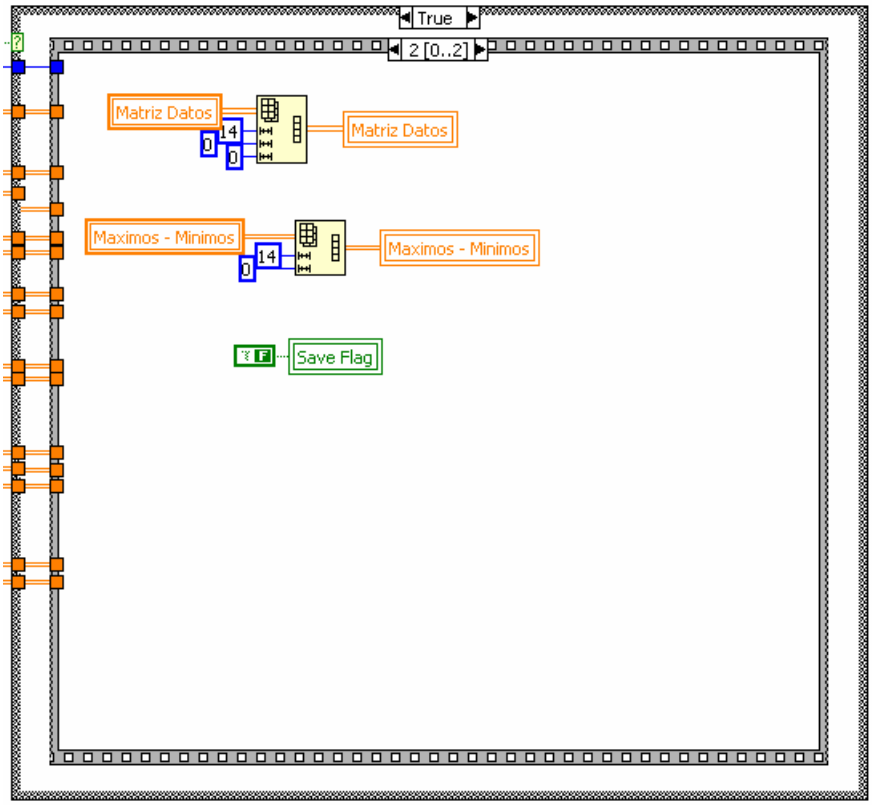

Figura 3.103 – Programación para Almacenamiento de Datos Críticos

3.5.3.5. Desconexión de la DBMS

Una vez finalizada la aplicación al pulsar el botón ‘Stop’ del panel principal se procede a realizar una desconexión de la DBMS para evitar cualquier pérdida o daño en los datos, se apaga los indicadores boléanos para mostrar que la adquisición de datos ha terminado y finalmente se hace un cierre de la automatización de ActiveX para finalizar en forma adecuada y liberar toda la memoria utilizada. Este proceso se visualiza en la figura 3.104.

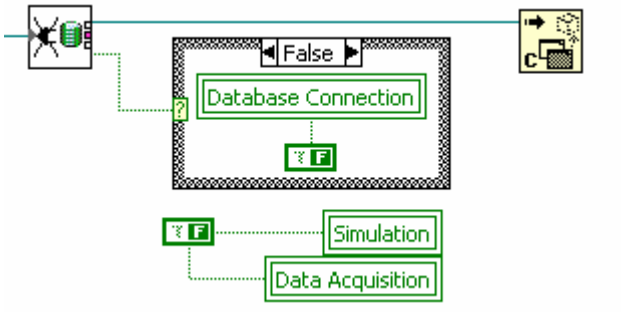

Figura 3.104 – Desconexión de la DBMS y Automatización de ActiveX

3.6. APLICACIÓN DEL CLIENTE

3.6.1. Esquema General

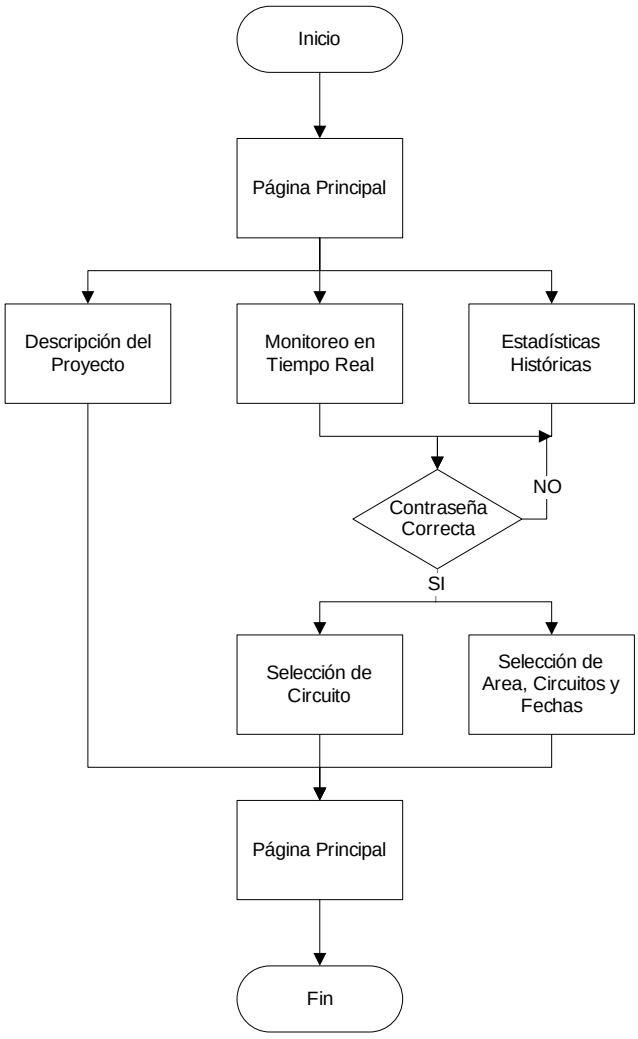

Figura 3.105 – Esquema General de Aplicación Cliente

La aplicación de cliente está totalmente desarrollada para que funcione sobre un entorno web, la misma es mucho más liviana y sencilla que la aplicación de servidor. En la figura 3.105 se visualiza un esquema general de la aplicación de cliente.

3.6.2. Descripción de la Aplicación de Cliente

La aplicación de cliente posee una página principal que es el menú principal de la aplicación. Desde este se puede dirigir hacia cualquier parte permitida de la aplicación en la secuencia programada. En la figura 3.106 se muestra la página principal con cada una de sus opciones.

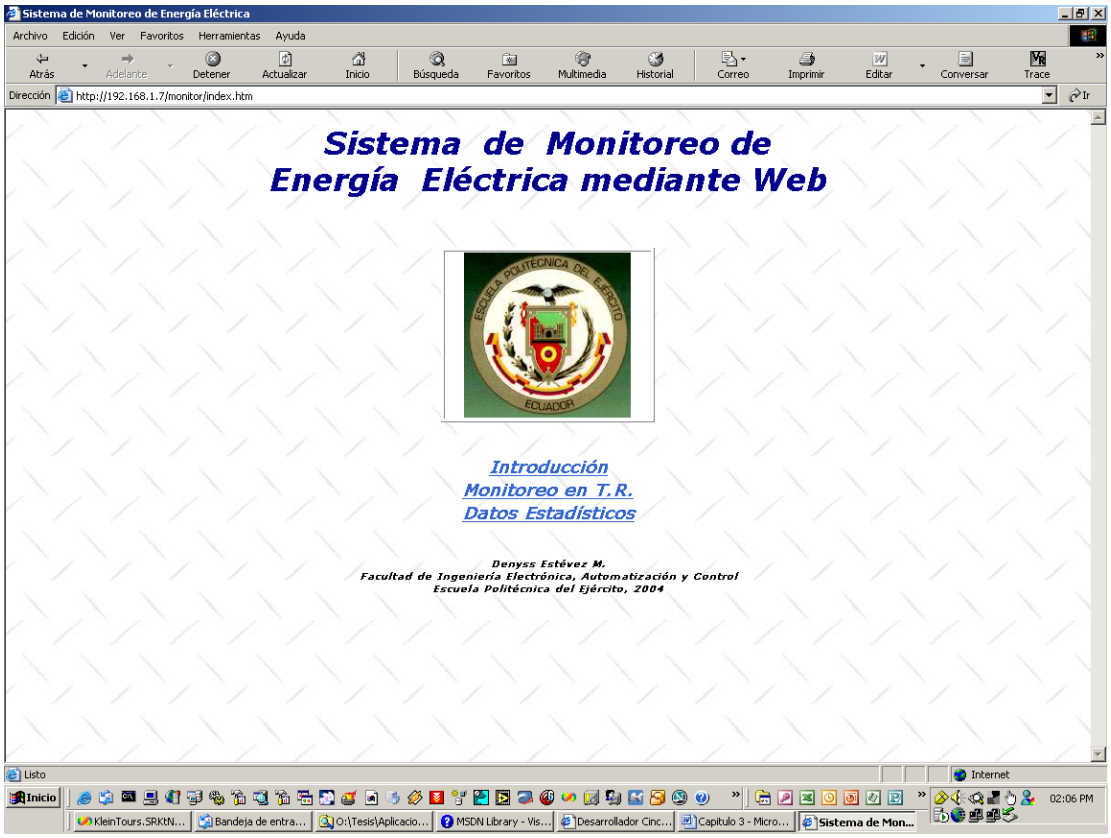

Figura 3.106 – Página Principal de la Aplicación de Cliente

La aplicación de cliente resulta de la adecuada unión de varios lenguajes y tecnologías para su funcionamiento. Existen tres secciones bien marcadas en la aplicación de cliente que son las siguientes:

- Página descriptiva del software
- Página de mediciones en tiempo real
- Página de estadísticas históricas de circuitos

A continuación se explicará en detalle cada una de estas secciones de la aplicación de cliente.

3.6.2.1. Página Descriptiva del Software

Esta es una página web sencilla realizada sobre lenguaje HTML plano, la cual contiene una descripción sobre el funcionamiento de los sistemas de adquisición de datos. Aquí también se describen las implementaciones tecnológicas realizadas sobre el proyecto de adquisición de datos de Energía Eléctrica mediante Web. En la figura 3.107 se muestra la página web que describe los sistemas DAQ y la implementación en el proyecto.

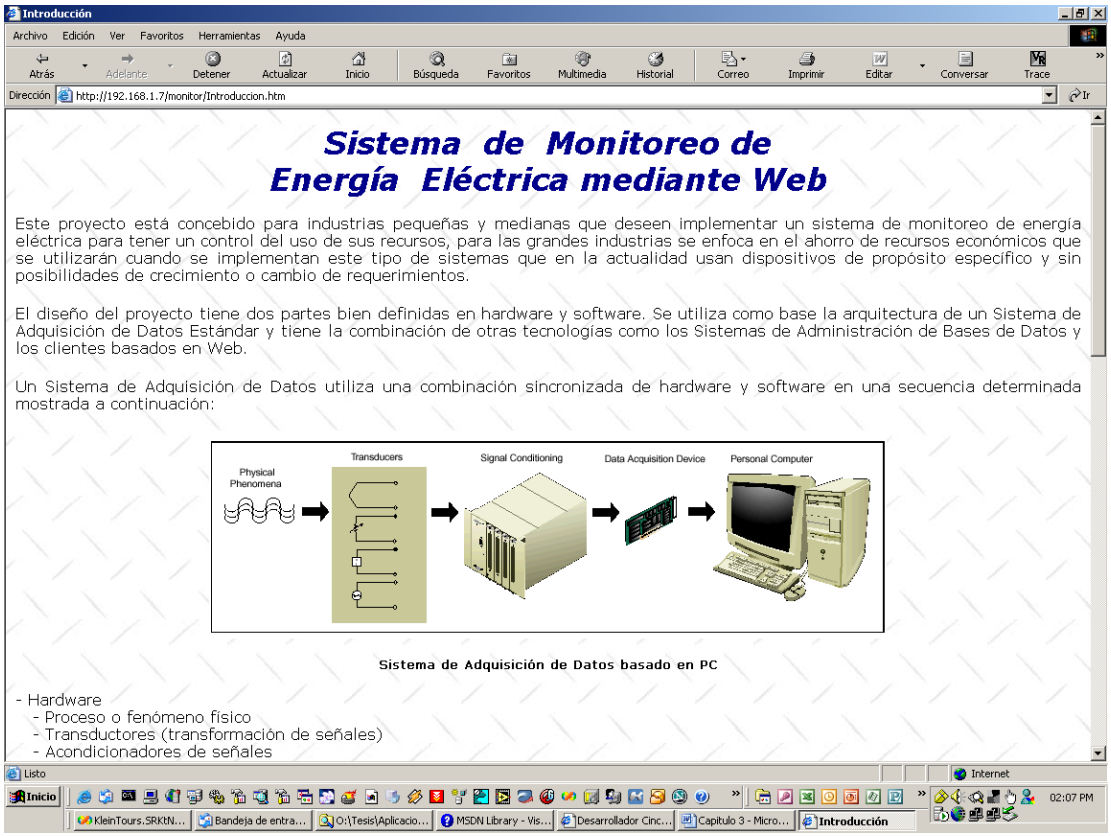

Figura 3.107 – Página Web Descriptiva de los Sistemas DAQ y el Proyecto

3.6.2.2. Página de Mediciones en Tiempo Real

Esta es una página web que contiene a un control ActiveX personalizado de 32 bits generado mediante Visual Basic 6.0 con controles ActiveX también, provenientes de LabVIEW 6.0i creados por National Instruments. Esta página contiene un botón que permite la conexión y desconexión del servidor de DataSocket que envía los datos de las ondas de los circuitos que están siendo monitoreados; un botón que permite habilitar y deshabilitar la visualización de las ondas de corriente y voltaje monitoreadas; una lista desplegable selectora de circuitos, donde estarán listados en orden alfabético e indicando el tipo del mismo todos los circuitos que comprenden el Sistema de Monitoreo; indicadores en forma de luces piloto de: error en las tarjetas DAQ cuando se ha producido un error en una tarjeta de adquisición de datos del sistema, adquisición de datos en forma parpadeante que indica cuando el sistema se encuentra en operación y simulación que indica cuando el sistema se encuentra simulando las ondas visualizadas; y finalmente la pantalla cuadriculada con escalas de tiempo en el eje X y escalas para voltios y amperios en el eje Y, que muestra las formas de onda de voltaje y corriente del circuito seleccionado mediante la lista desplegable antes mencionada.

Para el desarrollo de la nueva aplicación de cliente disponemos de los controles ActiveX de LabVIEW 6i que trae muchas mejoras sobre los controles de la versión 5.01 con que fue realizada la aplicación de cliente del proyecto de Cervecería Andina; entre las más importantes podemos mencionar una mejor integración con internet, disponibilidad de controles y visualizadores en 3 dimensiones que dan una apariencia más realista y natural para el operador; por lo tanto la aplicación reflejará un aspecto totalmente idéntico a cualquier panel de operación de maquinaria en la industria.

Para la recuperación de la información de conexión a la DBMS y al servidor de DataSocket se ha diseñado una porción de código de tal manera que lea el mismo registro de Windows que utiliza la aplicación de servidor, de tal manera que se reflejen inmediatamente los cambios realizados en la configuración como por ejemplo: el servidor de base de datos, la DBMS, el controlador de conexión o el servidor de DataSocket. Dado que este código se ejecuta en el servidor de componentes y lo activa el servidor de aplicaciones (Internet

Information Server para nuestro caso), no existe ningún problema que pueda presentarse al cambiar la aplicación de cliente de un servidor a otro; lo único que se requiere es realizar el registro del componente ocx de monitoreo en el servidor. A continuación se visualiza el código en Visual Basic 6.0 del control ocx que realiza la lectura del registro, la conexión con la DBMS y el servidor de Datasocket y, finalmente dibuja las ondas con los datos recibidos.

```
Option Explicit
Dim strProvider As String, strPassword As String, strUsername As String, strDBServer As String, strDatabase As String
Dim strDataSocket As String, astrUsers() As String, aintCircuitos() As Integer

Const HKEY_LOCAL_MACHINE = &H80000002
Const REG_SZ = 1
Const REG_DWORD = 4
Const KEY_READ = &H20000
Const KEY_QUERY_VALUE = &H1
Const KEY_SET_VALUE = &H2
Const KEY_CREATE_SUB_KEY = &H4
Const KEY_ENUMERATE_SUB_KEYS = &H8
Const KEY_NOTIFY = &H10
Const KEY_CREATE_LINK = &H20
Const KEY_ALL_ACCESS = KEY_QUERY_VALUE + KEY_SET_VALUE + KEY_CREATE_SUB_KEY + KEY_ENUMERATE_SUB_KEYS + _
KEY_NOTIFY + KEY_CREATE_LINK + KEY_READ
Private Declare Function RegOpenKeyEx Lib "advapi32" Alias "RegOpenKeyExA" (ByVal hKey As Long, ByVal lpSubKey As String, ByVal ulOptions As Long, ByVal samDesired As Long, ByRef phkResult As Long) As Long
Private Declare Function RegQueryValueEx Lib "advapi32" Alias "RegQueryValueExA" (ByVal hKey As Long, ByVal lpValueName As String, ByVal lpReserved As Long, ByRef lpType As Long, ByVal lpData As String, ByRef lpcbData As Long) As Long
Private Declare Function RegCloseKey Lib "advapi32" (ByVal hKey As Long) As Long
```

```
Private Sub cboCircuito_Click()
On Error GoTo Err_cboCircuito_Click
If Not IsNull(cboCircuito.Text) And cboCircuito.Text <> "" Then
dskSelector.Data = CInt(cboCircuito.ListIndex)
End If
Exit Sub

Err_cboCircuito_Click:
MsgBox Err.Description
End Sub

Private Sub cboCircuito_KeyDown(KeyCode As Integer, Shift As Integer)
On Error GoTo Err_cboCircuito_KeyDown

If KeyCode = 38 Or KeyCode = 40 Then cboCircuito_Click
Exit Sub

Err_cboCircuito_KeyDown:
MsgBox Err.Description
End Sub

Private Sub dskAdquisicion_OnDataUpdated(ByVal Data As CWDSLib.CWData)
On Error GoTo Err_dskAdquisicion_OnDataUpdated

If Data = True Then
shpDataAcquisition.BackColor = 49152
ElseIf Data = False Then
shpDataAcquisition.BackColor = 32768
End If
Exit Sub
```

Err_dskAdquisicion_OnDataUpdated:

MsgBox Err.Description

End Sub

Private Sub dskAdquisicion_OnStatusUpdated(ByVal Status As Long, ByVal Error As Long, ByVal Message As String)

On Error GoTo Err_dskAdquisicion_OnStatusUpdated

If Status = cwdsConnectionError Then

MsgBox "Error de conexion al Servidor de DataSocket", vbExclamation, "Sistema de Monitoreo"

End If

Exit Sub

Err_dskAdquisicion_OnStatusUpdated:

MsgBox Err.Description

End Sub

Private Sub dskEnable_OnStatusUpdated(ByVal Status As CWDSLib.CWDSStatus, ByVal Error As Long, ByVal Message As String)

On Error GoTo Err_dskEnable_OnStatusUpdated

If Status = cwdsConnectionError Then

MsgBox "Error de conexion al Servidor de DataSocket", vbExclamation, "Sistema de Monitoreo"

End If

Exit Sub

Err_dskEnable_OnStatusUpdated:

MsgBox Err.Description

End Sub

Private Sub dskErrorDAQ_OnDataUpdated(ByVal Data As CWDSLib.CWData)
On Error GoTo Err_dskErrorDAQ_OnDataUpdated

If Data = True Then
shpDAQError.BackColor = 255
ElseIf Data = False Then
shpDAQError.BackColor = 128
End If
Exit Sub

Err_dskErrorDAQ_OnDataUpdated:
MsgBox Err.Description
End Sub

Private Sub dskErrorDAQ_OnStatusUpdated(ByVal Status As Long, ByVal Error As Long, ByVal Message As String)
On Error GoTo Err_dskErrorDAQ_OnStatusUpdated

If Status = cwdsConnectionError Then
MsgBox "Error de conexion al Servidor de DataSocket", vbExclamation, "Sistema de Monitoreo"
End If
Exit Sub

Err_dskErrorDAQ_OnStatusUpdated:
MsgBox Err.Description
End Sub

Private Sub dskFactor_OnDataUpdated(ByVal Data As CWDSLib.CWData)
On Error GoTo Err_dskFactor_OnDataUpdated

```
If IsArray(Data) Then
txtFactor.Text = Format(Data.Value(0), "##,##0.00")
txtFrecuencia.Text = Format(Data.Value(1), "##,##0.00")
End If
Exit Sub

Err_dskFactor_OnDataUpdated:
MsgBox Err.Description
End Sub

Private Sub dskFactor_OnStatusUpdated(ByVal Status As Long, ByVal Error As Long, ByVal Message As String)
On Error GoTo Err_dskFactor_OnStatusUpdated

If Status = cwdsConnectionError Then
MsgBox "Error de conexion al Servidor de DataSocket", vbExclamation, "Sistema de Monitoreo"
End If
Exit Sub

Err_dskFactor_OnStatusUpdated:
MsgBox Err.Description
End Sub

Private Sub dskPotencias_OnDataUpdated(ByVal Data As CWDSLib.CWData)
On Error GoTo Err_dskPotencias_OnDataUpdated

If IsArray(Data) Then
txtPotenciaAparente.Text = Format(Data.Value(0), "##,##0.00")
txtPotencia.Text = Format(Data.Value(1), "##,##0.00")
```

```
txtPotenciaReactiva.Text = Format(Data.Value(2), "##,##0.00")
End If
Exit Sub

Err_dskPotencias_OnDataUpdated:
MsgBox Err.Description
End Sub

Private Sub dskPotencias_OnStatusUpdated(ByVal Status As Long, ByVal Error As Long, ByVal Message As String)
On Error GoTo Err_dskPotencias_OnStatusUpdated

If Status = cwdsConnectionError Then
MsgBox "Error de conexion al Servidor de DataSocket", vbExclamation, "Sistema de Monitoreo"
End If
Exit Sub

Err_dskPotencias_OnStatusUpdated:
MsgBox Err.Description
End Sub

Private Sub dskSelector_OnStatusUpdated(ByVal Status As CWDSLib.CWDSStatus, ByVal Error As Long, ByVal Message As String)
On Error GoTo Err_dskSelector_OnStatusUpdated

lblStatusSelector.Caption = Message
If Status = cwdsConnectionError Then
MsgBox "Error de conexion al Servidor de DataSocket", vbExclamation, "Sistema de Monitoreo"
End If
```

Exit Sub

Err_dskSelector_OnStatusUpdated:
MsgBox Err.Description
End Sub

Private Sub dskSimulacion_OnDataUpdated(ByVal Data As CWDSLib.CWData)
On Error GoTo Err_dskSimulacion_OnDataUpdated

If Data = True Then
shpSimulation.BackColor = 49152
ElseIf Data = False Then
shpSimulation.BackColor = 32768
End If
Exit Sub

Err_dskSimulacion_OnDataUpdated:
MsgBox Err.Description
End Sub

Private Sub dskSimulacion_OnStatusUpdated(ByVal Status As Long, ByVal Error As Long, ByVal Message As String)
On Error GoTo Err_dskSimulacion_OnStatusUpdated

If Status = cwdsConnectionError Then
MsgBox "Error de conexion al Servidor de DataSocket", vbExclamation, "Sistema de Monitoreo"
End If
Exit Sub

Err_dskSimulacion_OnStatusUpdated:

MsgBox Err.Description
End Sub

Private Sub dskOndas_OnDataUpdated(ByVal Data As CWDSLib.CWData)
On Error GoTo Err_dskOndas_OnDataUpdated

If IsArray(Data) Then
cwgOndas.PlotY Data.Value
End If
Exit Sub

Err_dskOndas_OnDataUpdated:
MsgBox Err.Description
End Sub

Private Sub dskOndas_OnStatusUpdated(ByVal Status As CWDSLib.CWDSStatus, ByVal Error As Long, ByVal Message As String)
On Error GoTo Err_dskOndas_OnStatusUpdated

lblStatusWaves.Caption = Message
If Status = cwdsConnectionError Then
MsgBox "Error de conexion al Servidor de DataSocket", vbExclamation, "Sistema de Monitoreo"
End If
Exit Sub

Err_dskOndas_OnStatusUpdated:
MsgBox Err.Description
End Sub

Private Sub dskVoltajes_OnDataUpdated(ByVal Data As CWDSLib.CWData)

```
On Error GoTo Err_dskVoltajes_OnDataUpdated

If IsArray(Data) Then
txtVL1.Text = Format(Data.Value(0), "##,##0.00")
txtVL2.Text = Format(Data.Value(1), "##,##0.00")
txtVL3.Text = Format(Data.Value(2), "##,##0.00")
txtVF1.Text = Format(Data.Value(3), "##,##0.00")
txtVF2.Text = Format(Data.Value(4), "##,##0.00")
txtVF3.Text = Format(Data.Value(5), "##,##0.00")
txtIL1.Text = Format(Data.Value(6), "##,##0.00")
txtIL2.Text = Format(Data.Value(7), "##,##0.00")
txtIL3.Text = Format(Data.Value(8), "##,##0.00")
End If
Exit Sub

Err_dskVoltajes_OnDataUpdated:
MsgBox Err.Description
End Sub

Private Sub dskVoltajes_OnStatusUpdated(ByVal Status As Long, ByVal Error As Long, ByVal Message As String)
On Error GoTo Err_dskVoltajes_OnStatusUpdated

If Status = cwdsConnectionError Then
MsgBox "Error de conexion al Servidor de DataSocket", vbExclamation, "Sistema de Monitoreo"
End If
Exit Sub

Err_dskVoltajes_OnStatusUpdated:
MsgBox Err.Description
```

```
End Sub

Private Sub tbtConnection_Click()
On Error GoTo Err_tbtConnection_Click

If tbtConnection.Value = True Then
dskEnable.ConnectTo "dstp://" & strDataSocket & "/habilita", cwdsWriteAutoUpdate
dskSelector.ConnectTo "dstp://" & strDataSocket & "/IDCircuito", cwdsWriteAutoUpdate
dskErrorDAQ.ConnectTo "dstp://" & strDataSocket & "/ErrorDAQ",
cwdsReadAutoUpdate
dskAdquisicion.ConnectTo "dstp://" & strDataSocket & "/adquisicion",
cwdsReadAutoUpdate
dskSimulacion.ConnectTo "dstp://" & strDataSocket & "/simulacion",
cwdsReadAutoUpdate
dskOndas.ConnectTo "dstp://" & strDataSocket & "/ondas", cwdsReadAutoUpdate
dskVoltajes.ConnectTo "dstp://" & strDataSocket & "/voltajes", cwdsReadAutoUpdate
dskPotencias.ConnectTo "dstp://" & strDataSocket & "/potencias", cwdsReadAutoUpdate
dskFactor.ConnectTo "dstp://" & strDataSocket & "/factor", cwdsReadAutoUpdate
tbtEnable.Enabled = True
cboCircuito.Enabled = True
tbtConnection.Caption = "Desconectar"
ElseIf tbtConnection.Value = False Then
dskEnable.Disconnect
dskSelector.Disconnect
dskErrorDAQ.Disconnect
dskAdquisicion.Disconnect
dskSimulacion.Disconnect
dskOndas.Disconnect
dskVoltajes.Disconnect
dskPotencias.Disconnect
dskFactor.Disconnect
```

```
shpDAQError.BackColor = 12632256
shpDataAcquisition.BackColor = 12632256
shpSimulation.BackColor = 12632256
cwgOndas.ClearData
tbtEnable.Enabled = False
cboCircuito.Enabled = False
tbtConnection.Caption = "Conectar"
End If
Exit Sub

Err_tbtConnection_Click:
MsgBox Err.Description
End Sub

Private Sub tbtEnable_Click()
On Error GoTo Err_tbtEnable_Click

If tbtEnable.Value = True Then
tbtEnable.Caption = "Deshabilitar"
ElseIf tbtEnable.Value = False Then
tbtEnable.Caption = "Habilitar"
cwgOndas.ClearData
End If
dskEnable.Data = tbtEnable.Value
Exit Sub

Err_tbtEnable_Click:
MsgBox Err.Description
End Sub

Private Sub UserControl_Initialize()
```

```
On Error GoTo Err_UserControl_Initialize

Dim acnDatabase As ADODB.Connection, arsCircuitos As ADODB.Recordset, strConnection As String, strSelection As String
Dim arsUsers As ADODB.Recordset, intIndex As Integer

Set acnDatabase = New ADODB.Connection
Set arsCircuitos = New ADODB.Recordset
Set arsUsers = New ADODB.Recordset

LoadConfig
strConnection = "Provider=" & strProvider & ";Password=" & strPassword & ";Persist Security Info=True;User ID=" & strUsername & ";Initial Catalog=" & strDatabase & ";Data Source=" & strDBServer
acnDatabase.CursorLocation = adUseServer
acnDatabase.Open strConnection
If acnDatabase.State = 1 Then
strSelection = "Select CRC_IDCircuito,CRC_Nombre + ' - ' +
TCT_Nombre,CRC_NivelSeguridad from Circuito join Tipo_Circuito on
Circuito.TCT_IDTipoCircuito=Tipo_Circuito.TCT_IDTipoCircuito order by
CRC_Nombre"
arsCircuitos.Open strSelection, acnDatabase, adOpenStatic, adLockReadOnly
strSelection = "Select USR_Username,USR_Password,USR_NivelSeguridad from Usuario
order by USR_Username"
arsUsers.Open strSelection, acnDatabase, adOpenStatic, adLockReadOnly
End If
cboCircuito.Clear
intIndex = 0
If arsCircuitos.RecordCount > 0 Then ReDim aintCircuitos(arsCircuitos.RecordCount, 2)
While Not arsCircuitos.EOF
```

```
cboCircuito.AddItem arsCircuitos.Fields(1)
aintCircuitos(intIndex, 0) = arsCircuitos("CRC_IDCircuito")
aintCircuitos(intIndex, 1) = arsCircuitos("CRC_NivelSeguridad")
intIndex = intIndex + 1
arsCircuitos.MoveNext
Wend
arsCircuitos.Close
Set arsCircuitos = Nothing
intIndex = 0
If arsUsers.RecordCount > 0 Then ReDim astrUsers(arsUsers.RecordCount, 3)
While Not arsUsers.EOF
astrUsers(intIndex, 0) = arsUsers("USR_Username")
astrUsers(intIndex, 1) = arsUsers("USR_Password")
astrUsers(intIndex, 2) = arsUsers("USR_NivelSeguridad")
intIndex = intIndex + 1
arsUsers.MoveNext
Wend
arsUsers.Close
Set arsUsers = Nothing
acnDatabase.Close
Set acnDatabase = Nothing
If cboCircuito.ListCount > 0 Then cboCircuito.ListIndex = 0
Exit Sub

Err_UserControl_Initialize:
MsgBox Err.Description
End Sub

Private Sub LoadConfig()
On Error GoTo Err_LoadConfig
```

```
Dim lngRegistryHandler As Long, lngResultHandler As Long, lngResultValue As Long, lngKeyType As Long, lngKeySize As Long
Dim strTemporal As String

lngResultHandler = RegOpenKeyEx(HKEY_LOCAL_MACHINE, "SOFTWARE\Sistema Monitoreo", 0, KEY_ALL_ACCESS, lngRegistryHandler)
If lngRegistryHandler = 0 Then
MsgBox "No se puede leer la configuración de conexión a los servidores", vbCritical, "Error de Aplicación"
Else
strTemporal = String$(15, 0)
lngKeySize = 15
lngResultValue = RegQueryValueEx(lngRegistryHandler, "Database", 0, lngKeyType, strTemporal, lngKeySize)
If lngResultValue <> 0 Then
MsgBox "No se puede leer la configuración de conexión a los servidores", vbCritical, "Error de Aplicación"
Else
strDatabase = Left(strTemporal, lngKeySize - 1)
End If
strTemporal = String$(15, 0)
lngKeySize = 15
lngResultValue = RegQueryValueEx(lngRegistryHandler, "DataSocket", 0, lngKeyType, strTemporal, lngKeySize)
If lngResultValue <> 0 Then
MsgBox "No se puede leer la configuración de conexión a los servidores", vbCritical, "Error de Aplicación"
Else
strDataSocket = Left(strTemporal, lngKeySize - 1)
End If
strTemporal = String$(15, 0)
```

```
lngKeySize = 15
lngResultValue = RegQueryValueEx(lngRegistryHandler, "Provider", 0, lngKeyType, strTemporal, lngKeySize)
If lngResultValue <> 0 Then
MsgBox "No se puede leer la configuración de conexión a los servidores", vbCritical, "Error de Aplicación"
Else
strProvider = Left(strTemporal, lngKeySize - 1)
End If
strTemporal = String$(15, 0)
lngKeySize = 15
lngResultValue = RegQueryValueEx(lngRegistryHandler, "DBServer", 0, lngKeyType, strTemporal, lngKeySize)
If lngResultValue <> 0 Then
MsgBox "No se puede leer la configuración de conexión a los servidores", vbCritical, "Error de Aplicación"
Else
strDBServer = Left(strTemporal, lngKeySize - 1)
End If
strTemporal = String$(15, 0)
lngKeySize = 15
lngResultValue = RegQueryValueEx(lngRegistryHandler, "Username", 0, lngKeyType, strTemporal, lngKeySize)
If lngResultValue <> 0 Then
MsgBox "No se puede leer la configuración de conexión a los servidores", vbCritical, "Error de Aplicación"
Else
strUsername = Left(strTemporal, lngKeySize - 1)
End If
strTemporal = String$(15, 0)
```

lngKeySize = 15

lngResultValue = RegQueryValueEx(lngRegistryHandler, "Password", 0, lngKeyType, strTemporal, lngKeySize)

If lngResultValue <> 0 Then

MsgBox "No se puede leer la configuración de conexión a los servidores", vbCritical, "Error de Aplicación"

Else

strPassword = Left(strTemporal, lngKeySize - 1)

End If

End If

Exit Sub

Err_LoadConfig:

MsgBox Err.Description

End Sub

Código Visual Basic del Control OCX de Monitoreo

Una vez compilado correctamente el código se genera el control ocx, luego éste se encuentra listo para utilizarse en otra aplicación que soporte controles ActiveX como las

páginas web para Microsoft Internet Explorer de la aplicación de cliente. En la figura 3.108 podemos visualizar el control ActiveX sobre una pagina web para que funcione en un explorador web, mejorar la interfase para el usuario y conservar los estándares de diseño de la aplicación. Aquí se muestra el control sin conexión al servidor de Datasocket, por lo tanto se visualiza la pantalla de gráficos sin ondas.

Figura 3.108 – Página Web de Mediciones en Tiempo Real sin Conexión

Como se puede visualizar en la figura 3.108 todo se encuentra apagado y deshabilitado, debido a que cuando se carga la interfase de cliente no es conveniente que se realicen todas las tareas de conexión al servidor porque tomará mayor tiempo en que se termine de cargar totalmente la interfase y se encuentre disponible y lista para la tarea de monitoreo de las ondas de un circuito trifásico.

El código posee las verificaciones necesarias para el caso de que el servidor de DataSocket no se encuentre levantado y aceptando conexiones, en la figura 3.109 se muestra el mensaje de error que nos indica la ocurrencia de este hecho y por lo tanto la imposibilidad de funcionamiento del monitoreo de ondas en tiempo real.

Figura 3.109 – Mensaje de Error de Conexión al Servidor de DataSocket

Una vez que la conexión al servidor de DataSocket ha sido realizada con éxito se encienden los indicadores de error, simulación, adquisición de datos, la cajas de texto que indican el estado de las conexiones de escritura hacia LabVIEW 6i: el selector de circuito y el habilitador de ondas; finalmente se activa la lista desplegable que contiene una lista de todos los circuitos registrados en el Sistema de Monitoreo. En la figura 3.110 se visualiza la interfase de cliente conectada al servidor de DataSocket.

Figura 3.110 – Página Web de Mediciones en Tiempo Real con Conexión sin Formas de Onda

Al poco tiempo de la conexión exitosa con el servidor de DataSocket se empiezan a recibir los datos de voltajes y corrientes, potencias, factor de potencia y frecuencia del circuito monitoreado que se encuentra seleccionado. En la figura 3.111 se puede visualizar a la interfase de monitoreo con los datos de voltajes y corrientes, potencias, factor de potencia y frecuencia del circuito seleccionado; así como los indicadores de de error en las tarjetas DAQ, adquisición de datos y simulación. Sin embargo no existen todavía gráficas de las formas de onda porque no está habilitada la transmisión de los datos de formas de onda desde LabVIEW.

En la figura 3.112 se puede visualizar que ya están dibujadas las formas de ondas de onda del circuito seleccionado, en este caso un trifásico de tres hilos. Esto se consigue al pulsar el botón de Habilitación de Ondas que permite que LabVIEW envíe los datos al servidor de DataSocket y a su vez la interfase de cliente grafica las formas de onda apenas recibe los datos del mismo.

En la figura 3.113 se puede visualizar las formas de onda de un circuito trifásico de cuatro hilos con sus respectivos parámetros eléctricos. Finalmente en la figura 3.114 se tiene las formas de onda de un circuito trifásico balanceado de tres hilos.

Figura 3.111 – Página Web de Mediciones en Tiempo Real con Conexión sin Formas de Onda

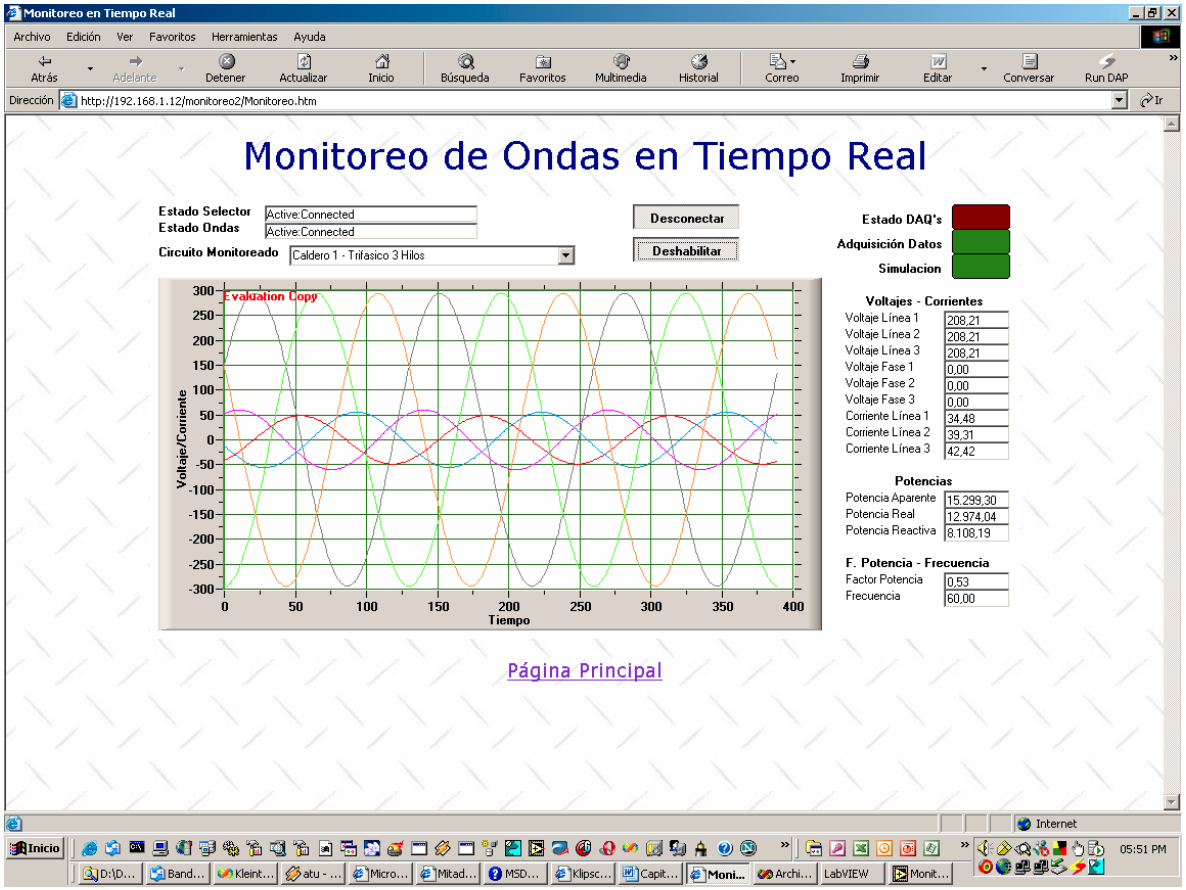

Figura 3.112 – Página Web de Mediciones en Tiempo Real con Conexión con Formas de Onda de Circuito Trifásico de 3 Hilos

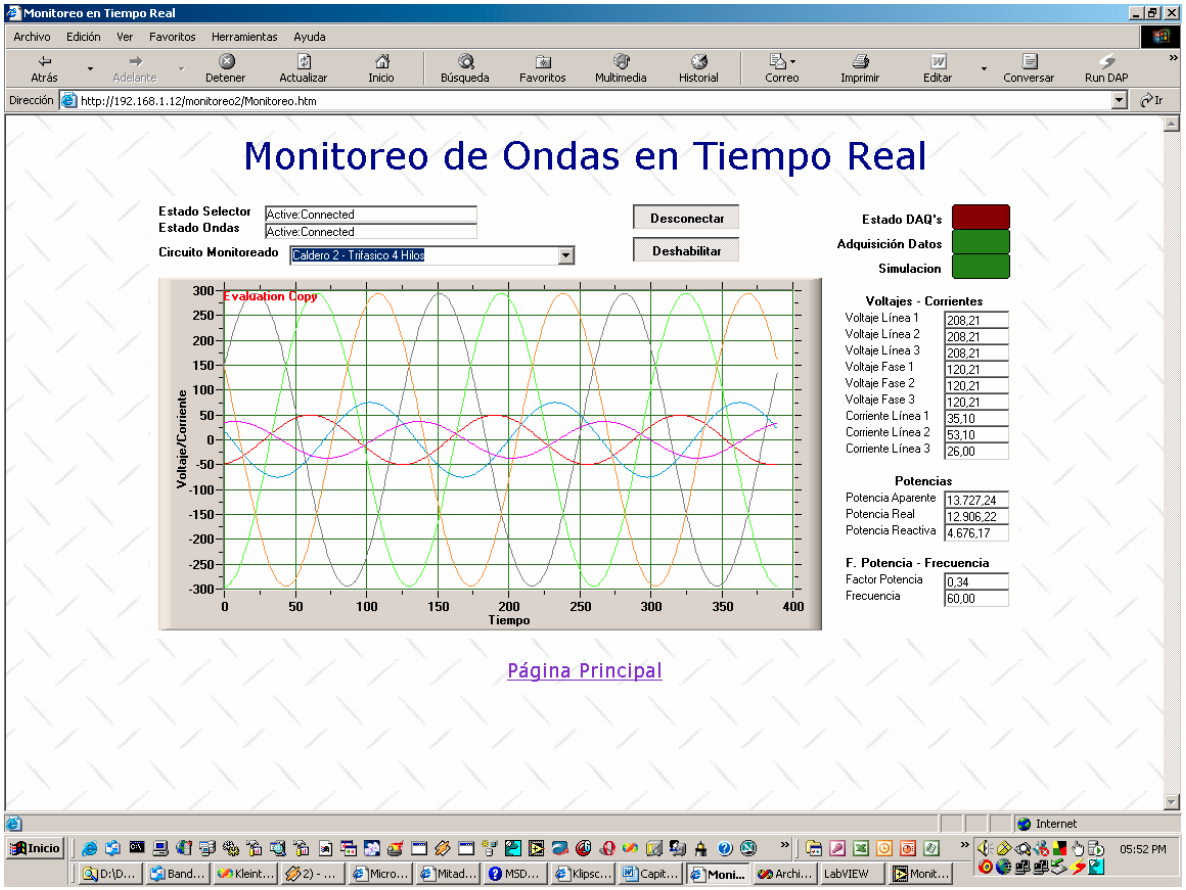

Figura 3.113 – Página Web de Mediciones en Tiempo Real con Conexión con Formas de Onda de Circuito Trifásico de 4 Hilos

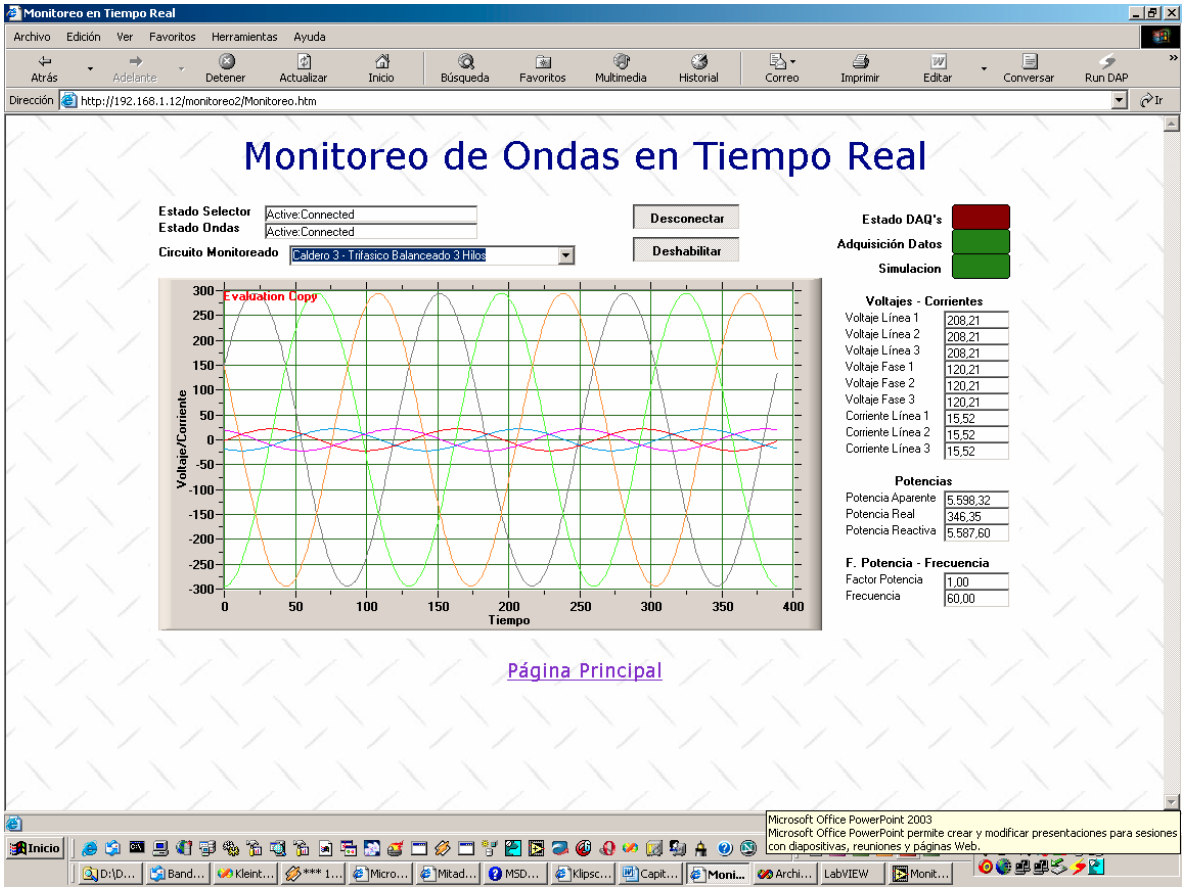

Figura 3.114 – Página Web de Mediciones en Tiempo Real con Conexión con Formas de Onda de Circuito Trifásico Balanceado de 3 Hilos

Cabe anotar que LabVIEW posee un servidor web integrado llamado Servidor Web G, que nos permite la publicación de imágenes estáticas de los paneles con los instrumentos virtuales de la aplicación; sin embargo en la versión 6i, que utilizamos para el desarrollo de la aplicación hasta la versión 6.1 sabemos que permite cierta animación que funciona solamente en Netscape Navigator en las versiones 4.x, no siendo así con las versiones 6.x y 7.x del navegador.

El navegador de internet predeterminado que viene con Windows es el Microsoft Internet Explorer que tiene otro tipo de funcionamiento, toda la página web se actualiza constantemente hasta que el usuario presione el botón de detener, sin embargo éste no es el comportamiento deseado ni el método más óptimo y adecuado, debido a las condiciones de enlaces de red remotos que no posean banda ancha.

Considerando estas limitadas capacidades del servidor web de LabVIEW, no será utilizado para el desarrollo de la interfase de cliente de monitoreo de ondas en tiempo real, pues se necesita un poco más de interacción entre el usuario y la aplicación y; principalmente que la interfase cumpla su cometido que es la graficación de formas de onda y el despliegue de valores de los parámetros eléctricos en tiempo real.

3.6.2.3. Página de Estadísticas Históricas de Circuitos

Esta es una página web de servidor dinámica que contiene elementos para acceso a la DBMS y a sus datos. Para mantener la seguridad y transparencia del sistema para el usuario, se ha creado un componente que accede a la base de datos que soporta la aplicación de servidor en LabVIEW y que se ejecuta en el servidor de aplicaciones mas no en la computadora del cliente. Este componente también utiliza la tecnología de OLEDB con los métodos de ADO para acceder a los datos en una forma rápida y segura. Esta página dinámica primero obtiene las configuraciones de conexión a la DBMS en forma interna e indica cualquier anomalía de conexión a la misma. Cuando la conexión con la DBMS es exitosa aparece la pantalla para inicio de sesión, donde se debe ingresar un nombre de usuario y una contraseña para ser verificada. En la figura 3.114 se puede visualizar la pantalla de inicio de sesión de usuario. Si el nombre de usuario o la contraseña han sido erróneos el sistema indica este hecho mediante un mensaje. En la figura 3.115 se visualiza el mensaje devuelto por el sistema cuando el usuario ha fallado en su intento de ingreso al sistema. Una vez que el usuario ha sido verificado correctamente ingresa al sistema y procede a seleccionar el área y/o circuito del cual desea visualizar los datos con los respectivos rangos de fecha para finalmente desplegar los datos históricos requeridos. En la figura 3.116 se muestra la pantalla de selección del área y/o circuito con las cajas de texto para ingreso de fechas, así como el botón consultar para desplegar los datos existentes.

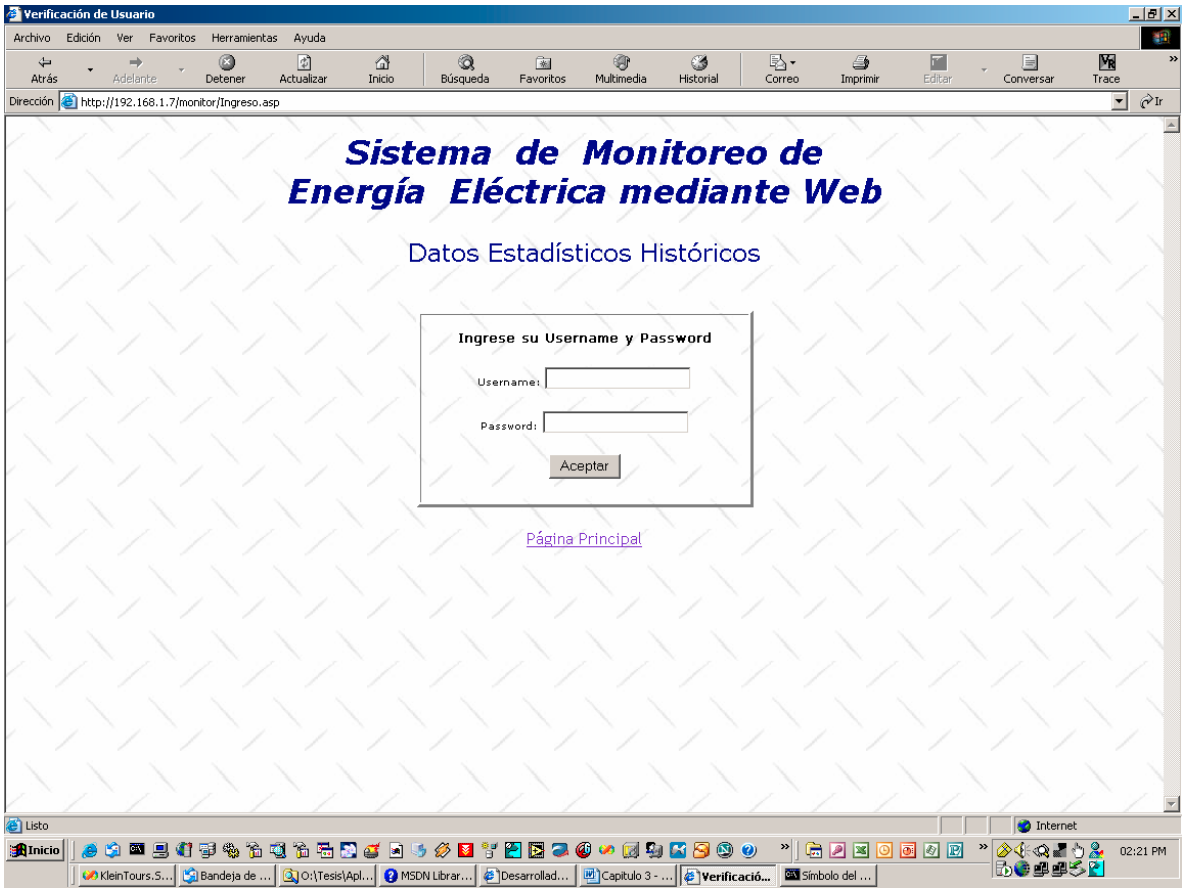

Figura 3.115 – Página Web de Datos Históricos - Ingreso de Usuario

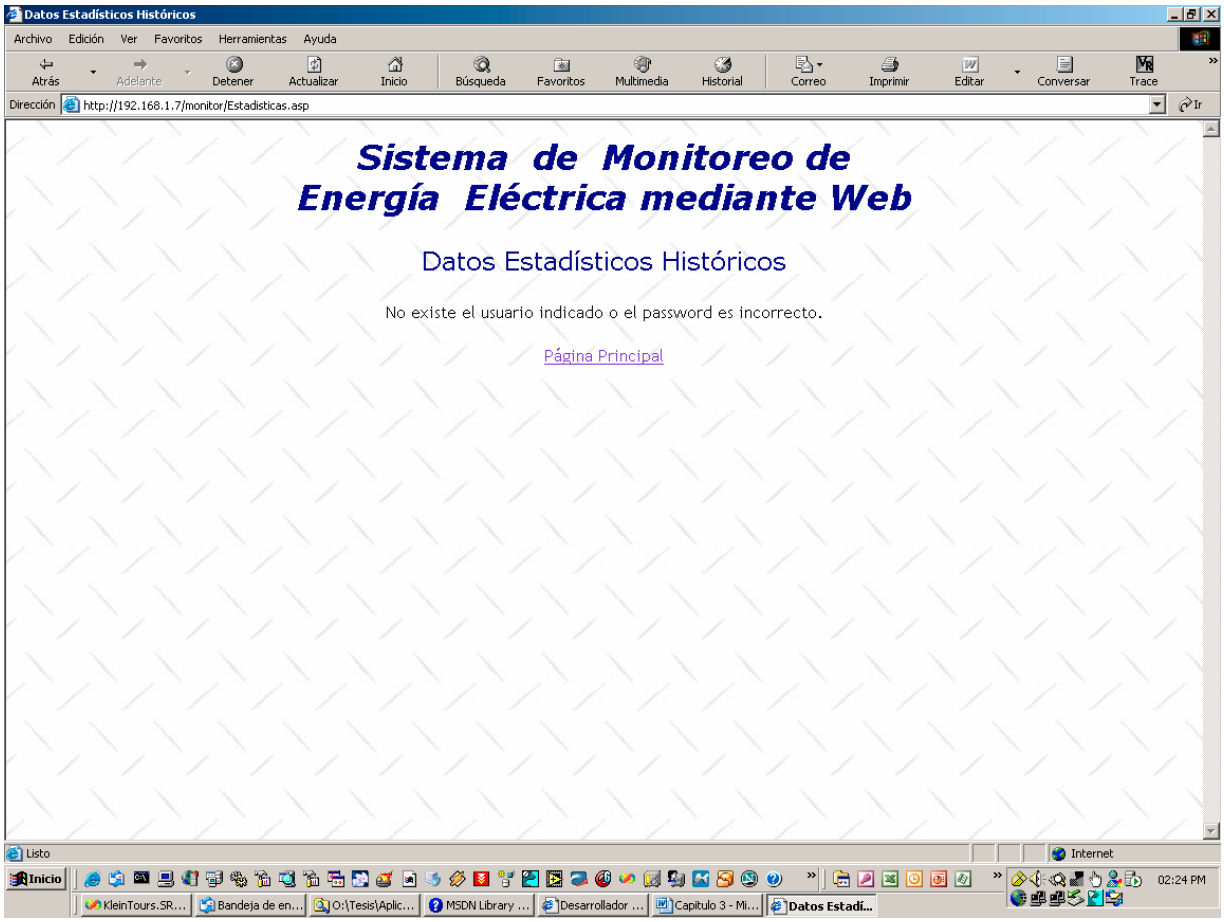

Figura 3.116 – Página Web de Datos Históricos - Ingreso de Usuario Incorrecto

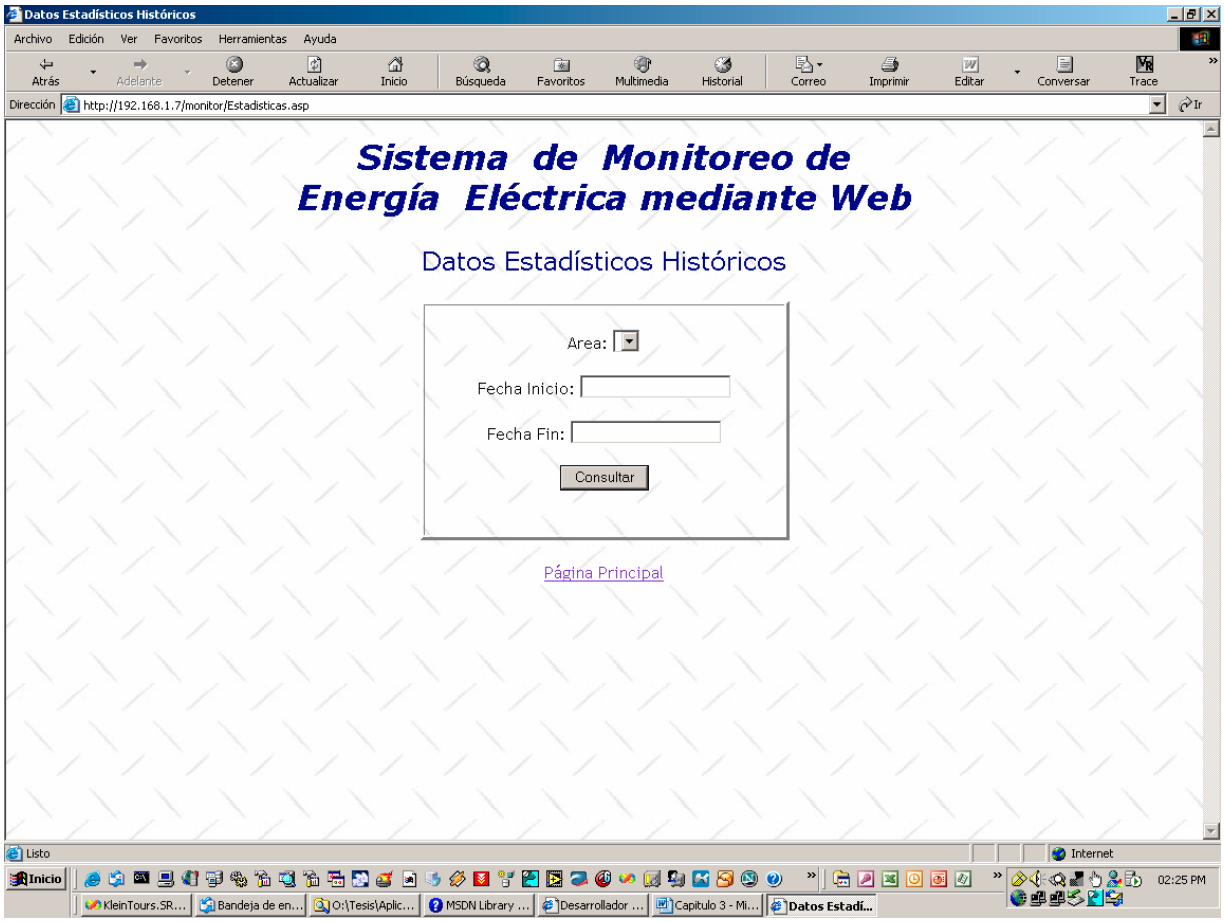

Figura 3.117 – Página Web de Datos Históricos - Ingreso de Usuario Correcto – Selección de Area y/o Circuito en ingreso de Fechas

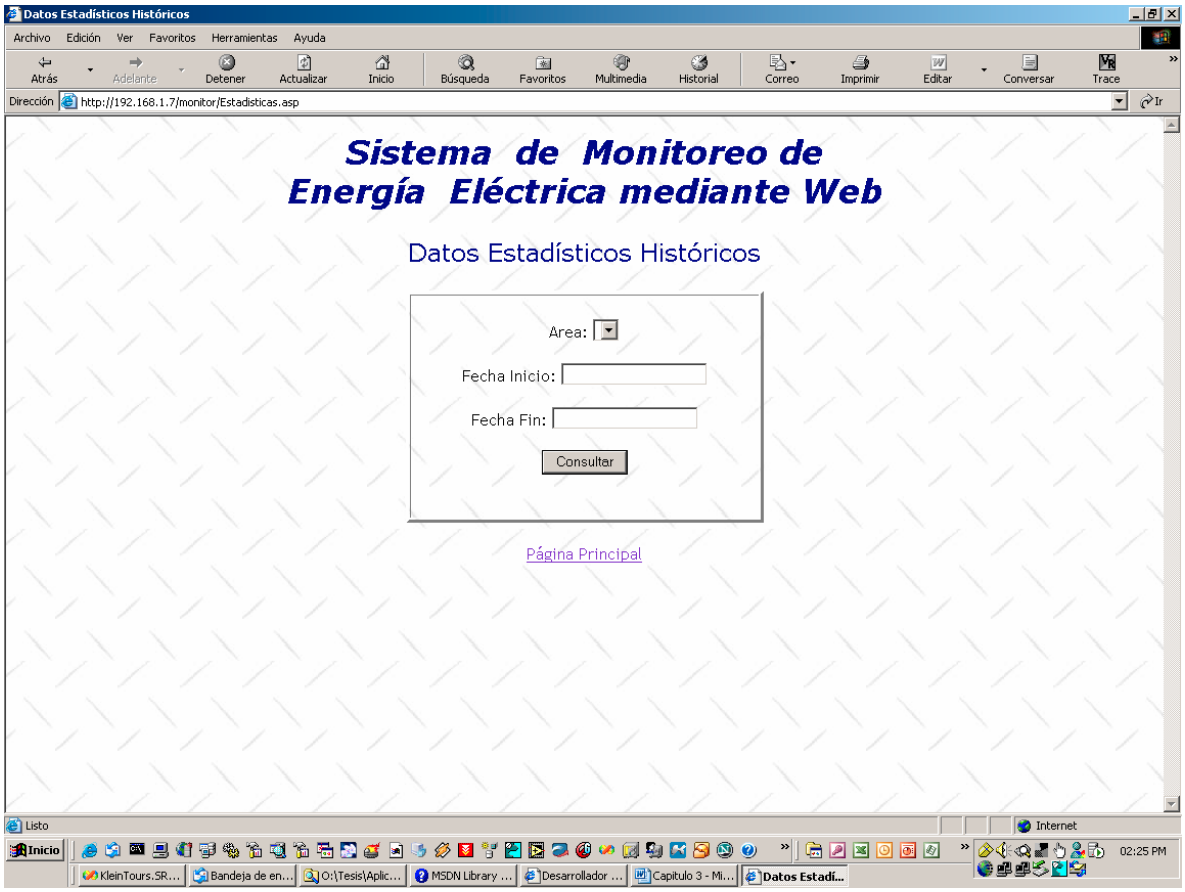

Figura 3.118 – Página Web de Datos Históricos – Datos Consultados en la DBMS

Una vez ingresados los parámetros se realiza la consulta a la DBMS y se despliegan los datos de los parámetros eléctricos medidos y los calculados como las potencias, factor de potencia y las energías. En la figura 3.117 se muestra un ejemplo de los datos obtenidos de la consulta a la DBMS para un área específica entre un rango de fechas determinado.

El componente de datos está realizado totalmente en Visual Basic 6.0 con los métodos de ADO. Como este código debe mantenerse oculto para el usuario se ha realizado una biblioteca de enlace dinámico que se la utiliza en la página web dinámica, la misma sólo expone sus métodos con sus parámetros y el tipo de resultado que envía, lo cual es suficiente para poder utilizarla. Además, esta biblioteca se ejecuta en el servidor de aplicaciones, no es transportada hacia el cliente, brindándonos un nivel de seguridad adicional.

A continuación se muestra el código de la librería que realiza el acceso a los datos históricos de mediciones en la DBMS. Esta librería de enlace dinámico debe registrarse en el servidor donde se encuentre el servidor web para que pueda funcionar adecuadamente, esto se realiza mediante el comando: *regsvr32 librería.dll.*

Option Explicit

Dim strProvider As String, strPassword As String, strUsername As String, strDBServer As String, strDatabase As String

Dim strDataSocket As String, acnDatabase As ADODB.Connection, bolLoadConfig As Boolean

Const HKEY_LOCAL_MACHINE = &H80000002

Const REG_SZ = 1

Const REG_DWORD = 4

Const KEY_READ = &H20000

Const KEY_QUERY_VALUE = &H1

Const KEY_SET_VALUE = &H2

Const KEY_CREATE_SUB_KEY = &H4

Const KEY_ENUMERATE_SUB_KEYS = &H8

Const KEY_NOTIFY = &H10

Const KEY_CREATE_LINK = &H20

Const KEY_ALL_ACCESS = KEY_QUERY_VALUE + KEY_SET_VALUE + KEY_CREATE_SUB_KEY + KEY_ENUMERATE_SUB_KEYS + _

KEY_NOTIFY + KEY_CREATE_LINK + KEY_READ

Private Declare Function RegOpenKeyEx Lib "advapi32" Alias "RegOpenKeyExA" (ByVal hKey As Long, ByVal lpSubKey As String, ByVal ulOptions As Long, ByVal samDesired As Long, ByRef phkResult As Long) As Long

Private Declare Function RegQueryValueEx Lib "advapi32" Alias "RegQueryValueExA" (ByVal hKey As Long, ByVal lpValueName As String, ByVal lpReserved As Long, ByRef lpType As Long, ByVal lpData As String, ByRef lpcbData As Long) As Long

Private Declare Function RegCloseKey Lib "advapi32" (ByVal hKey As Long) As Long

```
Private Function LoadConfig() As String
On Error GoTo Err_LoadConfig


Dim lngRegistryHandler As Long, lngResultHandler As Long, lngResultValue As Long, lngKeyType As Long, lngKeySize As Long
Dim strTemporal As String

lngResultHandler = RegOpenKeyEx(HKEY_LOCAL_MACHINE, "SOFTWARE\Sistema Monitoreo", 0, KEY_READ + KEY_ENUMERATE_SUB_KEYS + KEY_QUERY_VALUE, lngRegistryHandler)
If lngRegistryHandler = 0 Then
bolLoadConfig = False
LoadConfig = "No se puede leer la configuracion de conexion a los servidores."
Else
strTemporal = String$(15, 0)
lngKeySize = 15
lngResultValue = RegQueryValueEx(lngRegistryHandler, "Database", 0, lngKeyType, strTemporal, lngKeySize)
If lngResultValue <> 0 Then
LoadConfig = "No se puede leer los parametros de configuracion de conexion a los servidores."
Else
strDatabase = Left(strTemporal, lngKeySize - 1)
End If
strTemporal = String$(15, 0)
lngKeySize = 15
lngResultValue = RegQueryValueEx(lngRegistryHandler, "DataSocket", 0, lngKeyType, strTemporal, lngKeySize)
If lngResultValue <> 0 Then
```

LoadConfig = "No se puede leer los parametros de configuracion de conexion a los servidores."

Else

strDataSocket = Left(strTemporal, lngKeySize - 1)

End If

strTemporal = String$(15, 0)

lngKeySize = 15

lngResultValue = RegQueryValueEx(lngRegistryHandler, "Provider", 0, lngKeyType, strTemporal, lngKeySize)

If lngResultValue <> 0 Then

LoadConfig = "No se puede leer los parametros de configuracion de conexion a los servidores."

Else

strProvider = Left(strTemporal, lngKeySize - 1)

End If

strTemporal = String$(15, 0)

lngKeySize = 15

lngResultValue = RegQueryValueEx(lngRegistryHandler, "DBServer", 0, lngKeyType, strTemporal, lngKeySize)

If lngResultValue <> 0 Then

LoadConfig = "No se puede leer los parametros de configuracion de conexion a los servidores."

Else

strDBServer = Left(strTemporal, lngKeySize - 1)

End If

strTemporal = String$(15, 0)

lngKeySize = 15

lngResultValue = RegQueryValueEx(lngRegistryHandler, "Username", 0, lngKeyType, strTemporal, lngKeySize)

If lngResultValue <> 0 Then

LoadConfig = "No se puede leer los parametros de configuracion de conexion a los servidores."

Else

strUsername = Left(strTemporal, lngKeySize - 1)

End If

strTemporal = String$(15, 0)

lngKeySize = 15

lngResultValue = RegQueryValueEx(lngRegistryHandler, "Password", 0, lngKeyType, strTemporal, lngKeySize)

If lngResultValue <> 0 Then

LoadConfig = "No se puede leer los parametros de configuracion de conexion a los servidores."

Else

strPassword = Left(strTemporal, lngKeySize - 1)

End If

bolLoadConfig = True

RegCloseKey (lngRegistryHandler)

End If

Exit Function

Err_LoadConfig:

bolLoadConfig = False

LoadConfig = Err.Description

End Function

Private Function DBConnect() As String

On Error GoTo Err_DBConnect

Dim strConnection As String, strConfigError As String

Set acnDatabase = New ADODB.Connection

If bolLoadConfig = False Then strConfigError = LoadConfig

If bolLoadConfig = True Then

```
strConnection = "Provider=" & strProvider & ";Password=" & strPassword & ";Persist Security Info=True;User ID=" & strUsername & ";Initial Catalog=" & strDatabase & ";Data Source=" & strDBServer
acnDatabase.CursorLocation = adUseServer
acnDatabase.Open strConnection
DBConnect = acnDatabase.State
Else
DBConnect = strConfigError
End If
Exit Function

Err_DBConnect:
DBConnect = Err.Description
End Function

Private Function DBDisconnect() As String
On Error GoTo Err_DBDisconnect

acnDatabase.Close
Set acnDatabase = Nothing
DBDisconnect = acnDatabase.State
Exit Function

Err_DBDisconnect:
DBDisconnect = Err.Description
End Function

Public Function RecordCount(Table As String, Optional Conditions As String) As String
On Error GoTo Err_RecordCount
```

```
Dim strDBstatus As String, strSelection As String, arsRecords As ADODB.Recordset

strDBstatus = DBConnect
If strDBstatus = "1" Then
strSelection = "Select * from " & Table
If Conditions <> "" Then
strSelection = strSelection & " where " & Conditions
End If
Set arsRecords = New ADODB.Recordset
arsRecords.Open strSelection, acnDatabase, adOpenStatic, adLockReadOnly
End If
RecordCount = arsRecords.RecordCount
arsRecords.Close
Set arsRecords = Nothing
DBDisconnect
Exit Function

Err_RecordCount:
RecordCount = Err.Description
End Function

Public Function Retrieve(Columns As String, Table As String, Optional Conditions As
String, Optional Ordering As String) As Variant
On Error GoTo Err_Retrieve

Dim strDBstatus As String, strSelection As String, arsRecords As ADODB.Recordset

strDBstatus = DBConnect
If strDBstatus = "1" Then
strSelection = "Select " & Columns & " from " & Table
If Conditions <> "" Then
```

```
strSelection = strSelection & " where " & Conditions
End If
If Ordering <> "" Then
strSelection = strSelection & " order by " & Ordering
End If
Set arsRecords = New ADODB.Recordset
arsRecords.Open strSelection, acnDatabase, adOpenStatic, adLockReadOnly
Retrieve = arsRecords.GetRows
End If
Set arsRecords = Nothing
DBDisconnect
Exit Function

Err_Retrieve:
Retrieve = Null
End Function

Public Function Login(Username As String, Password As String) As Variant
On Error GoTo Err_Login

Login = Retrieve("*", "Usuario", "usr_Username=" & Username & " and usr_Password=" & Password)
Exit Function

Err_Login:
Login = Null
End Function

Public Function RetrieveAreas() As Variant
On Error GoTo Err_RetrieveAreas
```

```
RetrieveAreas = Retrieve("are_Nombre,are_IDArea,are_NivelSeguridad", "Area", , "are_Nombre")
Exit Function

Err_RetrieveAreas:
RetrieveAreas = Null
End Function

Public Function RetrieveMeasurement(IDCircuito As String, FechaInicio As String, Optional FechaFin As String) As Variant
On Error GoTo Err_RetrieveMeasurement

Dim strFechas As String

FechaInicio = Mid(FechaInicio, 2, Len(FechaInicio) - 2)
strFechas = "cmd_Fecha between '" & Format(FechaInicio, "mm/dd/yyyy") & "' and '"
If FechaFin <> "" Then
FechaFin = Mid(FechaFin, 2, Len(FechaFin) - 2)
strFechas = strFechas & Format(FechaFin, "mm/dd/yyyy") & "'"
Else
strFechas = strFechas & Format(Date, "mm/dd/yyyy") & "'"
End If
RetrieveMeasurement = Retrieve("*", "Circuito_Medicion", strFechas)
Exit Function

Err_RetrieveMeasurement:
RetrieveMeasurement = Null
RetrieveMeasurement = strFechas & vbCrLf & Err.Description
End Function
```

Código en Visual Basic de la Librería de Acceso a Datos para el Servidor

Una vez compilada y registrada la librería en el servidor se debe realizar la programación de las páginas web dinámicas donde se utiliza la misma. A continuación se muestra un ejemplo de uso de la librería en la página de verificación de usuario para ingreso al sistema.

```
<html>
<TITLE>Verificación de Usuario</TITLE>
<head></head>
<body bgcolor="white" background="indtextb.jpg" lang="ES" link="#3366cc" vlink="#666666" style="tab-interval: 35.4pt"><!--[if gte mso 9]><xml> <v:background id="_x0000_s1027" o:bwmode="white" o:targetscreensize="800,600"> <v:fill src="./Introduccion_archivos/image001.jpg" o:title="indtextb" type="frame"/> </v:background></xml><![endif]-->
<DIV align=center><FONT face=Verdana color=darkblue size=6>
<P align=center><STRONG><FONT color=darkblue><FONT face=Verdana
size=6><EM>Sistema  de  Monitoreo de <BR>Energía  Eléctrica
mediante Web</EM></FONT> </FONT></STRONG></P><FONT size=5>Datos Estadísticos
Históricos</FONT> </FONT><FONT size=5> </FONT> </DIV>
<DIV align=center> </DIV>
<DIV align=center>
<form action=Ingreso.asp method=post id=form1 name=form1></DIV>
<P align=center><TABLE WIDTH=330 ALIGN=center BORDER=3 CELLSPACING=1 CELLPADDING=1>
<TR><TD>
<P align=center><FONT size=1><FONT face="Verdana Ref"><STRONG><BR><FONT size=2>Ingrese su
Username y Password</SPAN></FONT></STRONG><FONT size=2> <o:p></O:P></FONT></FONT></FONT></P>
<P></P>
```

```
<P align=center><FONT face="Verdana Ref" size=1>Username: </FONT><INPUT
NAME="Username" id=txtUsername></P><o:p></O:P>
<P align=center><FONT face="Verdana Ref" size=1>Password:</FONT>
</FONT><INPUT NAME="Password" id=txtCondicion type=password>
<o:p></O:P></P>
<p></p>
<P align=center><INPUT TYPE="submit" ACTION="Ingreso.asp"
VALUE="Aceptar" METHOD="post" NAME="Aceptar"
id=btnAceptar><BR><BR></P></TD></TD></TR></TABLE></P></FORM>
<%Username=Request.Form("Username")
Response.Write(Username)
Password=Request.Form("Password")
Response.Write(Password)
set Objeto = CreateObject("Database.DBOperations")
if Username<>"" then
usuarios=objeto.Login("'" & Username & "'","'" & Password & "'")
if IsArray(usuarios) then
session("UserID")=usuarios(0,0)
session("Security")=usuarios(3,0)
session("Logged")="Yes"
Response.Redirect("Estadisticas.asp")
else
session("UserID")=""
session("Security")=""
session("Logged")="No"
Response.Redirect("Estadisticas.asp")
end if
end if
%>
<P align=center><A href="index.htm"><FONT color=darkorchid
face="Vendana">Página Principal</A></P></FONT>
```

```
</body>
</html>
```

Código en HTML y Visual Basic Script de la Página Web de Ingreso al Sistema

3.7. IMPLANTACIÓN DEL SISTEMA DE MONITOREO

La implantación del sistema de monitoreo se divide en dos etapas:

- Implantación de la DBMS y aplicación servidor
- Implantación del servidor de aplicaciones y aplicación cliente

3.7.1. Implantación de la DBMS y Aplicación Servidor

Primero se debe instalar la DBMS seleccionada por el usuario del sistema. El sistema de monitoreo actual está implementado con SQL Server 2000, luego se debe enlazar los archivos de datos al sistema administrador de base de datos para disponer de los mismos en las aplicaciones de servidor y cliente.

A continuación e debe instalar también LabVIEW 6i para poder ejecutar la aplicación del servidor o instalar la aplicación compilada para evitar usar el lenguaje de desarrollo y brindar mayor seguridad al sistema, así como también ahorrar en los costos de licenciamiento.

Finalmente se debe instalar la librería de acceso a datos, MDAC (Microsoft Data Access Componentes) versión 2.8, para que LabVIEW pueda tener acceso y comunicación con la DBMS. De otro modo no se podrá iniciar la aplicación debido a que las configuraciones de circuitos, usuarios y demás se encuentran almacenados en la misma.

Al iniciar la aplicación por primera vez no se encuentra registrado ningún usuario, por lo tanto se pedirá una contraseña para el usuario Administrador, luego permitirá definir las características de la planta como: áreas, circuitos por área y puntos de medición por circuito; finalmente se definirán los usuarios con su nivel de seguridad para acceso al sistema. La aplicación de servidor se encuentra lista para entrar en funcionamiento y comenzar la adquisición de datos.

La aplicación puede volver a ejecutarse pero siempre pedirá un nombre de usuario y una contraseña para que el sistema se asegure de que el usuario ingresado es el que está utilizando el sistema.

3.7.2. Implantación del Servidor de Aplicaciones y Aplicación Cliente

Primero se debe instalar un servidor web que soporte páginas activas de servidor (.asp), luego crear un sitio virtual para poder acceder desde cualquier computador y fijar las adecuadas seguridades de directorios para evitar problemas con la aplicación y la información. En ese sitio virtual se debe colocar las páginas de la aplicación (.html y .asp), el componente ActiveX de monitoreo (.ocx) y las librerías dinámicas de acceso a la DBMS (.dll). A continuación se procede a registrar en el servidor los componentes ActiveX y las librerías para que el servidor de aplicaciones las reconozca, pueda inicializarlas y hacer disponibles sus funciones y métodos para la aplicación cliente.

Finalmente se debe instalar Microsoft Internet Explorer desde la versión 5.0 en adelante en todas las computadoras que necesiten utilizar la aplicación cliente. Debe estar habilitada la ejecución de controles ActiveX para poder visualizar el monitoreo de ondas en tiempo real.

Para utilizar la aplicación cliente se abrirá Microsoft Internet Explorer y se colocará en la línea de dirección la cadena correspondiente a la aplicación utilizando el siguiente

formato: http://nombre completo del dominio o equipo:puerto/directorio virtual/página web; por ejemplo: (http://www.CerveceriaAndina.com/SistemaMonitoreo/index.htm).

En este caso se ha omitido el número del puerto porque se usa el predeterminado que es el 80 para el protocolo de transferencia de hipertexto (HTTP), sin embargo se puede utilizar otro puerto configurando en el servidor web para brindar otro punto adicional de seguridad, ya que si no se sabe a que puerto conectarse no se cargarán las páginas de la aplicación.

CAPÍTULO 4

PRUEBAS DEL SISTEMA

4. PRUEBAS DEL SISTEMA

Primero se realizará una descripción de las condiciones y equipos que se utilizaron en las pruebas del sistema. Como equipo servidor de web y base de datos se ha utilizado un computador con las siguientes características:

- Procesador Pentium 4 de 2.4 GHz (533MHz FSB)
- 512 MB memoria RAM DDR 266 MHz
- 80 GB disco duro (5400 RPM – modo ATA 100)
- 128 MB memoria RAM DDR de video a 1600*1200
- Sistema Operativo Windows 2000 Professional
- Tarjeta de red 3Com PCI a 100 Mbps

Para la realización de las pruebas se ha utilizado la tarjeta de red 3Com PCI a velocidades de 100 Mbps y 10 Mbps para un entorno de red de área local y; una conexión serial de 56 Kbps y 128 Kbps para simular un entorno de red de área extendida remota.

Con respecto a las DBMS’s, se tienen diferentes instaladas en el sistema operativo y entre las cuales cuentan las siguientes:

- Microsoft SQL Server 2000

- Oracle Database 9i
- Sybase Adaptive Server Enterprise 12.5
- IBM DB2 7.1
- MySQL 4.1.2

4.1. PRUEBAS DE ALMACENAMIENTO Y RECUPERACION DE DATOS

En estas pruebas se están considerando los tiempos de conexión a las DBMS`s, tiempos de recuperación de los datos de configuración de puntos de medición y tiempos de almacenamiento de los datos procesados a la DBMS; todo esto sobre un entorno de tipo local para obtener velocidades de respuesta de las DBMS`s y por tanto datos estadísticos para crear parámetros para posteriores recomendaciones de implementaciones para los futuros usuarios del sistema de monitoreo.

A continuación se realizará una explicación detallada de cada prueba que se ejecutará con las bases de datos para obtener los tiempos de respuesta:

- Conexión al Sistema en LabVIEW para registrarse y comenzar la adquisición (local)
- Almacenamiento de configuración de áreas, circuitos y puntos de adquisición (local)
- Recuperación de configuración de áreas, circuitos y puntos de adquisición con preparación del sistema (local)
- Recuperación de datos estadísticos en cliente web (local y remoto)

4.1.1. Conexión al Sistema en LabVIEW

Esta prueba mide el tiempo en milisegundos desde la lectura del registro de Windows con la configuración de la DBMS y el servidor de DataSocket hasta el momento en que la conexión está completada y se procede a la verificación de usuario. La prueba se realizará con las diferentes DBMS’s para medir los tiempos de respuesta de conexión inicial.

Microsoft SQL Server 2000
Tiempo medio: 1100 ms

Oracle Database 9i
Tiempo medio: 1800 ms

Sybase Adaptive Server 12.5
Tiempo medio: 1250 ms

IBM DB2 7.1
Tiempo medio 1350 ms

MySQL 4.1.2
Tiempo medio 1300 ms

La DBMS que mejor respuesta tiene en conexión es Microsoft SQL Server 2000, debido a que se cuenta con los drivers de OLEDB y es la más integrada con Windows.

4.1.2. Almacenamiento de Configuración de Circuitos Trifásicos

Esta prueba mide el tiempo en milisegundos para las operaciones de inserción, eliminación y actualización para cada uno de los casos de áreas, circuitos y puntos de medición. La prueba se realizará con las diferentes DBMS’s para medir los tiempos de respuesta de las operaciones comunes de manipulación con registros

Microsoft SQL Server 2000
Tiempo medio: 100 ms

Oracle Database 9i
Tiempo medio: 200 ms

Sybase Adaptive Server 12.5

Tiempo medio: 180 ms

IBM DB2 7.1
Tiempo medio 150 ms

MySQL 4.1.2
Tiempo medio 130 ms

La DBMS que mejor respuesta tiene en tiempo de almacenamiento de datos la configuración es Microsoft SQL Server 2000 con 100 ms.por sus drivers de OLEDB.

4.1.3. Recuperación de Configuración de Circuitos Trifásicos

Esta prueba mide el tiempo en milisegundos para las operaciones de recuperación de registros de configuración de cada uno de los puntos de medición hasta cuando la aplicación se encuentra lista para empezar las operaciones de adquisición de datos. La prueba se realizará con las diferentes DBMS's para medir los tiempos de respuesta de las operaciones comunes de manipulación con registros

Microsoft SQL Server 2000
Tiempo medio: 120 ms

Oracle Database 9i
Tiempo medio: 250 ms

Sybase Adaptive Server 12.5
Tiempo medio: 200 ms

IBM DB2 7.1
Tiempo medio 160 ms

MySQL 4.1.2
Tiempo medio 140 ms

La DBMS que mejor respuesta tiene en tiempo de recuperación de datos de la configuración es Microsoft SQL Server 2000 con 120 ms, por sus drivers de OLEDB.

4.1.4. Recuperación de Datos Estadísticos en Cliente Web

Esta prueba mide el tiempo en milisegundos para las operaciones de recuperación de registros de datos estadísticos de cada uno de los circuitos trifásicos especificados en la base de datos de la aplicación. La prueba se realizará con las diferentes DBMS's para medir los tiempos de respuesta de las operaciones de recuperación de registros con parámetros de filtrado. Esta prueba se realizará en forma local y también en forma remota mediante redes WAN de 56 Kbps y 128 Kbps y LAN de 10 Mbps y 100 Mbps respectivamente.

Microsoft SQL Server 2000
Tiempo medio: xxx ms

Oracle Database 9i
Tiempo medio: xxx ms

Sybase Adaptive Server 12.5
Tiempo medio: xxx ms

IBM DB2 7.1
Tiempo medio xxx ms

MySQL 4.1.2
Tiempo medio xxx ms

La DBMS que mejor respuesta tiene en conexión es xxxxxxxxxxxx.

4.2. PRUEBAS DE INTERFASES WEB EN TIEMPO REAL

En estas pruebas se están considerando los tiempos de conexión al Servidor de DataSocket así como a las DBMS`s. El servidor de DataSocket es el que sirve para la transferencia de datos en tiempo real en redes basadas en TCP/IP y; la DBMS es la que almacena los datos históricos que se podrán utilizar para la toma de decisiones.

Se medirán los tiempos de recuperación de los datos de configuración de puntos de medición así como los tiempos de recuperación de los datos procesados y almacenados en la DBMS sobre un entorno de tipo local y remoto, el cliente para realizar las pruebas mencionadas será un explorador web. El monitoreo de las ondas de corriente y voltaje en tiempo real de un circuito específico y los datos en valores RMS de todos los circuitos también se visualizarán mediante un explorador de web sobre un entorno local y remoto, aquí se realizarán mediciones de los tiempos de respuesta en los diferentes tipos de conexión y velocidades de transmisión de red para la obtención de datos estadísticos de tiempo, y así poder crear parámetros de desempeño y calidad para posteriores recomendaciones de implementaciones a los futuros usuarios del sistema de monitoreo.

A continuación se realizará una explicación detallada de cada prueba que se ejecutará con las bases de datos y el servidor de DataSocket para obtener los tiempos de respuesta:

- Visualización de Ondas de Voltaje y Corriente en tiempo real de un circuito(local y remoto)
- Visualización de valores RMS de voltaje, corriente, potencias y factor de potencia de todos los circuitos (local y remoto)

4.2.1. Visualización de Ondas de Voltaje y Corriente en Tiempo Real

Esta prueba trata de medir el tiempo en milisegundos de conexión y actualización del servidor de DataSocket hacia los clientes y viceversa, para permitir la visualización de las ondas de voltaje y corriente para cada uno de los puntos que posee un circuito trifásico de todos los circuitos ingresados en el sistema, siendo sólo uno a la vez. Las

pruebas se realizarán en forma local y remota con las diferentes velocidades mediante redes WAN de 56 Kbps y 128 Kbps y LAN de 10 Mbps y 100 Mbps, respectivamente.

DataSocket Server Local
Tiempo medio: 300 ms

DataSocket Server 56 Kbps analógico (44 Kbps reales)
Tiempo medio: 9500 ms

DataSocket Server 128 Kbps digital
Tiempo medio: 4000 ms

DataSocket Server 10 Mbps
Tiempo medio: 500 ms

DataSocket Server 100 Mbps
Tiempo medio: 310 ms

4.2.2. Visualización de Valores RMS de Voltaje, Corriente, Potencias y Factor de Potencia

Esta prueba trata de medir el tiempo en milisegundos de conexión y actualización del servidor de DataSocket hacia los clientes y viceversa, para permitir la visualización de los datos de valores RMS de voltaje, corriente, potencia aparente, potencia real, potencia reactiva y factor de potencia para cada uno de los circuitos trifásicos ingresados en el sistema. Las pruebas se realizarán en forma local y remota con las diferentes velocidades mediante redes WAN de 56 Kbps y 128 Kbps y LAN de 10 Mbps y 100 Mbps, respectivamente.

DataSocket Server Local
Tiempo medio: 110 ms

DataSocket Server 56 Kbps analógico (44 Kbps reales)

Tiempo medio: 3200 ms

DataSocket Server 128 Kbps digital
Tiempo medio: 1550 ms

DataSocket Server 10 Mbps
Tiempo medio: 200 ms

DataSocket Server 100 Mbps
Tiempo medio: 110 ms

CAPÍTULO 5

CONCLUSIONES Y RECOMENDACIONES

5. CONCLUSIONES Y RECOMENDACIONES

5.1 CONCLUSIONES

➢ Como apreciación general sobre el diseño y la implementación del proyecto se puede decir que se cumplió a cabalidad con las metas y objetivos iniciales planteados. Sin embargo, en el momento de las pruebas de la interfase de cliente se encontró algunos problemas y limitaciones presentadas por los navegadores de Internet de otras marcas diferentes a Microsoft; dejando de esta manera reducido el campo de los clientes que pueden realizar el monitoreo sobre Internet o una intranet para utilizar los productos de la línea de Microsoft, mediante Internet Explorer. Pese a este inconveniente se sabe que un porcentaje muy alto de usuarios utiliza Microsoft Windows e Internet Explorer.

➢ Se cumplió adecuadamente una de las metas del proyecto con el almacenamiento de la información obtenida de la adquisición de datos y procesamiento en estructuras que sean más fáciles de administrar que los archivos planos, mediante el uso de Bases de Datos.

➢ Se realizaron las aplicaciones de servidor y cliente cumpliéndose en un 50% las expectativas planteadas en el proyecto. La aplicación de servidor se realizó utilizando LabVIEW 6i y es totalmente transportable en cuanto a Bases

de Datos. La aplicación de cliente está conforme sólo en un 50% debido a que el Monitoreo en Tiempo Real se puede visualizar únicamente en Microsoft Internet Explorer.

➢ Se observó que los sistemas de administración de bases de datos relacionales ofrecen un buen rendimiento a la aplicación en lo concerniente a la manipulación y administración de los mismos, incluso desde las versiones personales que son diseñadas sólo para el desarrollo de aplicaciones. Las versiones de escritorio de los suites de oficina como Access, Approach o Paradox no son adecuadas.

➢ Se revisó 3 tecnologías de conexión a los sistemas de bases de datos: ODBC, JDBC y OLEDB. La primera es la más común pero ya obsoleta, la segunda es independiente de la plataforma pero no es estándar para Microsoft y es necesaria la instalación de un compilador para JAVA, finalmente se utilizó la tercera opción por ser de última tecnología y la más óptima al momento para una eficiente conectividad al sistema de base datos. Además ésta se encuentra disponible gratuitamente en el sitio web de Microsoft para cualquier sistema de Windows de 32 y 64 bits.

➢ Se realizó el proceso de creación de las estructuras de almacenamiento de datos; en el cual primero se realiza un modelo conceptual o lógico de la información, mediante una abstracción de los objetos de la vida real de los cuales se desea almacenar su información. Luego se procede a generar el modelo físico de datos a partir del modelo lógico, en el cual ya se separan las relaciones múltiples y se utilizan los tipos de datos del sistema de bases de datos como: enteros, reales, decimales, caracteres, moneda, etc. Finalmente se genera el script de la base de datos; el cual consiste en un conjunto de instrucciones para creación de tablas, índices, llaves primarias, llaves foráneas y relaciones.

➢ Para la transmisión de datos en tiempo real se escogió un servicio que funciona sobre la infraestructura del protocolo TCP/IP estándar. Esta tecnología propietaria y desarrollada por National Instruments se llama DataSocket, y se

encuentra implementada mediante un modelo muy conocido de Cliente – Servidor.

➢ El servidor de DataSocket permite al desarrollador de aplicaciones tener comunicaciones en tiempo cuasi-real entre equipos de datos sin tener que preocuparse por la implementación y administración de bajo nivel mediante el protocolo TCP/IP. Sin embargo se observó que el servidor de DataSocket está implementado mediante la tecnología ActiveX, lo cual hace limitada su utilización sobre sistemas operativos distintos de Microsoft Windows; razón por la cual no existe en las versiones de LabVIEW para Macintosh, Linux y Solaris.

➢ Se analizó el funcionamiento del servidor de DataSocket. Este intercambia la información con los clientes mediante los ítems de datos que son nombres con los cuales se identifica un dato y pueden ser de lectura, escritura o lectura/escritura. Además, el servidor utiliza fuentes y destinos de datos que los que suministran y consumen datos; soporta una serie de protocolos de comunicación como: http, ftp, opc, file, etc. debido a que está implementado sobre el TCP/IP estándar.

➢ El sistema fue diseñado siguiendo las tendencias de las nuevas tecnologías, el mismo está diseñado con una metodología de diseño de tres capas. En esta implementación no son fácilmente visibles todas las capas; sin embargo, mantiene la lógica de operación de una aplicación tradicional estructurada. La primera capa es el sistema de bases de datos, la segunda es el servidor web de aplicaciones y la tercera es la aplicación del cliente; la aplicación en el servidor de adquisición forma parte de la capa media ya que ésta puede contener varias subcapas.

➢ Debido a que el sistema se diseñó para ser los más autónomo e inteligente posible realiza algunas tareas de configuración cuando se lo ejecuta por primera vez. Requiere de una configuración básica donde solicita el proveedor de bases de datos, el nombre de la instancia de datos, el nombre del servidor de base de datos, el nombre de usuario, clave de usuario y nombre del servidor de DataSocket. Estos datos se almacenan en el registro de Windows

para el funcionamiento de todo el sistema; si por alguna razón el sistema no llega a comunicarse con el sistema de base de datos o con el servidor de DataSocket, el mismo permite cambiar la configuración o intentar de nuevo la conexión.

- Dado el estado actual de la tecnología se hizo necesario que la aplicación, que maneja datos de índole administrativa e informativa requiera la adecuada seguridad para evitar una filtración, alteración o libre distribución sin autorización de la información.

- Para almacenar la información de configuración de la aplicación se estudió diferentes métodos. Un archivo de datos de configuración adjunto, una línea en el archivo de configuración general de Windows o un conjunto de llaves en el registro de Windows. Se escogió la última debido a que es más seguro para su edición o eliminación y, además es el estándar actual de las aplicaciones basadas en Windows.

- La conexión a la base de datos se la realizó mediante ActiveX, tecnología de Microsoft que funciona sólo sobre plataformas Windows debido a que LabVIEW no tiene soporte integrado de conexión a sistemas de bases de datos. Esta limitación deja el campo cerrado para una implementación de ésta aplicación sobre sistemas basados en Linux o MacOS. Para la implementación multiplataforma se necesita disponer del código de conexión a los sistemas de bases de datos en lenguaje C, debido a que es el único que soporta LabVIEW. Una vez realizada la conexión se crearon instrumentos virtuales mediante SQL estándar que realicen las tareas de conteo de registro; selección de columnas bajo distintos parámetros de ordenamiento, agrupación y condiciones; inserción de registros simples y en matrices; eliminación y actualización de registros.

- Para implementar la seguridad en el sistema se creó una rutina de manejo de usuarios. El usuario deberá identificarse cuando ingrese al sistema, para que obtenga ciertos permisos sobre la aplicación de acuerdo a su nivel de seguridad. De esta manera solo los administradores pueden hacer cambios sobre los usuarios, áreas, circuitos y puntos de medición.

➢ Considerando que pueden suceder muchos eventos con respecto al hardware que se usa en la adquisición de datos en el transcurso de funcionamiento de la aplicación, como por ejemplo: daño de un canal, daño de una tarjeta, reactivación de una tarjeta. Se decidió añadir un registro que realice una traza de los eventos que suceden con las tarjetas de adquisición y uno de la inspección inicial en la aplicación.

➢ La aplicación que se realizó es totalmente flexible y dinámica, ya que permite adicionar tarjetas de adquisición de datos al computador y cambiar la configuración del sistema para el punto de medición aumentado de acuerdo a la configuración de la planta. De esta manera, al reiniciar la aplicación se activará la nueva configuración instalada de tarjetas de adquisición.

➢ La calidad y precisión de los datos que sean obtenidos dependen de la calidad de las tarjetas de adquisición así como de la capacidad del computador en el que están instaladas las mismas. Debido a que a mayor cantidad de puntos de muestreo se necesita mayor tiempo de proceso para las conversiones análogas a digitales.

➢ Es necesario obtener datos críticos de las mediciones que se están realizando, por esta razón se realizó una rutina que almacene los mismos cada 5 minutos en forma estadística.

➢ En esta aplicación es importante el tiempo de respuesta de la base de datos, debido a que a mayor número de puntos de medición se necesita que la base de datos responda rápidamente en las operaciones de inserción de registros nuevos de las mediciones realizadas, por esta razón se utilizó la más rápida y compatible con Windows.

➢ Se realizó una rutina para la desconexión ordenada y adecuada de la base de datos, para que se escriban en disco todos los datos que hayan quedado en memoria y también para garantizar que no exista una pérdida o alteración de los mismos.

➢ Se observó que LabVIEW consume muchos recursos de procesador cuando realiza la adquisición de datos, debido a que es un programa que funciona secuencialmente en un bucle indefinido y no en forma orientada a eventos como los lenguajes visuales de propósito general para Windows como Visual Basic, Visual C++ o Delphi. Cuando se realiza la adquisición, procesamiento y almacenamiento de datos se observó que el computador queda casi completamente inutilizado por la cantidad de recursos que consume, convirtiéndose en un proceso dedicado del mismo y dejándolo inutilizado para la ejecución de otras actividades.

➢ Existen una gran cantidad de datos y variables que se generan en la aplicación de monitoreo por el tiempo de almacenamiento de datos críticos obtenidos en las mediciones de los circuitos trifásicos. Se observó que la memoria del computador se consume rápidamente y ocasiona que se vuelva muy lento para operar.

➢ Mientras más puntos de medición existan en la planta mayor será la cantidad de información estadística que se almacene en la base datos, y esto aumentará el tamaño de la misma a un ritmo geométrico, pudiendo crear una saturación de disco duro y también del sistema operativo.

➢ Para la visualización de la aplicación de cliente se necesitó la ayuda de otro programa adicional que se encarga de procesar los datos para el cliente, debido a que un navegador de Internet no tiene las capacidades de proceso de una aplicación de escritorio de Win32. El servidor web es el encarga de realizar las operaciones de proceso y luego enviarlas al cliente. Existen muchos productos de servidores web disponibles en el mercado, sin embargo se ha optado por utilizar el servidor web integrado en el sistema operativo de Microsoft, el Internet Information Server.

➢ Se observó que una gran ventaja de las aplicaciones web es que cuando se desea cambiar su contenido se lo puede realizar desde cualquier lugar y después colocar en el mismo en forma remota. Otra ventaja es que no existen

elevados costos de mantenimiento de la aplicación, ya que los usuarios no necesitan reinstalar la aplicación o parches cada vez que se cambie la misma.

➢ La aplicación de cliente se vale de varios recursos para cumplir su cometido y funciona de manera semi-independiente desde cierto punto de vista. Para el monitoreo en tiempo real se necesita de que la aplicación de servidor de LabVIEW se encuentre funcionando ya que ésta toma los datos y los procesa, luego se necesita del servidor de DataSocket, que acepta los datos de los objetos publicadores y los pone a disposición de los objetos suscriptores. Por el otro lado, la visualización de los datos históricos estadísticos sólo necesita que esté levantado el servidor de base de datos y el servidor web, éste se encargará de realizar las peticiones a la base de datos y luego devolver los datos a la aplicación cliente.

➢ En las pruebas se observó que la aplicación de cliente tiene un rendimiento muy aceptable sobre entornos de redes locales donde la velocidad mínima en la actualidad es de 10 Mbps y puede llegar hasta 1 Gbps en entornos tecnológicos de punta. Sin embargo en entornos remotos las velocidades de los enlaces pueden ser tan bajas como 9600 bps en comunicaciones celulares hasta un máximo de 128 Kbps de un enlace de alta de banda ancha que no representa mucho costo económico, el rendimiento es pobre lo referente a la transmisión de datos de las formas de onda en tiempo real.

➢ Se observó que la parte de la aplicación de cliente que realiza el monitoreo en tiempo real funciona perfectamente en los navegadores web de Microsoft (Internet Explorer), sin embargo presenta problemas con los otros navegadores debido a que el control ActiveX utilizado para hacer el monitoreo no es un estándar de los navegadores web distintos de Microsoft Internet Explorer y que esta tecnología sólo funciona sobre sistemas operativos Windows. Además, para que pueda ejecutarse el control ActiveX sin problemas debe estar firmado digitalmente, de otro modo se debe reducir el nivel de seguridad en la configuración del navegador web para que se pueda descargar el control en el computador cliente y funcione la aplicación dejando de éste modo al sistema operativo vulnerable a ataques externos provenientes del Internet.

5.2 RECOMENDACIONES

➢ Para el uso de sistemas de bases de datos se debe conocer muy bien el lenguaje estructurado de consulta (SQL), ya que es estándar y muy poderoso en la administración de la información y sus estructuras contenedoras. El mismo consta de dos partes bien establecidas: el lenguaje de definición de datos y el lenguaje de manipulación de datos.

➢ Se recomienda que la implementación de la aplicación se la realice de tal forma que consuma la menor cantidad de recursos posible y no sature el computador que está realizando el monitoreo, mediante el uso de los recursos indispensables de programación y la limpieza de los objetos después de su uso.

➢ Se recomienda que se utilice una unidad de disco duro o de almacenamiento dedicado y con gran capacidad ya que los datos provenientes de la adquisición empezarán a crecer con gran rapidez y en forma exponencial de acuerdo al número de puntos monitoreados si no se hace un mantenimiento de la información o se desea mantener los registros históricos por largo tiempo.

➢ Se recomienda que cuando la aplicación de monitoreo empiece a crecer en cuanto a cantidad de puntos monitoreados, se utilicen 3 computadores distintos repartidos de la forma siguiente con enlaces de red de alta velocidad.

- Servidor de Base de Datos
- Servidor de Aplicaciones (Web) y DataSocket
- Aplicación de Servidor de Monitoreo

➢ Se recomienda realizar la aplicación en un lenguaje orientado a objetos, para de esta forma aprovechar las características del mismo y por lo tanto implementar un funcionamiento en eventos para evitar saturar el computador por los lazos indefinidos que utiliza LabVIEW.

➢ Hay que tomar en cuenta que a mayor cantidad de muestras para una onda, mejor es la calidad de la reconstrucción de la misma pero también toma

más tiempo de proceso de la matriz de datos adquiridos. Se recomienda disponer de un computador con una cantidad adecuada de memoria RAM.

- Debido a que LabVIEW es un lenguaje de programación de ejecución secuencial, National Instruments ha creado un paquete de controles y funciones para desarrollo de aplicaciones de adquisición de datos y control que se adapta a Visual Studio de Microsoft en el cual se puede programar en Visual Basic .NET, Visual C# .NET o Visual C++, cuyo nombre es Measurement Studio.

- El lenguaje sobre el que se desarrolle la aplicación debería tener un soporte nativo de conexión a sistemas de bases de datos, para que éste no sea un punto que utilice nuestros esfuerzos y de esta manera orientarlos a la aplicación específica de la adquisición de datos; nuevamente se recomienda utilizar Measurement Studio que se integra fácilmente en Visual Studio .NET.

- Las aplicaciones de monitoreo en la actualidad deben utilizar un sistema de base de datos que permita una mejor administración de la información, debido a que los sistemas de hoy en día no son aislados; por el contrario se pretende tener una integración completa desde los niveles operativos hacia arriba pasando por los administrativos y hasta los gerenciales para poder tener una visión real de la productividad de una empresa.

- Finalmente se debe buscar una solución tecnológica que sea estándar en todos los diferentes entornos de sistemas operativos, tal es el caso del protocolo TCP/IP. Esto permitiría crear una aplicación de cliente muy transportable y muy transparente con respecto al sistema operativo donde será utilizada y de esta forma reducir los costos que tendría una empresa por soporte técnico de la aplicación y principalmente el de usuario.

- Se recomienda realizar el transporte de datos en tiempo real mediante los servidores de OPC ya que son estándares a nivel industrial y existen en todos los sistemas operativos más populares.

- Del mismo modo se recomienda utilizar otra forma de implementación de la visualización de las ondas en la aplicación cliente mediante otra tecnología que no sea ActiveX, debido a que su funcionamiento está limitado a los navegadores Internet Explorer de Microsoft.

REFERENCIAS BIBLIOGRÁFICAS

- JOHNSON, Johnny, **Análisis de Circuitos,** 5ta. Edición, Ed. Prentice Hall, México – 1996, 835 pág.
- NATIONAL INSTRUMENTS CORPORATION, **LabVIEW 6i - User Manual,** 3ra. Edición, Ed. National Instruments, U.S.A. – 2000, 272 pág.
- NATIONAL INSTRUMENTS CORPORATION, **Component Works - User Manual,** 1ra. Edición, Ed. National Instruments, U.S.A. – 2001, 50 pág.
- MICROSOFT CORPORATION, **SQL Server 2000 - User Manual,** 2da. Edición, Ed. McGraw Hill, U.S.A. – 2000, 1200 pág.
- MICROSOFT CORPORATION, **Visual Studio 6.0 - User Manual,** 3ra. Edición, Ed. McGraw Hill, U.S.A. - 2000, 1100 pág.
- MICROSOFT CORPORATION, **Programming with HTML and Active Server Pages,** 2da. Edición, Ed. McGraw Hill, U.S.A. – 2002, 600 pág.

- www.microsoft.com/sql, Microsoft SQL Server
- www.oracle.com/database, Oracle Database
- www.sybase.com/ase, Adaptive Server Enterprise
- www.ibm.com/database/db2, IBM DB2 Database
- www.mysql.com/mysql, MySQL Database
- www.ingres.com/dbsystem, Ingres Database Sytem

ANEXO 1

SQL Server 2000 Product Overview

Content Updated: February 24, 2005

Business today demands a different kind of data management solution. Performance, scalability, and reliability are essential, but businesses now expect more from their key IT investments.

SQL Server 2000 exceeds dependability requirements and provides innovative capabilities that increase employee effectiveness, integrate heterogeneous IT ecosystems, and maximize capital and operating budgets. SQL Server 2000 provides the enterprise data management platform your organization needs to adapt quickly in a fast-changing environment.

With the lowest implementation and maintenance costs in the industry, SQL Server 2000 delivers rapid return on your data management investment. SQL Server 2000 supports the rapid development of enterprise-class business applications that can give your company a critical competitive advantage.

Benchmarked for scalability, speed, and performance, SQL Server 2000 is a fully enterprise-class database product, providing core support for Extensible Markup Language (XML) and Internet queries.

Easy-to-Use Business Intelligence (BI) Tools

Through rich data analysis and data mining capabilities that integrate with familiar applications such as Microsoft Office, SQL Server 2000 enables you to provide all of your employees with critical, timely business information tailored to their specific information needs. Every copy of SQL Server 2000 ships with a suite of BI services.

Self-Tuning and Management Capabilities

Revolutionary self-tuning and dynamic self-configuring features optimize database performance, while management tools automate standard activities. Graphical tools and wizards simplify setup, database design, and performance monitoring, allowing database administrators to focus on meeting strategic business needs.

Data Management Applications and Services

Unlike its competitors, SQL Server 2000 provides a powerful and comprehensive data management platform. Every software license includes extensive management and development tools, a powerful extraction, transformation, and loading (ETL) tool, business intelligence and analysis services, and new capabilities such as Notification Services. The result is the best overall business value available.

Evaluating Your Business Needs

SQL Server 2000 is available in several editions:

- SQL Server 2000 Enterprise Edition (64-bit)
- SQL Server 2000 Enterprise Edition
- SQL Server 2000 Standard Edition
- SQL Server 2000 Workgroup Edition
- SQL Server 2000 Developer Edition
- SQL Server 2000 Personal Edition
- SQL Server 2000 Desktop Engine (MSDE)
- SQL Server 2000 Windows® CE Edition (SQL Server CE)

All products are available in Simplified Chinese, Traditional Chinese, English, French, German, Italian, Japanese, Korean, and Spanish.

To determine which edition will best suit your business needs:

1. Visit the Features by Edition page to compare features in each edition of SQL Server.
2. When you're ready to make your selection, visit the Choosing an Edition of SQL Server 2000 page.

Product SKU Matrix

	Enterprise (64-bit) Edition	Enterprise Edition	Standard Edition	Workgroup Edition
Overview	Enterprise Edition (64-bit) provide the most scalable data platform to take advantage of the class of Intel Itanium-based servers. Addressing more memory than any other edition of SQL Server, it scales to the	Enterprise Edition includes the complete set of SQL Server data management and analysis features and is uniquely characterized by several features that make it the most scalable and available edition of SQL Server 2000. It scales to the performance levels required to support the largest Web sites, Enterprise Online Transaction Processing (OLTP) systems and Data Warehousing systems. Its support for failover clustering also makes it ideal for any mission	Standard Edition is the data management and analysis platform for small and medium-sized organizations. It includes the essential functionality needed for e-commerce, data warehousing, and line-of-business solutions. Standard Edition's integrated business	Workgroup Edition is the data management solution for small organizations or workgroups within larger entities. It includes all of the core database features needed for data management in an easy-to-use, affordable and simple to manage package. Workgroup Edition is ideal for the small organization for line-of-business (LOB), local data store, and e-commerce

	performance levels required to support the largest Data warehousing and analysis applications, ecommerce websites and Enterprise business systems. Supporting up to 8 nodes in failover clustering, SQL Server 2000 (64-bit) provides a high level of reliability and availability for your mission-critical applications.	critical line-of-business application. Additionally, this edition includes several advanced analysis features that are not included in SQL Server 2000 Standard Edition. There are four main areas in which the additional features of SQL Server 2000 Enterprise Edition are most evident: ▪ Scalability ▪ Availability/uptime ▪ Performance ▪ Advanced analysis	intelligence features provide organizations with the capabilities needed to manage and analyze their data. Reporting Services provides robust business intelligence features. Analysis Services provides data mining features and the core OLAP (On-Line Analytical Processing) functionality. Standard Edition includes all of the features of Workgroup Edition plus the following: ▪ Analysis Services ▪ Data Transformation Services (DTS) ▪ Reporting Services ▪ SQL Profiling and performance analysis tools	solutions. All of the management features that make SQL Server easy to support and simple to manage are included in Workgroup Edition. Workgroup Edition includes key features that allow you to easily manage and organize your data: ▪ Enterprise Manager ▪ Full-Text Search ▪ Import/Export ▪ Replication (snapshot, transactional, and merge) ▪ Stored procedure development and debugging tools
Operating Systems	Windows Server 2003 Enterprise Edition Windows Server 2003 Datacenter Edition	Windows Server™ 2003, Standard Edition* Windows Server™ 2003, Enterprise Edition* Windows Server™ 2003, Datacenter Edition* Windows® 2000 Server Windows® 2000 Advanced Server Windows® 2000 Datacenter Server	Windows Server 2003, Standard Edition* Windows Server 2003, Enterprise Edition* Windows Server 2003, Datacenter Edition* Windows 2000 Server Windows 2000 Advanced Server Windows 2000 Datacenter Server	Windows Server 2003, Standard Edition* Windows Server 2003, Enterprise Edition* Windows Server 2003, Datacenter Edition* Windows 2000 Server Windows 2000 Advanced Server Windows 2000 Datacenter Server Windows XP
Scalability	Up to **64 Processors** Up to **512 GB of Memory**	Up to **32 Processors** Up to **64 GB of Memory**	Up to **4 Processors** Up to **2 GB of Memory**	Up to **2 Processors** Up to **2 GB of Memory**

Purchasing and Licensing	How To Buy	How To Buy	How To Buy	How To Buy

* Requires SQL Server 2000 Service Pack 3 or later to be applied.

	Developer Edition	Personal Edition	Desktop Engine (MSDE)	Windows CE Edition
Overview	Developer edition is designed to allow developers to build any type of application on top of SQL Server. It includes all the functionality of Enterprise Edition but with a special development and test end-user license agreement (EULA) that prohibits production deployment. It is the ideal choice for Independent Software Vendors (ISVs), consultants, system integrators, solution providers, and corporate developers developing and testing applications because it is cost effective, runs on a variety of platforms, and can be upgraded for production use to SQL Server 2000 Enterprise Edition. It is the only edition of SQL Server 2000 that gives the licensee the right to download and install SQL Server 2000 Windows CE Edition (SQL Server CE). The Developer Edition licensee also has the right to redistribute SQL Server CE-based applications to an unlimited number of devices at no	Personal Edition is ideal for mobile users who spend some of their time disconnected from the network but run applications that require SQL Server data storage, and for stand-alone applications that require local SQL Server data storage on a client computer. This edition is functionally equivalent to Standard Edition, with the following exceptions: ▪ It includes a concurrent workload governor that limits its scalability; performance degrades when more than five Transact-SQL batches are executed concurrently. ▪ It cannot act as a transactional replication publisher (subscriber only).	SQL Server 2000 Desktop Engine (MSDE 2000) is a free, redistributable version of SQL Server. Non-Microsoft software developers can include it in applications they build that use SQL Server to store data. MSDE is an ideal solution for: ▪ Client applications that require an embedded database. ▪ Basic websites that serve up to 25 concurrent users. ▪ New developers who are learning how to build data-driven applications. This edition is functionally equivalent to Standard Edition with the following exceptions: ▪ Does not include graphical management tools (Enterprise Manager). ▪ Has a workload governor that limits the	Windows CE Edition (SQL Server CE) is the compact database for rapidly developing applications that extend enterprise data management capabilities to devices. SQL Server CE is a full-fledged member of the SQL Server 2000 family, with tools, application programming interfaces (APIs), and SQL grammar that developers and database administrators with existing SQL Server skills will recognize. SQL Server CE is the only edition of SQL Server 2000 that provides relational database management capabilities on Windows CE-based devices. The SQL Server CE engine provides an essential set of relational database features, including: ▪ an optimizing query processor ▪ support for transactions ▪ support for assorted data types ▪ a compact footprint that preserves

	additional cost beyond the purchase price of SQL Server 2000 Developer Edition.		number of concurrent transactions. ▪ Does not include any analysis capabilities (such as OLAP, DTS, data mining, and data warehousing features). ▪ Has different scalability limits.	system resources ▪ Remote data access and merge replication over Hypertext Transfer Protocol (HTTP)
Operating Systems	Windows Server 2003 Standard Edition* Windows Server 2003 Enterprise Edition* Windows Server 2003 Datacenter Edition* Windows® XP Professional Windows® 2000 Professional Windows 2000 Server Windows 2000 Advanced Server Windows 2000 Datacenter Server	Windows Server 2003, Standard Edition* Windows Server 2003, Enterprise Edition* Windows Server 2003, Datacenter Edition* Windows XP Professional Windows 2000 Professional Windows 2000 Server Windows 2000 Advanced Server Windows 2000 Datacenter Server	Windows Server 2003, Standard Edition* Windows Server 2003, Enterprise Edition* Windows Server 2003, Datacenter Edition* Windows XP Professional Windows XP Home Windows 2000 Professional Windows 2000 Server Windows 2000 Advanced Server Windows 2000 Datacenter Server	Windows CE version 2.11 or later ▪ Handheld PC Pro (H/PC Pro) ▪ Palm-size PC (P/PC) ▪ Pocket PC
Scalability	Up to **32 Processors** Up to **64 GB of Memory**	Up to **2 Processors** Up to **2 GB of Memory**	Up to **2 Processors** Up to **2 GB of Memory**	**1 Processor**
Purchasing and Licensing	How To Buy	Not available separately. Personal Edition is included with: ▪ SQL Server Enterprise Edition ▪ SQL Server Standard Edition	How To Buy	Free download for licensed SQL Server 2000 Developer Edition users. ▪ Download SQL Server CE

* Requires SQL Server 2000 Service Pack 3 or later to be applied.

Features at a Glance

Technology	Capabilities
Data Warehousing	
Analysis (OLAP) Services	Perform rapid, sophisticated analysis on large and complex data sets using multi-dimensional storage.
Closed-Loop Analysis	Take analysis one step further with OLAP actions, allowing results to drive next steps in the business process.
Data Mining	Discover patterns and trends with data mining, and make predictions about future trends in your business.
Data Transformation Services	Automate routines that extract, transform, and load data from heterogeneous sources.
Data Warehousing Alliance	Choose among broad offerings from non-Microsoft analysis tools vendors that extend Analysis Services functionality.
English Query	Enable your users to pose questions in English instead of using multi dimensional expressions (MDX).
Indexed Views	Gain performance from your existing hardware by storing query results and reducing response times.
Meta Data Services	Increase developer productivity and reduce administrative overhead by sharing meta data among different tools and environments, using standard information models.
Office 2000 Integration	Use Pivot Table Services in Microsoft Excel to enhance your data analysis.
OLAP Flexibility	Use multiple dimension types for flexible business analysis.
Web-Enabled Analysis	Analyze data from remote OLAP cubes that are Web accessible.
Enterprise Data Management	
Clickstream Analysis	Gain a deep understanding of online customer behavior, so that you can make better business decisions.
Distributed Partitioned Views	Partition your workload among multiple servers for additional scalability.
Full Text Search	Use and manage both your structured and unstructured data, including searching through Microsoft Office documents.
High Availability	Maximize the availability of your business applications with log shipping, online backups, and failover clusters.
Improved Developer Productivity	User-defined functions, cascading referential integrity, and the integrated Transact SQL (T-SQL) debugger allow you to reuse code to simplify the development process.
Integration with Microsoft Windows Server System™—Microsoft BizTalk and Microsoft Commerce Server	SQL Server 2000, in conjunction with other Microsoft Server products, provides even more power for your e-business.
Rich XML Support	Simplify the integration of your back-end systems and data transfer across firewalls using XML.
Security	Ensure your applications are secure in any networked

	environment, with role-based security and file and network encryption.
Simplified Database Administration	Automatic tuning and maintenance features enable administrators to focus on other critical tasks.
VI SAN	Improve your overall system performance with built-in support for a virtual system area network (VI SAN).
Web Access to Data	Connect to your SQL Server 2000 databases and OLAP cubes flexibly, by using the Web with no additional programming.
Web and Application Hosting	Have your e-commerce solution hosted by an application service provider, with SQL Server 2000 support for multiple instances.
Line of Business	
Analysis (OLAP) Services	Perform rapid, sophisticated analysis on large and complex data sets using multi-dimensional storage.
Application Hosting	With multi-instance support, SQL Server enables you to take full advantage of your hardware investments so that multiple applications can be run on a single server, or outsourced.
Extend Applications	Support for access by devices, such as Microsoft Windows® CE handheld units, provides broader access to applications and extends your user base.
High Availability	Maximize the availability of your business applications with log shipping, online backups, and failover clusters.
Replication	With SQL Server 2000 you can implement merge, transactional, and snapshot replication with heterogeneous systems.
Scalability	Scale your applications up to 32 CPUs and 64 gigabytes (GB) RAM. SQL Server 2000 has demonstrated record-breaking performance that you can leverage.
Security	Ensure your applications are secure in any networked environment, with role-based security and file and network encryption.
Simplified Database Administration	Automatic tuning and maintenance features enable administrators to focus on other critical tasks.
VI SAN	Improve your overall system performance with built-in support for a virtual system area network (VI SAN).
Web Access to Data	Connect to your SQL Server 2000 databases and OLAP cubes flexibly, by using the Web with no additional programming.
XML	XML support allows your data to be easily stored for use by Web applications.

ANEXO 2

Windows 2000 Server Benchmarks

Content Updated: February 13, 2004

Printer-friendly version

The Windows 2000 Server family has repeatedly demonstrated performance advantages over UNIX variants in a wide range of industry benchmarks.

The tables below summarize recent Windows 2000 Server family benchmark results and provide links to more information.

Performance Benchmarks

Online Transaction Processing (OLTP)	
Workload / Application	**Result**
TPC-C (Non-Clustered 8-ways)	69,901 tpmC
TPC-C (Non-Clustered 4-ways)	48,911 tpmC
TPC-C (Non-Clustered 2-ways)	34,473 tpmC
Decision Support	
Workload / Application	**Result**
TPC-H (3 TB Clustered)	21,053 QphH@3 TB
TPC-H (1 TB Clustered)	22,361 QphH @1 TB
TPC-H (300 GB Clustered)	12,995 QphH @300 GB
E-Business	
Workload / Application	**Result**

SAP R/3 Sales & Dist. (Three-tier)	24,000 concurrent users	
SAP R/3 Retail	3,165,000 line items/hr	
TPC-W (100,000 item count)	10,439 WIPS	
TPC-W (10,000 item count)	10,449 WIPS	
Great Plains	2,400 concurrent users	
Customer Relationship Management (CRM)		
Workload/Application	**Result**	
PeopleSoft 8 CRM	30,000 concurrent users	
PeopleSoft 8.4 CRM (SSCS)	25,200 concurrent users	
Onyx	57,000 concurrent users	
Pivotal eRelationship	20,000 concurrent users	

Top

Price-Performance Benchmarks

Online Transaction Processing (OLTP)		
Workload/Application	**Result**	**Rank**
TPC-C (Clustered Systems)	$13.02 $/tpmC	#1
TPC-C (Non-Clustered Systems)	$2.78 $/tpmC	#1
Decision Support		
Workload/Application	**Result**	**Rank**
TPC-H (3 TB Clustered)	$291.00 $/QphH	#1
TPC-H (1 TB Clustered)	$255.00 $/QphH	#1
TPC-H (300 GB Clustered)	$199.00 $/QphH	#1
TPC-H (100 GB Non-Clustered)	$82.00 $/QphH	#1
E-Business		
Workload/Application	**Result**	**Rank**
TPC-W (100,000 item count)	$34.60 $/WIPS	#1
TPC-W (10,000 item count)	$24.50 $/WIPS	#1

ANEXO 3

Oracle10g and SQL Server 2000 Price Comparison

Content Updated: June 24, 2004

Compare pricing for SQL Server 2000 and Oracle 10*g* by downloading this Microsoft Excel–based pricing calculator. You can use the calculator to determine pricing differences between these two relational database products or you can refer to the price comparison tables on this page.

SQL Server 2000 provides management, security, and business intelligence capabilities as an integrated solution at no additional cost. Oracle sells these important features as extra-charge options, and most of these features are available only with Oracle 10*g* Enterprise Edition. These include:

Key management features like diagnostic tools, performance monitoring tools, and tuning tools. These management tools are included as standard features in SQL Server 2000 Standard and Enterprise Edition. However, these tools are only available as extra-cost options with both editions of Oracle 10*g*.

Advanced security features like network data packet, public key infrastructure (PKI), and single sign-on support. These important security capabilities are included with SQL Server 2000 Standard and Enterprise Edition at no additional cost. Oracle provides these features as extra-cost options and only makes them available with the enterprise edition. This means users of Oracle 10*g* Standard Edition must find alternative solutions or install add-on security products from third-party software providers.

Business intelligence components like online analytical processing (OLAP), enterprise reporting, and data mining. Again, these features are included in SQL Server 2000 Standard and Enterprise Edition. Oracle, however, offers these items only as extra-cost options and only with Oracle 10*g* Enterprise Edition.

Significant price differences between SQL Server 2000 and Oracle 10*g* arise when comparing the costs associated with these key data management features.

Price Comparisons

The following tables provide clear comparisons of the significantly higher costs for Oracle database licenses compared to SQL Server licenses.

Click any of the following links to view specific price comparison information for Oracle 10*g* Enterprise Edition and SQL Server 2000 Enterprise Edition.

Click any of the following links to view specific price comparison information for Oracle 10*g* Standard Edition and SQL Server 2000 Standard Edition.

Note: All prices reflect pricing for purchases within the United States and are in U.S. dollars. Competitor pricing based on the June 1, 2004 list price from the Oracle Web site.

Oracle 10g Enterprise Edition and SQL Server 2000 Enterprise Edition

Basic Database Server

Number of CPUs	Oracle 10*g* Enterprise Edition	SQL Server 2000 Enterprise Edition	With SQL Server, you'll save...
1	$40,000 US	$19,999 US	$20,001 US
2	$80,000 US	$39,998 US	$40,002 US
4	$160,000 US	$79,996 US	$80,004 US
8	$320,000 US	$159,992 US	$160,008 US
16	$640,000 US	$319,984 US	$320,016 US
32	$1,280,000 US	$639,968 US	$640,032 US

Top

With Management Tools

Number of CPUs	Oracle 10*g* Enterprise Edition	SQL Server 2000 Enterprise Edition	With SQL Server, you'll save...
1	$46,000 US	$19,999 US	$26,001 US
2	$92,000 US	$39,998 US	52,002 US
4	$184,000 US	$79,996 US	$104,004 US

8	$480,000 US	$159,992 US	$320,008 US
16	$736,000 US	$319,984 US	$416,016 US
32	$1,472,000	$639,968 US	$832,032 US

Top

With Advanced Security Features

Number of CPUs	Oracle 10*g* Enterprise Edition	SQL Server 2000 Enterprise Edition	With SQL Server, you'll save...
1	$50,000 US	$19,999 US	$30,001 US
2	$100,000 US	$39,998 US	$60,000 US
4	$200,000 US	$79,996 US	$120,004 US
8	$400,000 US	$159,992 US	$240,008 US
16	$800,000 US	$319,984 US	$480,016 US
32	$1,600,000 US	$639,968 US	$960,032 US

Top

With Business Intelligence Features

Number of CPUs	Oracle 10*g* Enterprise Edition	SQL Server 2000 Enterprise Edition	With SQL Server, you'll save...
1	$80,000 US	$19,999 US	$60,001 US
2	$160,000 US	$39,998 US	$120,002 US
4	$320,000 US	$79,996 US	$240,004 US
8	$640,000 US	$159,992 US	$480,008 US
16	$1,280,000 US	$319,984 US	$960,016 US
32	$2,560,000 US	$639,968 US	$1,920,032 US

Top

With Management Tools, Advanced Security Features, and Business Intelligence Features

Number of CPUs	Oracle 10*g* Enterprise Edition	SQL Server 2000 Enterprise Edition	With SQL Server, you'll save...
1	$96,000 US	$19,999 US	$76,001 US
2	$192,000 US	$39,998 US	$152,002 US

4	$384,000 US	$79,996 US	$304,004 US
8	$768,000 US	$159,992 US	$608,008 US
16	$1,536,000 US	$319,984 US	$1,216,016 US
32	$3,072,000 US	$639,968 US	$2,432,032 US

Top

Oracle 10g Standard Edition and SQL Server 2000 Standard Edition

Basic Database Server

Number of CPUs	Oracle 10*g* Standard Edition	SQL Server 2000 Standard Edition	With SQL Server, you'll save...
1	$15,000 US	$4,999 US	$10,001 US
2	$30,000 US	$9,998 US	$20,002 US
4	$60,000 US	$19,996 US	$40,004 US
8	$120,000 US	$39,992 US	$80,008 US
16	$240,000 US	$79,984 US	$160,016 US
32	$480,000 US	$159,968 US	$320,032 US

Top

With Management Tools

Number of CPUs	Oracle 10*g* Standard Edition	SQL Server 2000 Standard Edition	With SQL Server, you'll save...
1	$21,000 US	$4,999 US	$16,001 US
2	$42,000 US	$9,998 US	$32,002 US
4	$84,000 US	$19,996 US	$64,004 US
8	$168,000 US	$39,992 US	$128,008 US
16	$336,000 US	$79,984 US	$256,016 US
32	$672,000 US	$159,968 US	$512,032 US

Top

With Advanced Security Features

Number of	Oracle 10*g*	SQL Server 2000 Standard

CPUs	Standard Edition	Edition
1	Features are not available.*	$4,999 US

*Advanced security features are not available with Oracle 10*g* Standard Edition. Customers will need to purchase Oracle 10*g* Enterprise Edition and then purchase the Advanced Security option. SQL Server 2000 includes these key security features with both the standard and enterprise edition at no additional cost.

Top

With Business Intelligence Features

Number of CPUs	Oracle 10*g* Standard Edition	SQL Server 2000 Standard Edition
1	Features are not available.*	$4,999 US

*OLAP and data mining are not available with Oracle 10*g* Standard Edition. Customers will need to purchase Oracle 10*g* Enterprise Edition and then purchase the OLAP and data mining option. SQL Server 2000 includes integrated OLAP and data mining with both the standard and enterprise edition at no additional cost.

Top

With Management Tools, Advanced Security Features, and Business Intelligence Features

Number of CPUs	Oracle 10*g* Standard Edition	SQL Server 2000 Standard Edition
1	Features are not available.*	$4,999 US

*Advanced security features, OLAP, and data mining are not available with Oracle 10*g* Standard Edition. Customers will need to purchase Oracle 10*g* Enterprise Edition and then purchase the advanced security, OLAP, and data mining options. SQL Server 2000 includes all key data management features like security, management tools, and business intelligence as integrated features with both the standard and enterprise edition at no additional cost.

ANEXO 4

SQL Server 2000 and IBM DB2 V8.1 Feature Comparison

Posted: June 21, 2002

Learn about the differences between SQL Server 2000 Enterprise Edition and IBM DB2 8.1 Enterprise or Extended Enterprise Editions by reviewing this feature comparison table.

Features	SQL Server 2000 Enterprise Edition	IBM DB2 V8.1 Enterprise or Extended Enterprise Editions (Microsoft Windows® and Unix version)
Scalability and Availability		
Adaptive page/row locking	Yes	No
Automated standby database	Yes	No
Graphical user interface (GUI) tool for standby database management	Yes	No
Readable standby database	Yes	Additional charges may apply
4-Node failover clustering	Yes	No
Security		
Single sign-on support	Yes	Additional charges may apply
C2 security certification	Yes	No
User auditing	Yes	Yes
Database roles	Yes	Yes
Operating system based authentication	Yes	Yes
Database based authentication	Yes	Yes
Kerberos support	Yes	Yes

Network traffic encryption (i.e. stream cipher)	Yes	AIX only
Management Automation		
Automated memory management	Yes	No
Single Universal Memory Pool	Yes	No, multiple buffer pools
Automated table space growth	Yes	No
Automated table space shrinkage	Yes	No
Automated table space reclamation	Yes	No
Automated statistics gathering	Yes	Manual process with "runstats" command
SQL Profiler	Yes	No
Auto-recompiled stored procedures	Yes	No
Visual data modeler	Yes	No
Database copying utility	Yes	No
Integrated notification services	Yes	No
Single Management Console	Yes	No, multiple tool fragments
Rich Database checking utility	Yes	New in v8.1
User-defined functions	Yes	Yes
Rich XML support	Yes	Additional charges may apply
English Query for relational data	Yes	Additional charges may apply
Microsoft Visual Studio® wizards	Yes	No
Distributed queries	Yes	Additional charges may apply
Heterogeneous queries	Yes	Additional charges may apply
SQL stored procedure support	Yes (since 1993)	Yes (since 2000)
Basic Web and database integration wizard	Yes	No
JDBC support	Yes	Yes
Text and document search	Yes	Additional charges may apply
On-delete/update referential integrity	Yes	No
Column-level collation	Yes	No
Heterogeneous database access	Yes	Additional charges may apply
Replication		
Procedural replication	Yes	No
Merge replication with conflict resolution	Yes	Limited conflict resolution
Synchronous replication	Yes	No

Anonymous pull subscriptions	Yes	No
Data definition language (DDL) replication support	Yes	No
Heterogeneous replication	Yes	Additional charges may apply
Remote agent activation	Yes	Additional charges may apply
Schema change replication	Yes	No
Transforming published data	Yes	Additional charges may apply
Microsoft Active Directory® integration for publications	Yes	No
64-bit replication support	Yes	Not with v8.1
Desktop and Mobile Device Support		
On demand desktop memory and disk tuning	Yes	Yes
Integration with Microsoft Access	Yes	Additional charges may apply
Automatic device synchronization	Yes	Additional charges may apply
Business Intelligence		
ROLAP support	Yes	Yes
MOLAP support	Yes	Additional charges may apply
HOLAP support	Yes	No
Web-enabled analysis	Yes	Additional charges may apply
Clickstream analysis	Yes	No
Integration with Microsoft Excel	Yes	No
English Query support for OLAP cubes	Yes	No
Data Mining	Yes	Additional charges may apply
Distributed partitioned cubes	Yes	Additional charges may apply
Integrated data transformation services	Yes	Additional charges may apply
View based OLAP cube partitions	Yes	No
User defined OLAP cube partitions	Yes	Additional charges may apply
OLAP cube Partitioning wizard	Yes	No
Linked OLAP cubes	Yes	No
Virtual cubes	Yes	No
Virtual cube edition	Yes	No
Partition-wise joins	Yes	No
Function based indexes	Yes	No
Integrated multi-dimensional query language	Yes	Additional charges may apply

Histogram statistics	Yes	No
Pass-through web authentication for OLAP users	Yes	Additional charges may apply
XML support		
Integrated XML support	Yes	Yes
XML Load API	Yes	No
Automatic-mode XML document shredding	Yes	No
64-bit XML support	Yes	Not with v8.1

ANEXO 5

IBM DB2 Version 8.1 and SQL Server 2000 Price Comparison

Posted: October 18, 2002

Compare pricing for SQL Server 2000 and IBM DB2 version 8.1 with this pricing calculator. Download **SQL-DB2v8.xls** on this page to use the Microsoft Excel–based calculator to determine pricing differences between these two relational database products.

The base price for IBM DB2 version 8.1 Enterprise Server Edition is 25 percent more expensive than SQL Server 2000 Enterprise Edition, and IBM charges considerably more for important add-on features such as online analytical processing (OLAP), data transformation, and data mining—items that Microsoft includes as standard product features in SQL Server 2000.

IBM DB2 Version 8.1 and SQL Server 2000 Price Comparison

Pricing for IBM DB2 version 8.1 and SQL Server 2000 is based on a per-processor licensing model, and is shown for one processor. Savings with SQL Server 2000 are greater with more processors.

IBM DB2 Version 8.1[1]		SQL Server 2000	
Product	**Price[2]**	**Product**	**Price**
Workgroup Edition	$7,500 US	Standard Edition	$4,999 US
Total	**$7,500 US**	**Total**	**$4,999 US**
Product	**Price**	**Product**	**Price**
Enterprise Edition	$25,000 US	Enterprise Edition (EE)	$19,999 US
Add-on	**Price**	**Add-on**	**Price**
Warehouse Manager	$10,600 US	Data Transformation Services	Included with EE
Intelligent Miner	$60,000 US	Data Mining	Included with EE
OLAP Server	$28,000 US	OLAP Server	Included with EE
Total	**$123,600 US[3]**	**Total**	**$19,999 US**

[1] This comparison includes IBM DB2 for UNIX and Windows only. Mainframe DB2 and AS/400 DB2 are different products with different features and pricing options.
[2] All prices reflect pricing for purchases within the United States and are in U.S. dollars. The prices listed are estimated retail prices; reseller pricing may vary.
[3] Estimated cost with add-on products required to achieve the functionality of SQL Server 2000 Enterprise Edition.

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

GLOSARIO

- **ActiveX.-** tecnología de encapsulación de controles y código de Microsoft.
- **ADO.-** Objetos de datos de ActiveX (ActiveX Data Objects)
- **ANSI.-** Instituto nacional americano de estándares (American National Standard Institute)
- **Apache.-** Servidor web de distribución libre que utiliza Linux y Oracle.
- **API.-** Interfase de programación de aplicaciones (Aplication Programmer Interfase)
- **ASP.-** Páginas activas de servidor (Active Server Pages)
- **CASE.-** Software para realizar ingeniería de sistemas mediante la ayuda del computador (Computer Aided Systems Engineering)
- **CGI.-** Intrefase de pasarela común (Common Gateway Intrefase)
- **COM.-** Modelo de componentes de objetos de Microsoft (Component Object Model)
- **DAQ.-** Adquisición de datos (Data Acquisition)
- **DataSocket.-** Conector de datos, protocolo de transferencia de datos en tiempo real creado por National Instruments.
- **DBMS.-** Sistema de administración de Bases de Datos (Database Management System)
- **DDE.-** Intercambio dinámico de datos (Dynamic Data Exchange)
- **DDL.-** Lenguaje de definición de datos (Data Definition Lenguaje)
- **DLL.-** Librería de enlace dinámico de Windows (Dynamic Link Library)
- **DML.-** Lenguaje de manipulación de datos (Data Manipulation Language)
- **DSN.-** Nombre de fuentes de datos (Data Source Name)
- **DSTP.-** Protocolo de transferencia de DataSocket (DataSocket Transfer Protocol)
- **FTP.-** Protocolo de transferencia de archivos (File Transfer Protocol)
- **HTML.-** Lenguaje de marcas de hipertexto (Hyper Text Markup Language)
- **HTTP.-** Protocolo de transferencia de hipertexto (Hyper Text Transfer Protocol)
- **ISP.-** Proveedor de servicios de Internet (Internet Service Provider)

- **JDBC.-** Conectividad de bases de datos en Java (Java Database Conectivity)
- **JSP.-** Páginas de servidor en Java (Java Server Pages)
- **Kbps.-** Kilobits por segundo (Kilobits per second)
- **LabVIEW.-** Lenguaje de programación gráfico para ingenieros creado por National Instruments (Laboratory for Windows)
- **LAN.-** Red de área local (Local Area Network)
- **Linux.-** Sistema operativo escritorio, estación de trabajo y servidor de distribución libre con orígenes en UNIX
- **MacOS.-** Sistema operativo de escritorio, estación de trabajo y servidor con orígenes en UNIX para computadores Macintosh
- **Mbps.-** Megabits por segundo (Megabits per second)
- **MDAC.-** Componentes de acceso a datos de Microsoft (Microsoft Data Access Objects)
- **ODBC.-** Conectividad abierta de bases de datos (Open Database Conectivity)
- **OLEDB.-** Conectividad de bases de datos mediante OLE (OLE Database)
- **OPC.-** OLE para control de procesos (OLE for Process Control)
- **PHP.-** Páginas phyton (Phyton Pages)
- **RAM.-** Memoria de acceso aleatorio (Ramdom Access Memory)
- **RMS.-** Valor medio cuadrático (Root Mean Square)
- **SDK.-** Kit de desarrollo de software (Software Development Kit)
- **SPI.-** Interfase del proveedor de servicios de bases de datos (Service Provider Interfase)
- **Solaris.-** Sistema operativo de escritorio, estación de trabajo y servidor con orígenes en UNIX creado por Sun MicroSystems
- **SQL.-** Lenguaje estructurado de consulta (Structured Query Language)
- **TCP/IP.-** Transfer Control Protocol / Internet Protocol
- **TNS.-** Servicio de nombres de pantillas para Oracle (Template Name Service)
- **UDL.-** Enlace universal de datos (Universal Data Link)
- **URL.-** Localizador uniforme de recursos (Uniform Resource Locutor)
- **WAN.-** Red de área extendida (Wide Area Network)
- **Windows.-** Sistema operativo de escritorio, estación de trabajo y servidor creado por Microsoft
- **XBase.-** Archivos antiguos de bases de datos

ACTA DE ENTREGA

En Sangolquí a los 29 días de Marzo de 2005 se realiza la entrega del proyecto de tesis para la obtención del título en ingeniería electrónica, para constancia firman

Tcrn. Marcelo Gómez	Dr. Jorge Carvajal
Decano	Secretario Académico

Denyss Estévez

Autor