UNIVERSIDAD DON BOSCO
FACULTAD DE INGENIERÍA
ESCUELA DE ELECTRÓNICA

TEMA:
DESARROLLO DE UN HISTORIADOR DE PROCESOS CLIENTE OPC DATA ACCESS

TRABAJO DE GRADUACIÓN PARA OPTAR AL TÍTULO DE INGENIERO EN AUTOMATIZACIÓN

PRESENTADO POR:
JOSUÉ ANÍBAL CAMPOS ALFARO
HÉCTOR OVIDIO TOVAR CASTRO

NOMBRE DEL ASESOR:
LIC. SANTIAGO ABARCA

NOVIEMBRE 2006

SOYAPANGO – SAN SALVADOR – EL SALVADOR

UNIVERSIDAD DON BOSCO
FACULTAD DE INGENIERÍA

AUTORIDADES

ING. FEDERICO MIGUEL HUGUET RIVERA
RECTOR

PBRO. VÍCTOR BERMUDEZ YÁNEZ
VICERRECTOR ACADÉMICO

LIC. MARIO OLMOS ARGUETA
SECRETARIO GENERAL

ING. ERNESTO GODOFREDO GIRÓN
DECANO FACULTAD DE INGENIERÍA

UNIVERSIDAD DON BOSCO
TRIBUNAL EXAMINADOR

LIC. SANTIAGO ABARCA

ASESOR

ING. ALEXANDER GUZMÁN

JURADO

ING. ERICK BLANCO

JURADO

ING. EDGARDO CRUZ ZELEDÓN

JURADO

ÍNDICE

I. DEFINICIÓN DEL TEMA

El proyecto consiste en el desarrollo de un historiador de procesos cliente *OPC DATA ACCESS*, una aplicación que se encargará de almacenar de manera histórica, datos o valores de un proceso. La fuente de los datos será un servidor *OPC*[1], el cual a su vez obtendrá sus datos de un *PLC*[2] o *DCS*[3]. Además del almacenamiento histórico, la aplicación contará también con herramientas para la creación de consultas, diagramas de tendencias y reportes.

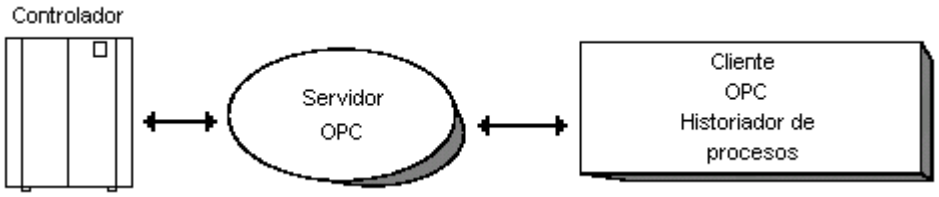


Figura 1: Los clientes OPC obtienen sus datos de un servidor OPC.

[1] OPC (*OLE for Process Control*) es un OLE que trata los datos como colecciones de objetos para ser compartidos por aplicaciones que soportan las especificaciones OLE. OPC provee extensiones para OLE para soportar el intercambio de datos en el control de procesos. OLE (*Object Linking and Embedding*) es un estándar desarrollado por Microsoft, que permite crear objetos en una aplicación y luego insertarlos en otra aplicación compartiendo información entre las aplicaciones.

[2] Controlador lógico programable (*Programmable Logic Controller*).

[3] Sistema de control distribuido (*Distributed Control System*).

II. PLANTEAMIENTO DEL PROBLEMA

Los valores de planta que son obtenidos en tiempo real son importantes, ya que permiten monitorearla y controlarla. Sin embargo, estos datos son poco útiles para realizar análisis de producción o para determinar causas de fallas en el producto terminado, si no se almacenaran de forma histórica. Y esta ausencia de información histórica constituye la problemática a solucionar.

Para dar un ejemplo de los problemas a los que puede dar solución un historiador, mencionaremos lo sucedido a una empresa que fabrica bicicletas. Esta empresa produce dos tipos de bicicletas: estándar y de alta calidad; ambas en dos colores diferentes: azul y rojo. Las diferencias entre ambos estilos se basan sólo en la estructura, aunque las dos usan los mismos componentes.

La fábrica comprende las siguientes áreas:

- Maquinaria y fabricación
- Pintura
- Sub-ensamblaje
- Ensamblaje final y transporte

Además existe dentro de la fábrica una máquina que se encarga de hacer los asientos de las bicicletas. Esta máquina entrega los asientos sólo para ensamblarlos a las bicicletas ya terminadas.

En el área de maquinaria y fabricación se hace la manufactura de la estructura, los soportes del asiento, las barras frontales, y los soportes de las ruedas. Estos componentes son producidos procesando materia prima a través de células[4] de trabajo los cuales se encargan de cortar, doblar y soldar. Luego de que las partes están terminadas, las estructuras se agrupan en lotes de producción y se mandan al

[4] Definición de célula de trabajo Filosofía de fabricación que intenta identificar partes similares y agruparlas juntas en familias para lograr las ventajas de sus similitudes en el diseño y manufactura de las mismas.

área de pintura, mientras los otros componentes van a inventario donde pueden sacarse de almacén para suplir la demanda en el área de ensamblaje. En la próxima estación, cada bicicleta es ensamblada en base al pedido del cliente con partes suplidas del inventario. En el último paso las bicicletas, con sus ruedas y barras de manos removidas, se empacan en una caja de cartón corrugado para ser entregadas.

La máquina que hace los asientos de la bicicleta, opera una maquinaria interna de inyección de moldes para formar la parte acolchonada del asiento sobre el soporte respectivo.

Cuando la máquina estaba siendo probada, se produjo un lote piloto de asientos. Al mismo tiempo un historiador almacenó varios parámetros, como la temperatura y presión de la máquina y se compararon contra las especificaciones de diseño de los asientos.

Unos pocos meses después, se reportó un aumento en el rango de asientos con fallas, y la empresa sufrió varias devoluciones a causa de asientos defectuosos. Desafortunadamente el problema era intermitente y latente.

El historiador se usó para analizar los parámetros de producción contra las muestras pilotos, y se encontraron con una variación ocasional en los perfiles de temperatura y presión de la máquina cuando la temperatura de la materia prima era extremadamente baja, como en el caso de los meses de invierno.

Se identifico la causa principal de la falla, de esa manera la empresa fue capaz de considerar en el control de su proceso esa fluctuación en la temperatura ambiente, lo que les permitió eliminar con facilidad la inconsistencia en la calidad de sus productos. Como resultado, los rangos de asientos con fallas fueron extremadamente reducidos, y las devoluciones ya no ocurrieron.

III. DESCRIPCIÓN DE LA PROPUESTA DEL PROYECTO

Se propone realizar una aplicación para el sistema operativo Windows, en el lenguaje de programación Visual Basic .NET y con el manejador de base de datos SQL Server.

La aplicación, llamada ***"historiador de procesos"***, se someterá al estándar OPC Data Access; estándar que en general define la forma de escritura y lectura de datos en tiempo real entre el cliente y el servidor. Por lo tanto podrá acceder a datos de cualquier servidor OPC Data Access. Además, se agregarán las siguientes herramientas de análisis de datos:

- Consultas.
- Reportes.
- Diagramas de tendencias.

Las consultas servirán para visualizar datos parametrizados de la base de datos.

Los reportes servirán para crear reportes con los valores, tiempo de estampado y calidad, del ***tag***[5] configurado. También se podrá agregar su respectivo diagrama de tendencia.

Los diagramas de tendencia servirán para crear diagramas del valor del tag deseado, con respecto al tiempo. Podrán crearse también diagramas de múltiples tag con respecto al tiempo.

El estándar OPC incluye documentación para el desarrollo de aplicaciones cliente y de aplicaciones servidor. Sin embargo, en este trabajo de graduación sólo se hace uso de la parte relacionada con el desarrollo de aplicaciones cliente. No obstante, en los anexos se describe brevemente la otra parte del estándar, para efectos de que sirva como fuente de consulta.

[5] Etiqueta.

IV. IMPORTANCIA DE LA INVESTIGACIÓN

En la actualidad, gran parte de la industria utiliza procesos automatizados para elaborar sus productos finales. Estos procesos de producción se rigen rigurosamente por factores como control de calidad, eficiencia y mejora de la productividad. Dichos factores tienen un fuerte impacto en el aspecto financiero de la empresa, por lo que la supervisión y el monitoreo de la planta es sumamente importante. Hoy en día se utilizan diferentes formas para llevar a cabo estas tareas, por ejemplo una de ellas es haciendo uso de componentes de Software y Hardware de LabView; otra se basa en tecnología OPC, la cual define aplicaciones cliente y servidor de manera estandarizada. Esta última no requiere componentes de hardware adicional, lo que la convierte en una opción muy deseable, razón por la cual se decidió hacer uso de ella. Un servidor OPC se asemeja a una fuente de datos, en la cual una de sus principales características es la de obtener en tiempo real los datos que están siendo procesados por un dispositivo de control, datos provenientes de lecturas análogas/digitales de temperatura, presión, nivel, sensores discretos, interruptores de nivel, etc.

Por otra parte, un cliente OPC trabaja con los datos que toma el servidor, y éste suele orientarse a diferentes aplicaciones como: interfaces humano-máquina (*HMI*[6]), historiadores de procesos, diagramadores de tendencias, generadores de reportes, bases de datos, entre otras.

Un historiador de procesos es usualmente utilizado en la industria para almacenar datos. Pero, para poder hacer aun más efectiva y útil esta colección de datos debe ser posible construir reportes cuando se desee; también es deseable elaborar una base de datos que pueda ser de gran utilidad para el análisis y el estudio de la planta.

Es por ello que muchos administradores, ingenieros y supervisores de plantas se apoyan en historiadores de procesos para poder determinar la variabilidad del

[6] Interfaz hombre-máquina (*Human Machine Interface*).

proceso y las causas que lo originan. La variabilidad se refiere a la inestabilidad de los parámetros de interés de la planta, lo que puede conducir a que los sistemas de control no operen correctamente. El historiador le permite al ingeniero el análisis de dicha variabilidad, lo cual repercute en un mejor control y una reducción de costos.

Es necesario mencionar que un historiador de procesos se enfoca en coleccionar datos; sin embargo resulta atractivo y útil construir herramientas que ayuden a analizar los mismos. La ventaja de esto es que no es necesario adquirir otros paquetes de software para hacer un análisis posterior utilizando la base de datos.

Los beneficios que se adquieren al desarrollar un historiador de procesos OPC Data Access son notorios ya que con esta herramienta se contribuye a identificar fluctuaciones en los procesos de producción. Es por ello que el desarrollo o adquisición de un historiador de puede llegar a ser vital en una empresa al momento de determinar factores como los mencionados anteriormente.

V. OBJETIVOS

V.1. OBJETIVO GENERAL

Desarrollar una aplicación para el sistema operativo Windows para el almacenamiento y análisis de datos históricos de procesos industriales, provenientes de un servidor OPC conectado a un PLC o DCS.

V.2. OBJETIVOS ESPECÍFICOS

- Desarrollar una aplicación cliente OPC Data Access.
- Almacenar históricamente datos provenientes de un servidor OPC Data Access.
- Convertir a escala los valores de los datos provenientes del servidor.
- Crear consultas para la visualización y análisis de los datos almacenados.
- Crear reportes parametrizados que permitan analizar la información de la base de datos.
- Realizar diagramas de tendencias de los valores almacenados con respecto al tiempo.

VI. ALCANCES

- La aplicación cliente estará apegada al estándar OPC Data Access y será desarrollada en el lenguaje Microsoft Visual Basic Express 2005.
- La base de datos estará implementada en Microsoft SQL Server EXPRESS 2005.
- Los datos proporcionados en tiempo real serán almacenados históricamente.
- Los valores a almacenarse podrán escalarse.
- Se desarrollará una consulta parametrizada, la cual desplegará información del nombre de la variable, la descripción, el intervalo de tiempo de la consulta, así como el valor, el tiempo de estampado y la calidad de los datos. En esta consulta se indicará el tag a consultar, así como también el intervalo de tiempo al que pertenecerán los datos de la consulta. Por ejemplo, si se configura una consulta del tag llamado "`Temperatura_1`", entre el intervalo de tiempo que corresponde desde el mes de enero de 2005 hasta el mes de diciembre de 2005, la consulta desplegará los valores correspondientes al tag, junto con sus respectivos tiempos de estampado y la calidad de adquisición de cada dato.
- Los datos de las consultas podrán exportarse a Microsoft Excel.
- La información presentada en los reportes será la siguiente: nombre de la variable, descripción, intervalo de tiempo configurado, así como el valor, el tiempo de estampado y la calidad de los datos. Opcionalmente podrán agregarse al reporte el título, el logo de la empresa y el diagrama de tendencia correspondiente.
- La aplicación generará los siguientes reportes:
- Reportes programados, los cuales podrán ser configurados para generarse diariamente, semanalmente, o mensualmente.
- Reportes no programados, los cuales podrán ser configurados para generarse inmediatamente, es decir en cualquier momento cuando el usuario lo desee.

- Reportes de la consulta actual, los cuales se apoyarán en los valores de configuración de la consulta para realizar el reporte.
- Se podrán realizar los siguientes diagramas de tendencias con respecto al tiempo:
- De un solo Tag, llamados `UniTag`, los cuales graficarán el valor de una única Tag con respecto al tiempo.
- De varias Tag, llamados `MultiTag`, los cuales graficarán el valor de varias Tag con respecto al tiempo haciendo uso de un único par de ejes.

VII. LIMITACIONES

- No pueden realizarse acercamientos ni alejamientos en los diagramas de tendencia.
- No pueden hacerse diagramas de tendencias en tiempo real.

VIII. VALIDACIÓN DE RESULTADOS

La validación de resultados se hará de dos maneras:

- Con un servidor OPC DA[7] que posee un simulador:

Se instalarán en una computadora el servidor OPC DA de cualquier fabricante y el historiador de procesos cliente OPC DA. El historiador será configurado para almacenar los datos simulados por el servidor. Posteriormente se harán pruebas de todas las herramientas incluidas en la aplicación.

- Con un servidor OPC DA y dos Controladores Lógicos Programables:

Se instalarán en una computadora el servidor OPC DA de cualquier fabricante y el historiador de procesos cliente OPC DA. La computadora que se utilice debe soportar comunicación con el PLC que se conectará. El servidor será configurado para acceder a los datos de un PLC correspondientes a sus entradas, salidas y variables internas. El historiador será configurado para almacenar datos provenientes del servidor. Posteriormente se harán pruebas de todas las herramientas incluidas en la aplicación.

La prueba se repetirá para el otro modelo de PLC.

[7] Abreviación de *Data Access*.

IX. MARCO TEÓRICO

IX.1. ¿QUÉ ES OLE?

OLE hace referencia a las siglas "*Object Linking and Embedding*". Es un protocolo desarrollado por Microsoft, que permite incorporar parte de un documento desarrollado en una aplicación dentro de otra aplicación, capaz de mantener enlaces activos entre dos documentos o incluso implantar un tipo de documentos en otro. En 1996, Microsoft renombró la tecnología OLE 2.0 como *ActiveX*[8].

IX.2. ¿QUÉ ES OPC?

OPC hace referencia a las siglas "*OLE for Process Control*". En otras palabras, es la tecnología OLE de Microsoft adaptada al control de procesos. El OPC nace en 1996 cuando el grupo antecesor de la Fundación OPC, formado por las empresas Fisher-Rosemount, Rockwell Software, Opto 22, Intellution e Intuitive Technology, fue capaz de desarrollar una especificación básica en un solo año de trabajo. Esta especificación estaba basada en las tecnologías OLE/COM y DCOM, y fue nombrado como "Especificación OPC".

El OPC fue diseñado para hacer puente entre aplicaciones basadas en Windows y aplicaciones de hardware y software para el control de procesos. Es un estándar abierto que permite un método consistente de acceso a datos de campo desde los dispositivos de la planta.

Ya son 7 los estándares desarrollados por la Fundación OPC y dirigidos a vencer todas las barreras de la comunicación en todos sus niveles. Estos siete estándares se nombran a continuación:

- OPC Data Access

  Utilizado para mover datos en tiempo real desde PLC, DCS, y otros dispositivos de control hacia HMI y otros clientes de visualización. La última versión de la

[8] Componente de software que se puede insertar en una página Web para ofrecer una funcionalidad que no está disponible en HTML.

especificación es la 3.0, la cual se apoya en versiones anteriores mientras mejora sus capacidades de búsqueda e incorpora el esquema *XML*[9]-Data Access.

- OPC Alarms & Events

  Provee notificaciones demandadas de alarmas y eventos. Esto incluye alarmas de procesos, acciones de operador, mensajes de información y mensajes auditivos.

- OPC Batch

  Esta especificación lleva la filosofía de OPC hacia las necesidades específicas de los procesos de producción. Provee interfaces para el intercambio de capacidades de equipo y condiciones de operación actual.

- OPC Data Exchange

  Esta especificación comunica a un servidor con otro servidor a través de redes Ethernet/*Field bus*[10]. Esto provee interoperabilidad entre muchos vendedores, agrega configuraciones remotas y servicios de diagnósticos, monitoreo y administración.

- OPC Historical Data Access

  Donde el OPC Data Access provee acceso en tiempo real, el OPC Historical Data Access provee acceso a datos previamente almacenados. Desde un simple sistema serial para el registro de datos hasta un sistema SCADA[11] complejo, los archivos históricos pueden ser recuperados de una manera uniforme.

- OPC Security

  Todos los servidores OPC proveen información que es valiosa para la empresa y si no es actualizada correctamente, podría tener consecuencias significativas para los procesos de planta. El OPC Security especifica cómo controlar el acceso de los clientes a estos servidores para proteger la información importante y como protegerse contra las modificaciones no autorizadas de los parámetros del proceso.

[9] Lenguaje de marcas ampliable (*Extensible Markup Language*).

[10] Tipo de red industrial para el control distribuido en tiempo real.

[11] Supervisión de Control y Adquisición de Datos (Supervisory Control and Data Adquisition).

- OPC XML-DA

  Provee reglas flexibles y consistentes, y formatos para exponer datos de la planta usando XML, apoyándose en el trabajo hecho por Microsoft y otros sobre SOAP[12] y los servicios Web.

  La Fundación OPC esta trabajando actualmente en dos estándares más llamados OPC Complex Data y OPC Commands.

- OPC Complex Data

  Una especificación compañera del OPC DA y OPC XML-DA que permite a los servidores exponer y describir tipos de datos complicados, así como estructuras binarias y documentos XML.

- OPC Commands

  Un nuevo conjunto de interfaces, para servidores y clientes OPC, que definen comandos de identificación, envío y monitoreo, los cuales se ejecutan sobre un dispositivo.

IX.3. ANTECEDENTES DEL ESTÁNDAR OPC DATA ACCESS

En tiempos anteriores al primer estándar llamado “Especificación OPC” ,ahora OPC Data Access, la comunicación entre los sistemas de control y los paquetes de software para monitoreo, control y análisis de datos de dichos sistemas, tales como paquetes HMI/*SCADA*, bases de datos y hojas de cálculo, era un proceso muy complejo desde la perspectiva del usuario. En primer lugar, se debía tener mucho cuidado a la hora de adquirir equipos de control y paquetes de software, porque debían ser compatibles para poderse comunicar, siendo preferible incluso que el equipo y el software fueran del mismo fabricante. En caso de tomar una decisión incorrecta a la hora de comprar el software, el usuario tenía la opción de buscar un paquete de software extra, generalmente desarrollado por el mismo fabricante del software, llamado “*driver*”[13] para poder comunicarse. Esto significaba gastos

[12] Protocolo simple de acceso a objetos (*Simple Object Access Protocol*).

[13] Controlador de dispositivo.

adicionales debido a que este componente difícilmente venía incluido en el paquete original.

Por parte del fabricante la dificultad radicaba precisamente en la creación de drivers, debido a que eran muchas las marcas y tipos de dispositivos de control a comunicar. Además, si ya se poseía un driver para comunicar un dispositivo controlador determinado, este no garantizaba su funcionamiento con modelos mejorados del mismo dispositivo. Los fabricantes de software entonces se enfocaban en aumentar sus características de comunicación y no en las cualidades principales del programa.

La mejor comparación de esta situación puede ser lo sucedido con las impresoras antes de que los sistemas operativos las soportaran. En los tiempos del MS DOS los desarrolladores de cada aplicación o paquetes de software debían desarrollar también drivers para cada impresora en el mercado. De esta forma los fabricantes de paquetes de software como AutoCAD y WordPerfect desarrollaron sus aplicaciones y también sus drivers. Tuvieron que escribir drivers separados para cada modelo de impresora que quisieran soportar porque un solo driver no soportaba más de un modelo de impresora.

Luego aparece Windows y soluciona este problema incorporando el soporte de impresoras al sistema operativo. Ahora un solo driver sirve para todas las aplicaciones de Windows y quien debe de escribirlo es el fabricante del impresor, no el desarrollador de la aplicación.

En el caso de la automatización, Wonderware, Rockwell, Intellution, entre otros, tuvieron que desarrollar su paquete de software HMI y un driver propietario para cada dispositivo. A partir de 1996, con el nacimiento del OPC, los mismos fabricantes de los dispositivos de control pueden desarrollar sus servidores OPC DA y sus paquetes de software como HMI’s pueden convertirse en aplicaciones cliente a dichos servidores. La ganancia para el usuario final es la flexibilidad, ya que ahora puede adquirir paquetes de software basándose en sus características principales y no en conectividad.

IX.4. APLICACIONES DEL ESTÁNDAR OPC DATA ACCESS

IX.4.1. APLICACIONES HMI / SCADA

El desarrollo del estándar OPC DA ha permitido el avance en sistemas de monitoreo y control tal como el sistema SCADA. Este sistema tiene por finalidad adquirir datos como le sea posible para el monitoreo de la planta, de manera que pueda generar realimentación útil, como por ejemplo: el valor actual de sensores, alarmas y fallas, de manera que ayude a fortalecer el sistema de control. Los sistemas SCADA se caracterizan por que constituyen el complemento indispensable de los sistemas de control, el cual se encuentra constituido por dispositivos como PLC, adquisidores de datos, Unidades Terminales Remotas (RTU) y los instrumentos inteligentes tales como analizadores eléctricos, químicos y caudalímetros.

Es por ello que hoy en día los sistemas SCADA son comunes en la industria donde los desarrolladores de este tipo de sistemas se enfocan principalmente en los siguientes aspectos: integración con el hardware, facilidad de uso, aplicación de herramientas innovadoras y apertura de comunicaciones.

IX.4.2. HISTORIADORES DE PROCESOS

Por otra parte el estándar ha promovido el desarrollo de historiadores de procesos, generadores de reportes, diagramadores de tendencias, entre otras aplicaciones.

No se sabe exactamente cuándo, dónde o quién desarrolló por primera vez un historiador de procesos o en otras palabras una aplicación para la industria que almacenara datos o valores de procesos de forma histórica. Pero lo que si se puede asegurar es la razón por la cual nacieron; se necesitaba almacenar datos reales de los dispositivos de planta para su posterior análisis. Hoy en día, empresas fuertes en la automatización industrial como Rockwell Software, Matrikon y Kepware, fabrican historiadores de procesos que varían en complejidad y en precios. Los más básicos simplemente almacenan los datos en una base de datos propietaria y éstos pueden

ser vistos sólo desde la aplicación. Otros ofrecen además de herramientas poderosas de análisis, la opción de crear funciones extras para adecuar la aplicación a las necesidades de la empresa. Muchos también ofrecen comunicación con otras bases de datos como MySQL, MS SQL Server y Oracle.

IX.5. ¿QUÉ ES UNA BASE DE DATOS?[14]

Como su nombre lo indica una Base de Datos es un lugar donde se puede almacenar información, prácticamente se refiere a una colección de datos que son almacenados y que en un momento posterior serán recuperados. Algo muy importante es que los datos que son almacenados en la base de datos pueden ser de diferentes tipos, tales como cadenas de texto, que pueden representar el nombre de personas o lugares de un lugar en particular; o números reales, que sirven como identificadores o que simplemente representan el nombre de alguien o algo, como es el caso del ItemID, el cual sirve para identificar una entrada/salida de un PLC o algún otro dispositivo. Como una base de datos es un archivo independiente de un lenguaje de programación específico, muchos programas se pueden conectar a esa base de datos con el fin ver, insertar, actualizar o borrar datos.

Una base de datos prácticamente esta formada por una o más tablas, esto depende de de la aplicación, a la vez cada tabla contiene campos. Una base de datos posee un nombre propio y las tablas que lo conforman también.

IX.5.1. ¿QUÉ ES UNA RDBMS?

Una Relational Database Management System (RDBMS) es un intermediario para manipular la información que contiene una base de datos, el RDBMS recibe instrucciones sobre acciones que se desea realizar como inserción, actualización y borrado de filas. El RDBMS es el encargado de traducir las instrucciones en acciones concretas: creación de tablas, extracción, borrado e inserción de filas. También se le conoce como servidores de datos [véase 44.i]. Para este caso particular se esta utilizando SQL Server Express 2005 como RDBMS.

[14] Adaptado de [XXXX].

IX.5.2. ESTRUCTURA FÍSICA DE UNA BASE DE DATOS

Al momento de crear una base de datos nueva se crean dos archivos como mínimo en el disco. Uno de ellos se le conoce como archivo principal que es el que contiene los datos y un archivo de registro de transacciones.

El archivo de datos contiene los datos, como otra información, como procedimientos almacenados, relaciones… etc.

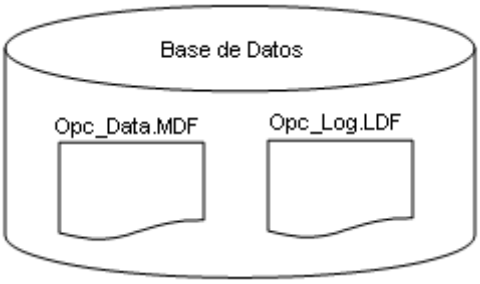
El archivo de registro de transacciones se encarga de conservar la base de datos en un estado integro lo cual facilita la recuperación en caso de ser necesario ya que contiene los cambios que se levan a cabo en la base de datos.

IX.5.3. NOMBRES DE ARCHIVO

El nombre de las base de datos como se menciono previamente poseen un nombre propio, y son definidas por el operador que las crea y estas utilizan extensiones fijas, como ejemplo ilustrativo nuestra base de datos tiene el siguiente nombre:

- Nombre de la base de datos: `Opc`
- Nombre del archivo de datos creado: `Opc_Data.MDF`
- Nombre del archivo de registro creado: `Opc_Log.LDF`

La siguiente figura muestra una representación grafica de la estructura física de la base de datos que ha sido creada en nuestra aplicación.



IX.5.4. ESTRUCTURA LÓGICA DE LAS BASE DE DATOS

La estructura lógica de la base de datos o *Enterprise Manager*, en el caso de SQL, nos permite conocer cada unas de las partes existentes de las mismas, por citar un ejemplo la estructura lógica muestra las tablas, Relaciones existentes “si las hay”, procedimientos almacenados de una base de datos en particular. Prácticamente nos permite visualizar sus elementos.

La estructura lógica de una Base de Datos contiene:

Tablas: Es la entidad básica de almacenamiento de información en una base de datos. Todos los elementos que se pueden añadir ya sea desde una aplicación o desde otra herramienta se almacenan en tablas. Las tablas están formadas por filas o registros, y columnas. En cada columna existen datos del mismo tipo y en cada fila hay un representante de cada columna.

Procedimientos Almacenados: Son procedimientos escritos en un lenguaje llamado TRANSACT-SQL, los cuales permiten crear tablas, borrar, insertar y actualizar los datos de las tablas. Todo mediante Código.

Otros elementos lógicos: Además de tablas y procedimientos almacenados existen vistas y relaciones, la primera corresponde a consultas realizadas en la base de datos y la segunda es como relacionar las diferentes tablas que pertenecen a una base de datos en específica.

IX.6. ACCESO DE DATOS[15]

Una necesidad indispensable en la mayoría de aplicaciones, independientemente cual sea la finalidad de estas, es el acceso a Base de Datos, Un Historiador no es la excepción, ya que los datos que lee el Cliente OPC del Servidor OPC se almacenan en una base de datos, para realizar posteriormente consultas, diagramas de tendencia y reportes.

El acceso a datos contenidos en bases de datos no se limita únicamente a su acceso si que al almacenamiento o inserción, básicamente el historiador que se esta realizando posee una fuente de datos RDBMS como se esta trabajando con SQL Server Express 2005.

En la actualidad se ha simplificado el acceso de datos desde los lenguajes de programación más usuales tales como C ++, Java Script, Visual Basic entre otros, considerando que las soluciones son de diversas maneras, generalmente dependiente del fabricante, sistema operativo e incluso lenguaje de programación.

Como se está trabajando bajo el ambiente Windows, cabe mencionar que originalmente este sistema operativo no incluía mecanismo alguno para facilitar el acceso de datos[16]. Esto causó que algunos fabricantes junto con sus herramientas de desarrollo incluyeran productos propios para el tratamiento y manipulación de datos. Así que Visual Basic incluía un mecanismo llamado Data Access Objects (DAO).

Con la aparición de Windows 2000, el mismo sistema operativo incorporaba los mecanismos necesarios para acceder a las base de datos. Los mecanismos eran los controladores OLEDB para sistemas RDBMS y ActiveX Data Objects (ADO) como interfaz de alto nivel para comunicarse con estos controladores. La ventaja de estas interfaces es que todas las herramientas de desarrollo que trabaje bajo Windows 2000 ó superior pueden ser utilizadas para comunicarse con base de datos. Con la

[15] Adaptado de [XX].

[16] [véase [XX], pág. 564].

llegada de la plataforma .NET trae consigo la evolución de los mecanismos para el acceso de datos lo que produce ADO.NET.

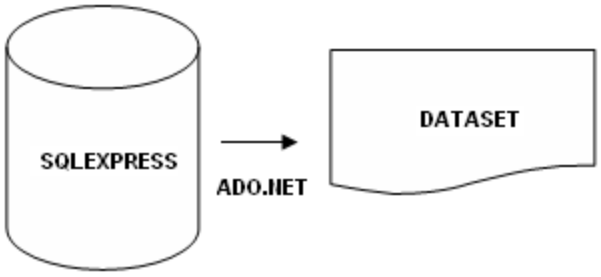
ADO.Net trae consigo un conjunto de objetos más avanzados que simplifica la construcción de aplicaciones en múltiples niveles, al facilitar el trabajo sobre los diferentes conjuntos de datos.

Por lo que la inserción de datos a la base de datos del historiador se realiza con ADO.Net mediante Visual Basic .NET.

IX.7. COMPONENTE `DATASET`[17]

Básicamente el `DataSet` es un adaptador el cual tiene por función conectarse a una fuente de datos, recuperando un conjunto de datos y cerrando la conexión. Prácticamente el `DataSet` seria una copia de ese conjunto de datos el cual puede ser transferido entre aplicaciones y puede operarse sobre el como si fuese la Base de Datos real. Una de las principales ventajas de utilizar el Objeto `DataSet` es que no requiere una conexión continua con una fuente de Datos.

En la siguiente figura se presenta un esquema representativo, de utilizar esta forma de conexión.



Para las consultas realizadas en esta aplicación, los datos únicamente se transfieren es una sola dirección. Es necesario mencionar que se podría realizar el proceso inverso, es decir, del objeto `DataSet` hacia la Base de Datos; sin embargo esto no se implementó al momento de realizar consultas, debido a que la base de

[17] Adaptado de [XX].

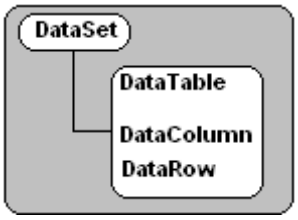
datos no se actualiza en dicho proceso. El DataSet es un objeto muy completo ya que prácticamente es una copia de la base de datos, ya que al existir más de una tabla guarda hasta las relaciones entre ellas.

IX.7.1. ESTRUCTURA DE UN OBJETO DATASET

Un objeto DataSet facilita el trabajo con una Base de Datos, esto sin necesidad de mantener una conexión continua. Un DataSet prácticamente es una representación en memoria de una Base de Datos, hay que aclarar que este objeto puede contener el número de tablas que contiene la Base de Datos así como las relaciones y las reglas de validación, permitiendo trabajar como si fuera en la propia base de datos[tomado de 5, ver ii].

Internamente el DataSet contiene una colección de tablas, representadas por un objeto DataTable. Cada Tabla puede incluso provenir de Base de Datos diferentes. Además el objeto DataTable contiene DataColumn y DataRow que también son objetos, de esta manera es que se representan las columnas y las filas de una tabla.

La siguiente figura muestra ilustra la estructura simplificada de un Objeto DataSet.



IX.7.2. CREACIÓN Y RECUPERACIÓN DE UN OBJETO DATASET

Para crear un objeto DataSet, se realiza el mismo procedimiento de crear cualquier otra variable, al crearlo se tendría un objeto DataSet vacío por lo que se necesitaría posteriormente establecer una conexión a la fuente de datos con la que se desea establecer conexión y de esa manera llenar el DataSet con datos propios de la Base de Datos. Como se esta trabajando con el componente DataSet de

ADO.NET, para la conexión se utilizara el objeto de conexión DataAdapter y como se con SQL Express sería SqlDataAdapter, este objeto pertenece a la librería system.Data.SqlClient, Este objeto de conexión permite realizar la conexión y la consulta en la misma sentencia, ya que recibe por parámetros la sentencia de consulta y la cadena de conexión. Una vez establecida la conexión bastaría utilizar el método Fill(), de SqlDataAdapter para llenar el resultado de la consulta en el DaaSet.

X. UTILIZACIÓN DEL ESTÁNDAR

En este capítulo se resume cómo se utiliza el estándar “OPC Data Access Automación Interface” para realizar la aplicación. Para mayor referencia sobre los detalles de la especificación del estándar, consúltese el Anexo A.

Aunque el estándar va más allá de una simple descripción de las propiedades, métodos y eventos de los objetos de la interfaz, todavía deja mucho qué desear a la hora de mostrar un ejemplo más cercano a la realidad que nos permita comprender, de mejor manera, cómo usar esta poderosa herramienta. Sin embargo, se recomienda leerlo para poder entender la naturaleza de los objetos de la interfaz.

También se recomienda revisar este capítulo simultáneamente al código de las clases `ServidorOPC`, `BrowserOPC`, `GrupoOPC` e `ItemOPC` que viene como anexo.

La interfaz de automatización también es limitada a la hora de proporcionar herramientas para realizar una aplicación verdaderamente dinámica. Esto nos llevó a la reutilización de la interfaz para crear cuatro nuevas clases, que son la base de la aplicación.

X.1. UTILIDADES DEL OBJETO `OPCSERVER` EN LA APLICACIÓN.

Antes de describir la utilidad de todas las propiedades del objeto `OPCServer` que son reutilizadas en la aplicación, cabe mencionar que la mayoría de ellas son utilizadas para brindar información al usuario del servidor.

Del objeto `OPCServer` se reutilizaron las siguientes propiedades:

Propiedad	**Utilidad**
`StartTime`	Es la propiedad con la cual se sabe la hora en la que se inició el servidor.
`CurrentTime`	Es de utilidad para realizar una comparación con la propiedad `TimeStamp` del objeto `OPCItem`.

Propiedad	Utilidad
LastUpdateTime	Brinda al usuario información de la última actualización del servidor hecha por el fabricante.
MajorVersion	Brinda al usuario información de la versión del servidor.
MinorVersion	Provee al usuario información de la versión del servidor. Concatenándola con la propiedad anterior se obtiene la versión completa del servidor.
BuildNumber	Brinda al usuario información del fabricante del servidor.
VendorInfo	También brinda información del fabricante del servidor.
ServerState	Brinda al usuario información del estado del servidor.
LocaleID	Es de utilidad al usuario al momento de depurar errores en la comunicación entre cliente-servidor.
Bandwidth	Indica qué tan saturada puede estar la comunicación entre cliente-servidor.
OPCGroups	Propiedad básica del servidor. De mucha utilidad, ya que es la colección a la cual pertenecerán todos los grupos que se creen en la aplicación.
ServerName	Importantísima al momento de definir la aplicación servidor en el cliente. Si no se conoce esta propiedad o se escribe de manera incorrecta, la comunicación entre cliente-servidor no se daría.

Los métodos reutilizados del objeto `OPCServer` fueron los siguientes:

Método	Utilidad
GetOPCServers	Esta propiedad hace innecesario memorizar el nombre del servidor, para que éste sea definido en el cliente. En la aplicación se utiliza para desplegar una lista seleccionable de posibles servidores.
Connect	Único método que permite la conexión directa con el servidor.
Disconnect	Único método que permite la desconexión del servidor.

Método	Utilidad
`CreateBrowser`	Este método es de suma importancia si se quiere realizar una aplicación dinámica. En la aplicación se utiliza para facilitarle al usuario la búsqueda de ítems creados en el servidor.
`GetErrorString`	De suma utilidad para la depuración de errores por parte del usuario. En la aplicación se utiliza para que cuando se realice un evento que cause error, la descripción de éste será capturada y luego desplegada.

X.2. UTILIDADES DEL OBJETO `OPCBROWSER` EN LA APLICACIÓN

El objeto `OPCBrowser` es de suma utilidad si se desea realizar una aplicación dinámica. Si en la configuración del servidor se creara solamente un canal, un dispositivo, y uno, dos o tres ítems, no habría dificultad a la hora realizar la conexión del cliente con los ítems creados. El usuario podría incluso memorizar los nombres de los ítems y no tendría problemas al crear la conexión con el servidor. Pero si el número de canales aumenta a cinco, el número de dispositivos por canal aumenta a diez, y ahora, en lugar de ser 3 ítems, son de veinte a treinta ítems por dispositivo, lo que equivaldría a 1000 ó 1500 ítems configurados en el servidor, y resultaría sumamente complicado encontrar un único ítem para realizar la conexión.

El objeto `OPCBrowser` hace que el problema de encontrar ese ítem entre los 1499 restantes sea una tarea fácil.

A continuación se describen las utilidades de las propiedades del objeto `OPCBrowser` utilizadas en la aplicación.

Propiedad	Utilidad
Organization	En la aplicación se lee primero esta propiedad antes de llamar al objeto TreeView que desplegará al objeto OPCBrowser, de esta forma se sabe de que forma se debe desplegar.
DataType	Importante propiedad que permite filtrar los ítems configurados en servidor por el tipo de dato al que pertenecen.
Count	Esta propiedad aparece en todas las colecciones de la interfaz. En la aplicación es muy utilizada como parámetro de comparación en bucles.

Los métodos del objeto OPCBrowser reutilizados en la aplicación con su respectiva utilidad son los siguientes:

Método	Utilidad
Item	Al igual que la propiedad Count, este método aparece en todas las colecciones de la interfaz. En la aplicación es utilizado para referenciar a los elementos de la colección del Browser, por medio de un índice.
ShowBranches	En la aplicación este método es utilizado para que el usuario pueda encontrar la configuración de canales y dispositivos del servidor.
ShowLeafs	En la aplicación este método es utilizado para que el usuario colocado ya en un Branch, pueda encontrar los ítems configurados en él.
MoveDown	Se utiliza para que el usuario pueda desplazarse una posición inferior en la colección presentada por el objeto OPCBrowser.
GetItemID	Se utiliza para que cuando el usuario se posicione sobre un Leaf automáticamente se despliegue el identificador de ítem correspondiente.
MoveToRoot	Se utiliza para que el usuario pueda

	desplazarse hasta la raíz del `Browser` que en la mayoría de casos será el servidor.

X.3. UTILIDADES DEL OBJETO `OPCGROUPS` EN LA APLICACIÓN

El objeto `OPCGroups` es una propiedad del objeto `OPCServer` y por lo tanto es dependiente del mismo, es decir, que debe existir antes un objeto `OPCServer` para que haya un objeto `OPCGroups`. No podría crearse un objeto `OPCGroups` si no existiera antes uno `OPCServer`.

Las propiedades y métodos reutilizados de este objeto son incorporados a la clase `ServidorOPC`, para que, de esa manera, en un solo objeto se manejen los miembros de dos objetos de la interfaz; `OPCServer` y `OPCGroups`.

En este objeto, las propiedades reutilizadas son las siguientes:

Propiedad	**Utilidad**
`Parent`	Esta propiedad facilita bastante la referencia al servidor al que pertenece un grupo. En la aplicación aparece siempre que sea necesario hacer referencia a un servidor a partir de un grupo.
`DefaultGroupIsActive`	En la aplicación se utiliza para establecer el estado de un grupo recién creado, sin necesidad de que el usuario lo establezca directamente.
`DefaultGroupUpdateRate`	Al igual que la propiedad anterior, esta propiedad es utilizada en la aplicación, para establecer el rango de actualización de un grupo recién creado, sin necesidad de que el usuario lo establezca directamente.

Propiedad	Utilidad
DefaultGroupDeadband	En la aplicación se utiliza para establecer el valor de la propiedad DeadBand de un grupo recién creado, sin necesidad de que el usuario lo establezca directamente.
DefaultGroupLocaleID	En la aplicación es utilizada para establecer el valor de la propiedad LocaleID de un grupo recién creado, sin necesidad de que el usuario lo establezca directamente.
DefaultGroupTimeBias	Se utiliza para establecer el valor de la propiedad TimeBias de un grupo recién creado, sin necesidad de que el usuario lo establezca directamente.
Count	Es útil como parámetro de comparación en los bucles que tengan que ver con el conteo de grupos.

Los métodos reutilizados del objeto OPCGroups son los siguientes:

Método	Utilidad
Item	Es útil para referenciar los elementos de la colección en base a un índice. En la aplicación estas referencias suelen aparecer al momento de llamar a un objeto OPCItem.
Add	Único método de la interfaz que permite agregar grupos a la colección del objeto OPCServer. Sin embargo, en las clases de la aplicación actual este método es reutilizado y se optimiza.
Remove	Al contrario del método anterior, este método no es el único para remover grupos del objeto OPCServer; sin embargo si es el único que hace la remoción grupo por grupo. En la aplicación sí es el único método de remoción.

X.4. UTILIDADES DEL OBJETO OPCGROUP EN LA APLICACIÓN

El objeto OPCGroup es de suma utilidad para las aplicaciones cliente, ya que permite establecer un orden en los ítems configurados en el objeto OPCServer. Al

igual que los otros objetos, cuenta con miembros poderosos; sin embargo, por las características de la aplicación, sólo fueron utilizadas las algunas propiedades. No se reutilizaron métodos ni eventos.

Las propiedades reutilizadas son las siguientes:

Propiedad	**Utilidad**
`Parent`	Esta propiedad facilita bastante la referencia al servidor al que pertenece un grupo. En la aplicación aparece siempre que sea necesario hacer referencia a un servidor a partir de un grupo.
`Name`	Le sirve al usuario para clasificar el orden de los ítems configurados.
`IsActive`	Propiedad de mucha utilidad cuando se quiere hacer buen uso del ancho de banda de comunicación entre cliente-servidor, ya que permite activar o desactivar todo un conjunto de ítems de una vez.
`DeadBand`	Puede ahorrar ancho de banda.
`UpdateRate`	Puede ahorrar ancho de banda.
`OPCItems`	Propiedad básica del `Grupo`. De mucha utilidad, ya que es la colección a la cual pertenecerán todos los ítems que se creen en el cliente.

X.5. UTILIDADES DEL OBJETO `OPCITEMS` EN LA APLICACIÓN

En la aplicación, los miembros de este objeto se agrupan junto con los miembros del objeto `OPCGroup`, en un solo objeto llamado `GrupoOPC`.

Las propiedades reutilizadas de este objeto son las siguientes:

Propiedad	Utilidad
`Parent`	Esta propiedad facilita bastante la referencia al grupo al que pertenece un ítem. En la aplicación aparece siempre que sea necesario hacer referencia a un grupo a partir de un ítem.
`DefaultRequestedDataType`	En la aplicación es utilizada para establecer el valor de la propiedad `RequestedDataType` de un ítem recién creado, sin necesidad de que el usuario lo establezca directamente.
`DefaultIsActive`	Se utiliza para establecer el valor de la propiedad `IsActive` de un ítem recién creado, sin necesidad de que el usuario lo establezca directamente.
`Count`	Es útil como parámetro de comparación en los bucles que tengan que ver con el conteo de ítems.

Los métodos reutilizados de este objeto son los siguientes:

Método	Utilidad
`Item`	Es útil para referenciar los elementos de la colección en base a un índice.
`AddItem`	No es el único método para agregar elementos a la colección. Pero sí es le único que agrega elementos de uno en uno. En la aplicación, es este el método utilizado para agregar ítems. Sin embargo ha sido optimizado.
`Remove`	Único método de la interfaz que sirve para remover ítems de esta colección.

X.6. UTILIDADES DEL OBJETO `OPCITEM` EN LA APLICACIÓN

En la aplicación el objeto `OPCItem` es el objeto que se ha optimizado en mayor medida.

Es en este objeto en donde la naturaleza del almacenamiento de los valores y la generación de reportes tienen sentido, así como también la escala de los valores. Sin embargo muchas propiedades y métodos son reutilizadas.

Las propiedades reutilizadas son las siguientes:

Propiedad	**Utilidad**
`Parent`	Esta propiedad facilita bastante la referencia al grupo al que pertenece un ítem. En la aplicación aparece siempre que sea necesario hacer referencia a un grupo a partir de un ítem.
`ServerHandle`	Muchos métodos piden este valor como parámetro. Es un índice asignado por el servidor.
`ItemID`	Es la referencia directa de una conexión real entre cliente-servidor.
`IsActive`	Es de utilidad al usuario para asignarle un estado al ítem.
`RequestedDataType`	Siempre que se defina un ítem debe asignársele un tipo de dato. Con la reutilización de esta propiedad, el usuario puede definirlo.
`Value`	Es la propiedad que tiene el valor leído del servidor.
`Quality`	Es de utilidad para que el usuario pueda identificar la calidad de la comunicación entre cliente-servidor.
`TimeStamp`	Es de utilidad para que el usuario pueda identificar el momento en que se leyó un valor del servidor. También permite la consulta por tiempo a la base de datos.

Los métodos reutilizados son los siguientes:

Método	**Utilidad**
`Read`	Existen varias formas de hacer una lectura. En la aplicación se elige este método como único método para realizar lecturas.

XI. DISEÑO DE LA APLICACIÓN

XI.1. CRITERIOS DE DISEÑO

Muchos desarrolladores de clientes OPC concuerdan que la librería de clases OPC DA Automation Wrapper es tan básica y general que resultaría muy complicado realizar verdaderas aplicaciones con ella; y tienen razón. Es por ello que se hace necesario desarrollar una librería de clases propias que entregue mayor versatilidad y facilidad para el desarrollo de verdaderas aplicaciones. ‘

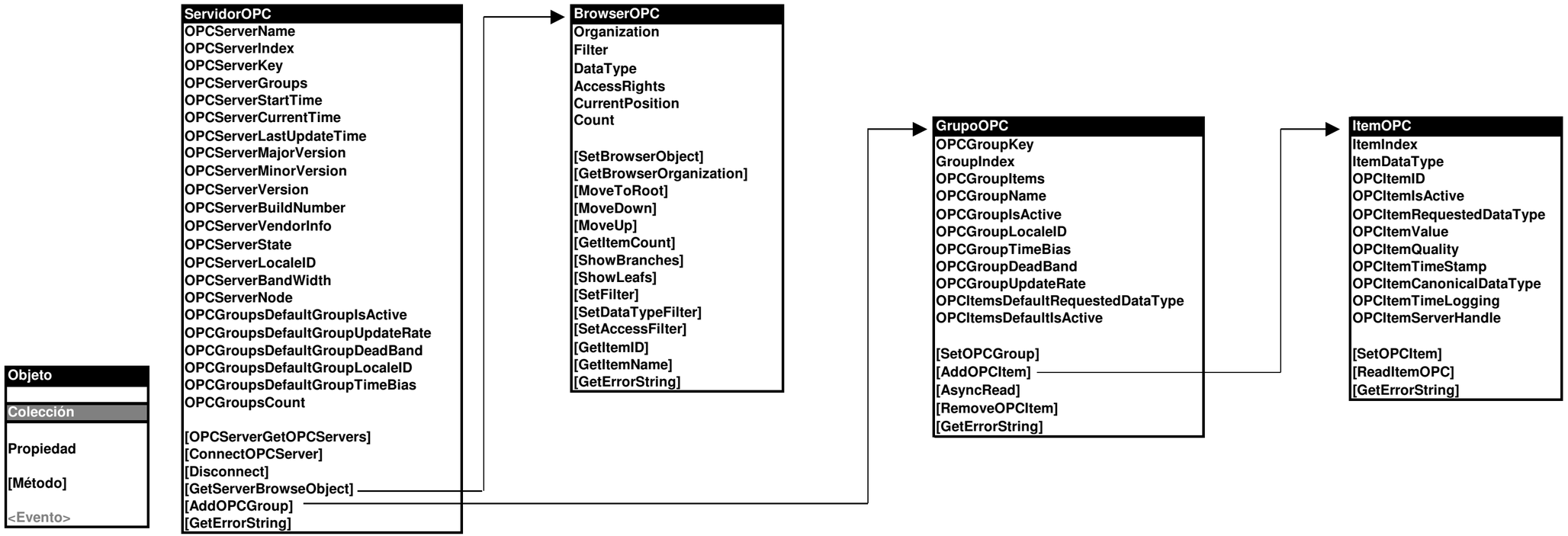
Por ejemplo, el objeto `itemOPC` no cuenta con propiedades y métodos útiles para el almacenamiento en una base de datos, y se hace muy complicado brindarle a la aplicación las características de un historiador sólo programando en el nivel de la aplicación.

También se hacía difícil hacer referencia a los métodos de las colecciones dentro de la aplicación, de la manera en que la especificación está definida.

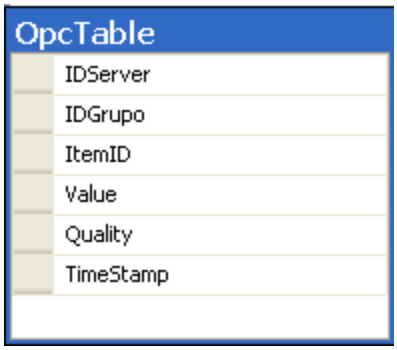
Esas y muchas razones más nos llevaron a crear nuestras cuatro clases: `ServidorOPC`, `GrupoOPC`, `ItemOPC` y `BrowserOPC`, que suprimen deficiencias del estándar y al mismo tiempo reflejan nuestro diseño.

En cuanto al almacenamiento de datos, en primer lugar, se creó una base de datos debido a la enorme cantidad de información que puede significar la aplicación. En segundo lugar, en la base de datos diseñada se toma mucho en cuenta el fácil acceso, tanto para almacenar valores como para leer valores.

XI.1.1. MODELO DE OBJETOS DE LA APLICACIÓN



XI.1.2. ARQUITECTURA DE LA BASE DE DATOS DE LA APLICACIÓN



Nombre de archivos:

- Nombre de la base de datos: `Opc`
- Nombre del archivo de datos creado: `Opc_Data.MDF`
- Nombre del archivo de registro creado: `Opc_Log.LDF`

Descripción de campos:

Columna	Descripción	Tipo
`IDServer`	Permite identificar a que servidor pertenece un ítem especifico	Carácter
`IDGrupo`	Permite identificar un ítem de un grupo en particular	Carácter
`ItemID`	Nombre del ítem configurado para almacenarse en la base de datos	Carácter
`Value`	Representa el valor del ítem	Real
`Quality`	Representa la confiabilidad del ítem	Carácter
`TimeStamp`	Es el tiempo en el cual el cliente a leído el dato del servidor	DataTime

A continuación se ilustra un ejemplo con valores reales almacenados en cada uno de los campos de la base de datos.

IDServer **=** `KEPware.KEPServerEx.V4`

`IDGrupo = Group0`

`IDItem = Channel_0_User_Defined.Sine.Sine3`

```
Value        = -0.9980261

Quality      = Good

TimeStamp    = 29/06/2006 11:38:23 p.m.
```

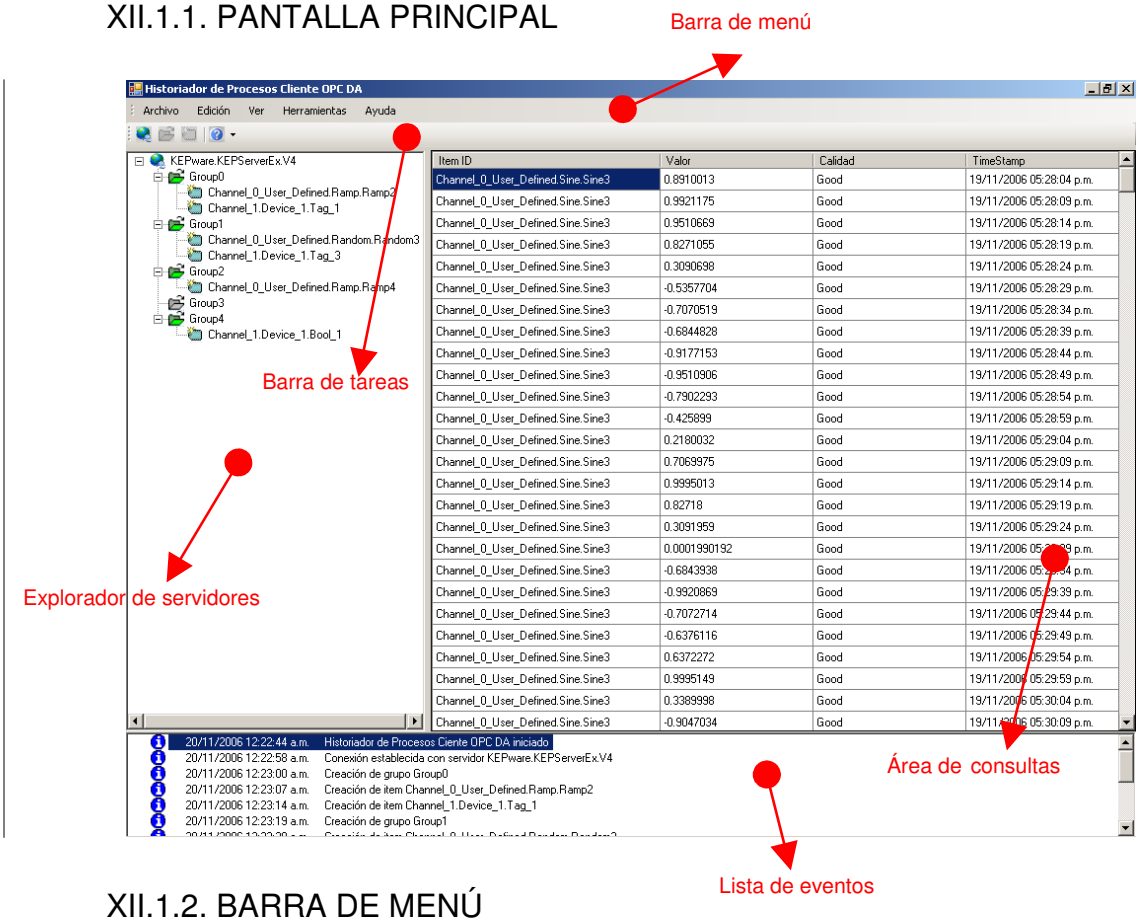
Es necesario recalcar que las bases de datos de la aplicación no cuentan con una llave primaria, lo que limita la rapidez con la que se puede acceder a un registro en particular. El hecho por el cual no se ha establecido una llave primaria radica en que no existe un campo que identifique específicamente un registro de la base de datos. En teoría se podría utilizar como llave primaria los campos `IDServer`, `IDGrupo`, e `ItemID`, ya que son éstos los utilizados para acceder a registros deseados en las consultas efectuadas a la base de datos.

XII. DISEÑO DE LA INTERFAZ GRÁFICA DE USUARIO

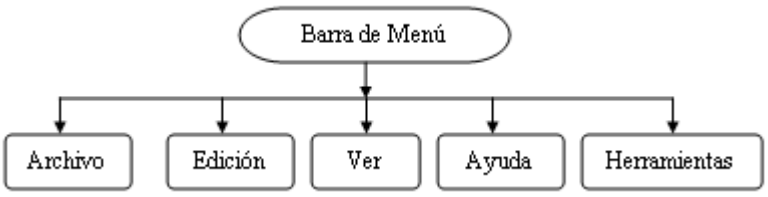
XII.1. DESCRIPCIÓN DE LA APLICACIÓN

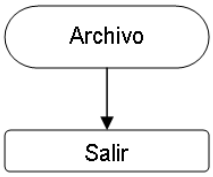
La finalidad de la interfaz gráfica de usuario es hacer fácil e intuitivo el manejo de la aplicación.

XII.1.1. PANTALLA PRINCIPAL

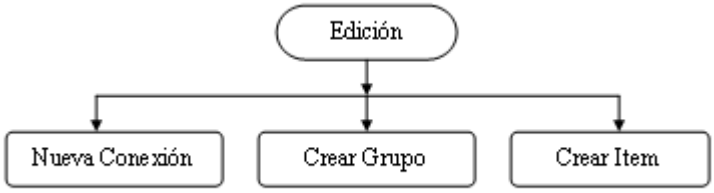


XII.1.2. BARRA DE MENÚ

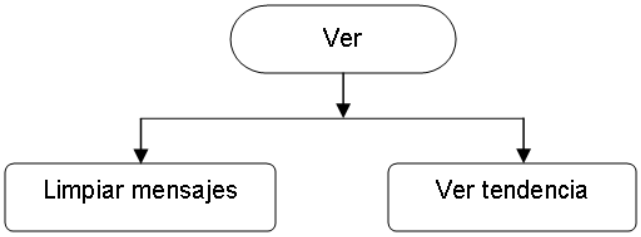




- Salir: Cierra la aplicación cliente.

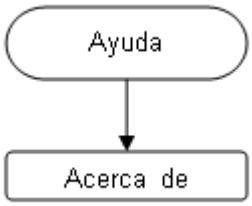


- Nuevo servidor: Establece una nueva conexión con un servidor.
- Crear grupo: Crea un nuevo objeto grupo.
- Crear ítem: Crea un nuevo objeto ítem.



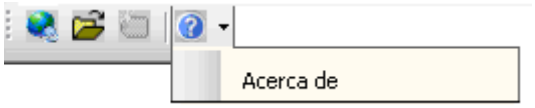
- Limpiar mensajes: Limpia la lista de eventos.
- Ver tendencia: Muestra las tendencias actuales sobre la consulta y la configuración de reporte existente.

- Crear consulta: Abre el cuadro de diálogo para realizar una consulta a la base de datos.
- Crear reporte instantáneo: Permite configurar un reporte instantáneo de la consulta actual.
- Crear Tendencia: Permite efectuar tendencias de tags que son previamente configurados.
- Exportar a Excel: Muestra un asistente que permite exportar la consulta actual hacia Microsoft Excel.

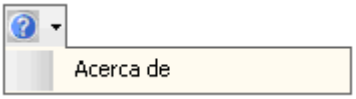


- Acerca de: Muestra el nombre de la aplicación, la versión y el nombre de los creadores.

XII.1.3. BARRA DE HERRAMIENTAS

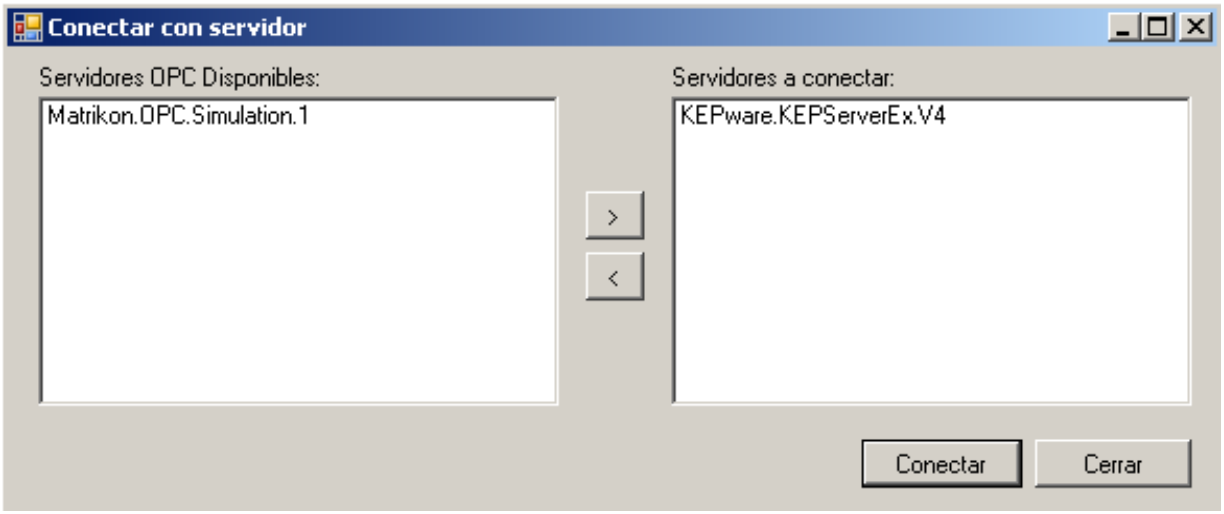


- Conectar con Servidor: Establece una nueva conexión con un servidor OPC
- Agregar Grupo: Permite agregar un nuevo grupo a un servidor.
- Agregar Ítem: Permite agregar un nuevo `Item` a un grupo.



- Ayuda - Acerca de: Muestra el nombre de la aplicación, la versión y el nombre de los creadores.

XII.1.4. FORMULARIO CONECTAR CON SERVIDOR

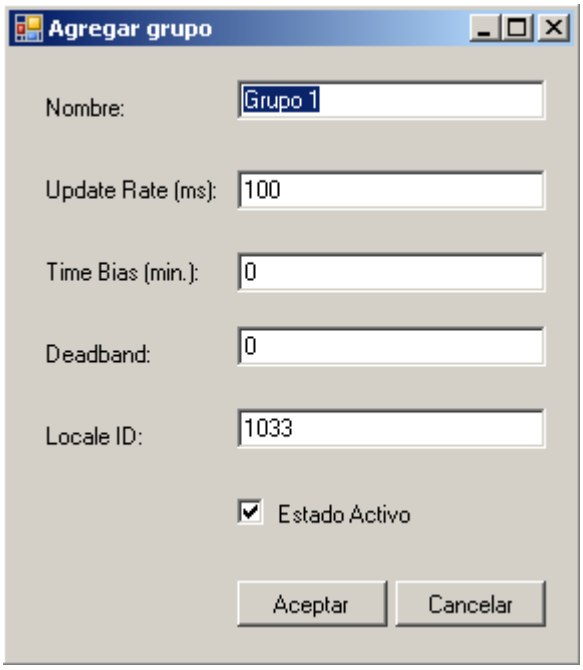


Este formulario se utiliza para escoger los servidores a los que el cliente se conectará. Cuenta con dos listas, la primera muestra los servidores disponibles en la computadora, y la segunda muestra los servidores escogidos para ser conectados.

En la lista Servidores OPC Disponibles se muestran todos los servidores instalados en la computadora. Para incluir un servidor a la lista de servidores a conectar se debe hacer click sobre él y luego presionar el botón central “>”. También puede incluirse un servidor haciendo doble click sobre él.

En la lista Servidores a conectar se muestran los servidores que han sido seleccionados por el usuario para realizar una conexión. Si se desea eliminar un servidor de esta lista, se debe hacer click sobre él y luego presionar el botón central “<”. También puede eliminarse un servidor haciendo doble click sobre él.

XII.1.5. FORMULARIO AGREGAR GRUPO



Este formulario se utiliza para agregar un grupo, así como también para definir sus valores iniciales.

Nombre

Se debe introducir un nombre que identifique al grupo. De lo contrario el servidor asignará uno.

Update Rate

Especifica el tiempo, en segundos, en el que se actualizarán los valores de los datos de los Items que pertenecen al grupo.

Time Bias

Especifica la diferencia en minutos entre el Cliente/Servidor y el dispositivo actual. Esta información permite que el cliente convierta la propiedad Time Stamp recibida por el servidor nuevamente al tiempo del dispositivo.

Dead Band

Especifica el porcentaje en que el dato debe cambiar para que el cliente sea notificado.

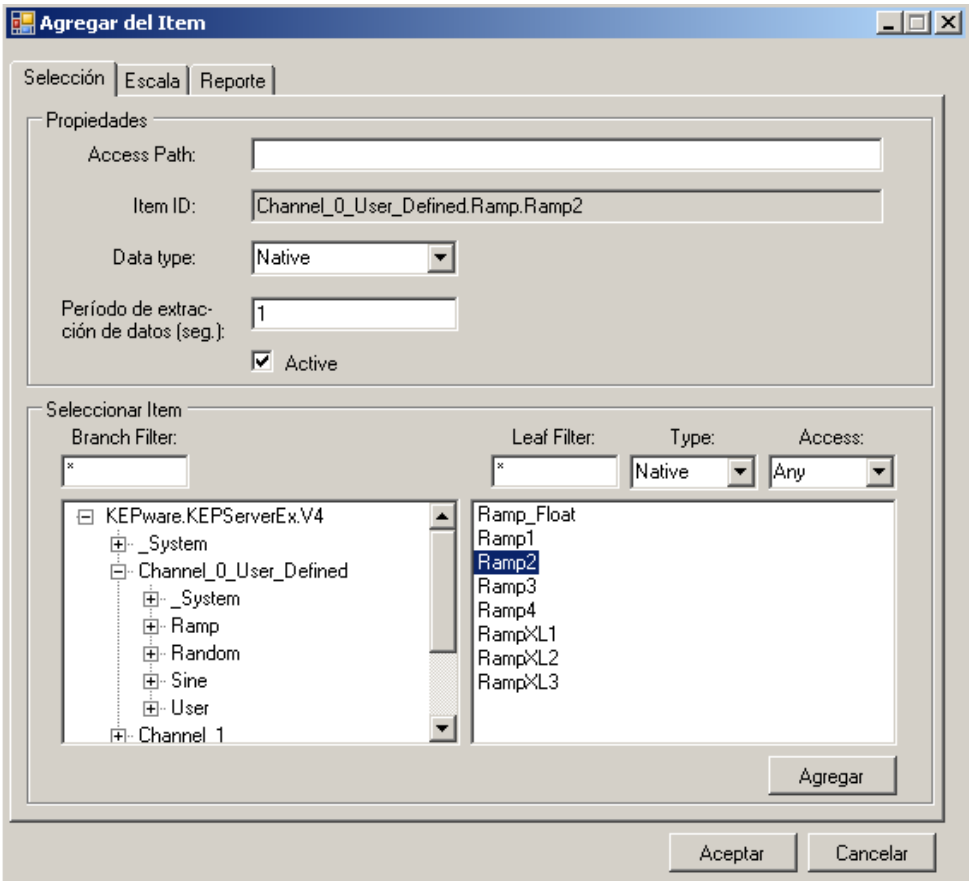
`LocaleID`

Especifica la localidad que debe ser utilizada para el retorno de cadenas de caracteres desde el servidor.

`Estado Activo`

Especifica el estado del grupo. Cuando el grupo esté activo el cliente recibirá los datos correspondientes a todos los `Items` configurados de acuerdo al `update rate`.

XII.1.6. FORMULARIO AGREGAR ÍTEM



Este formulario se utiliza para agregar un ítem, así como también introducir sus valores iniciales

Access Path

Campo requerido por algunos servidores OPC para complementar la definición del item. El usuario debe referirse a la documentación del servidor para verificar si debe llenar este campo.

Item ID

Es el item del servidor que se usa para hacer referencia a los datos. Si el servidor soporta la clase OPCBrowser, entonces pueden buscarse los ítems usando los controles del objeto.

`Data type`

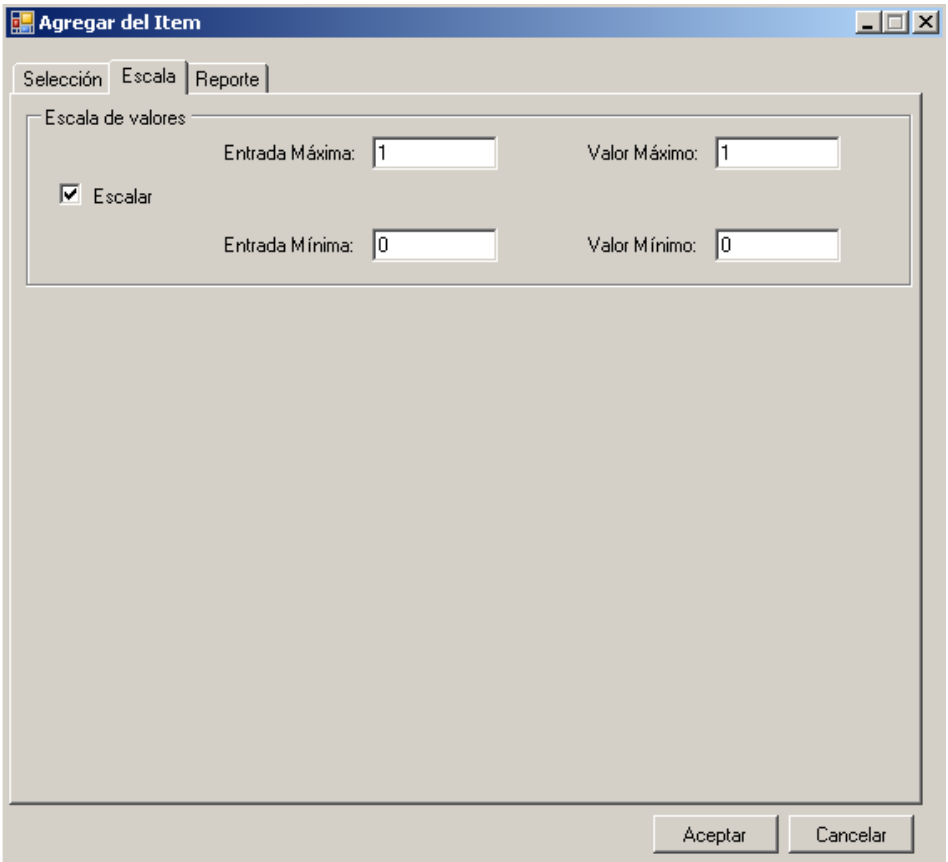
Debe ser especificado para concordar con el tamaño y el tipo de registro en memoria que es utilizado. Los tipos de datos soportados son los siguiente: `Native`, `Boolean`, `Byte`, `Short`, `Word`, `Long`, `Dword`, `Real` y `Double`.

`Período de extracción de datos`

Especifica el tiempo en segundos en el que las propiedades del `Item` se almacenarán en la base de datos.

`Active`

Especifica el estado inicial del `item`.



`Escalar`

Se activa la opción de escala de valores.

`Entrada Máxima`

Se especifica el valor máximo que se espera leer desde el dispositivo.

`Entrada Mínima`

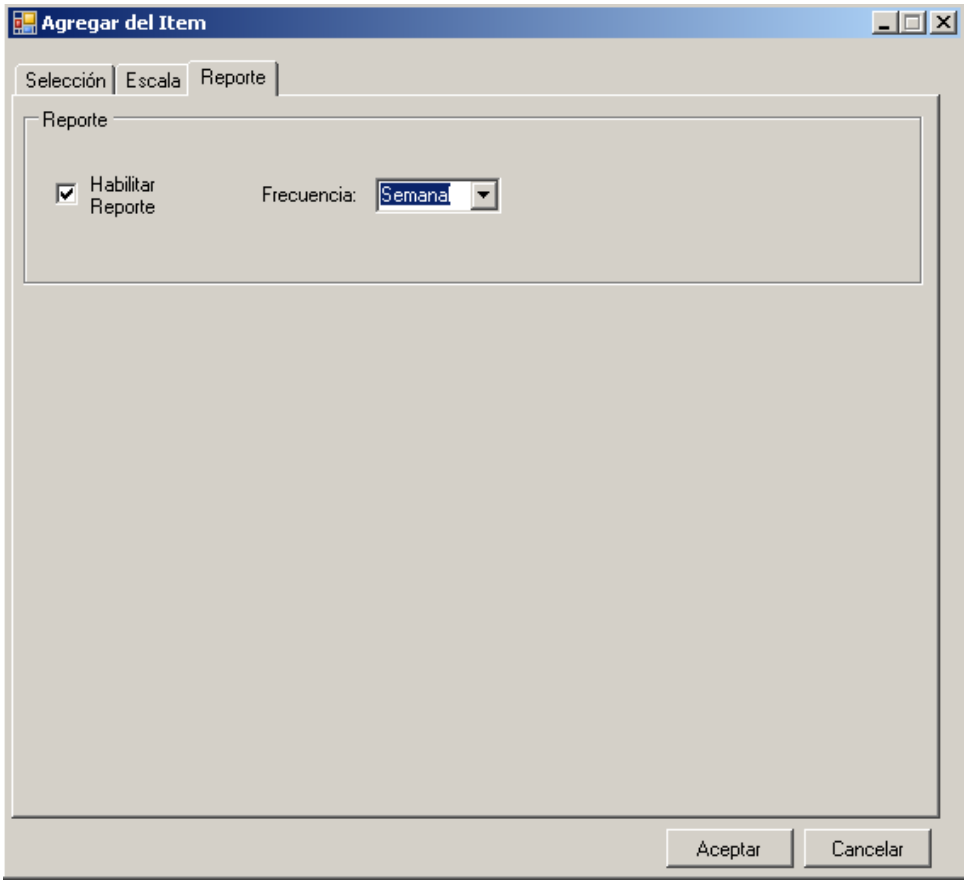
Se especifica el valor mínimo que se espera leer desde el dispositivo.

`Valor Máximo`

Se especifica el valor de escala máximo.

`Valor Mínimo`

Se especifica el valor de escala mínimo.



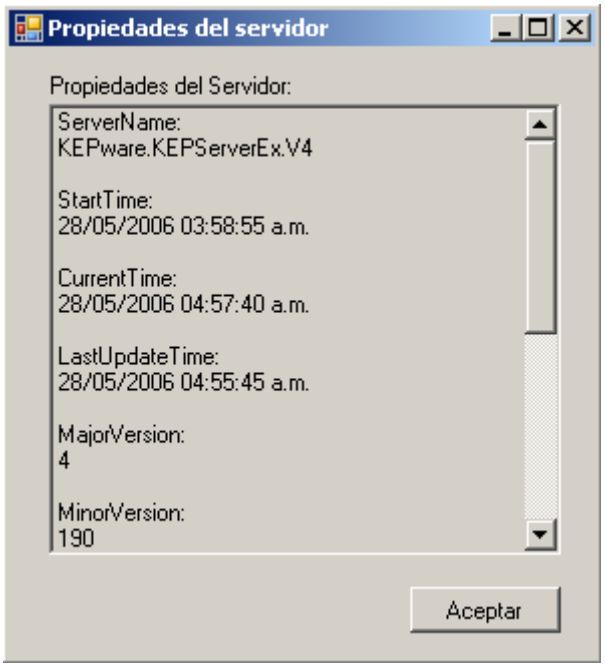
`Habilitar Reporte`

Se activa la opción de reporte programado para este `item`.

Frecuencia

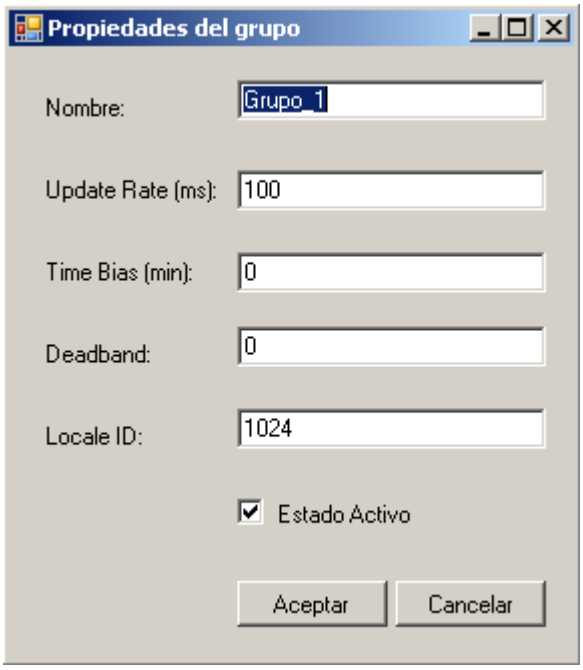
Se establece la frecuencia con la cual se generarán los reportes.

XII.1.7. FORMULARIO PROPIEDADES DEL SERVIDOR



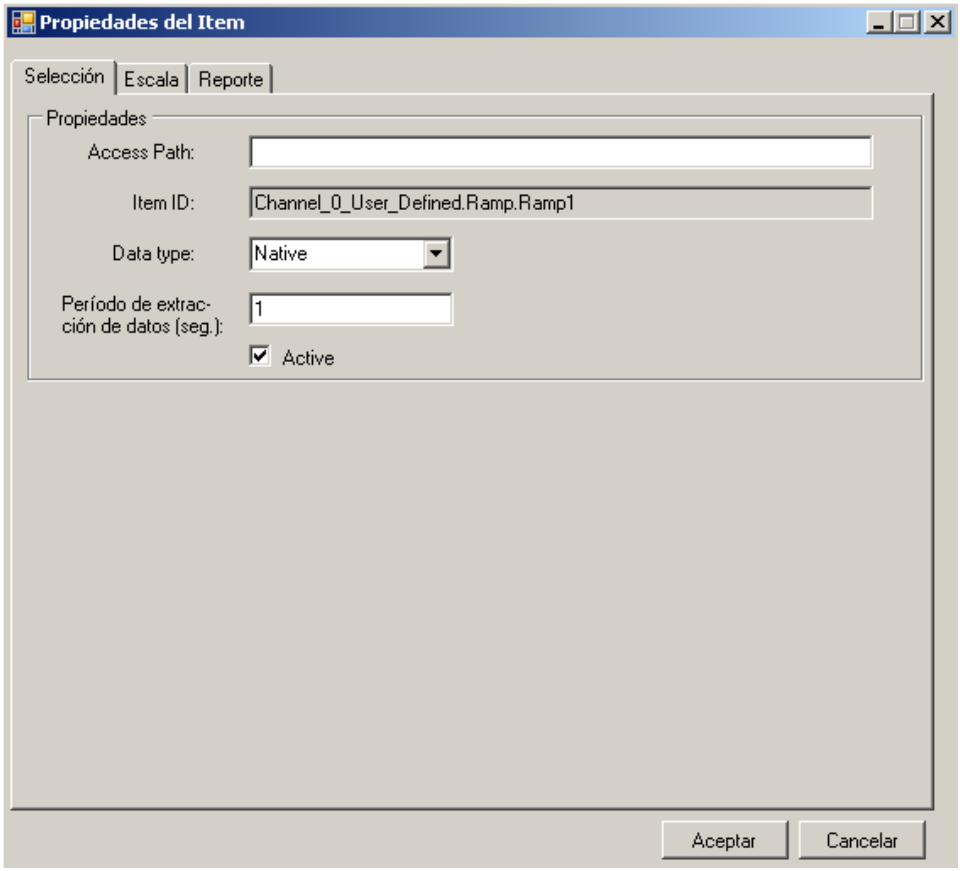
Este formulario se utiliza para visualizar las propiedades del servidor conectado. Ninguno de sus campos se puede editar.

XII.1.8. FORMULARIOFORMULARIO PROPIEDADES DEL GRUPO



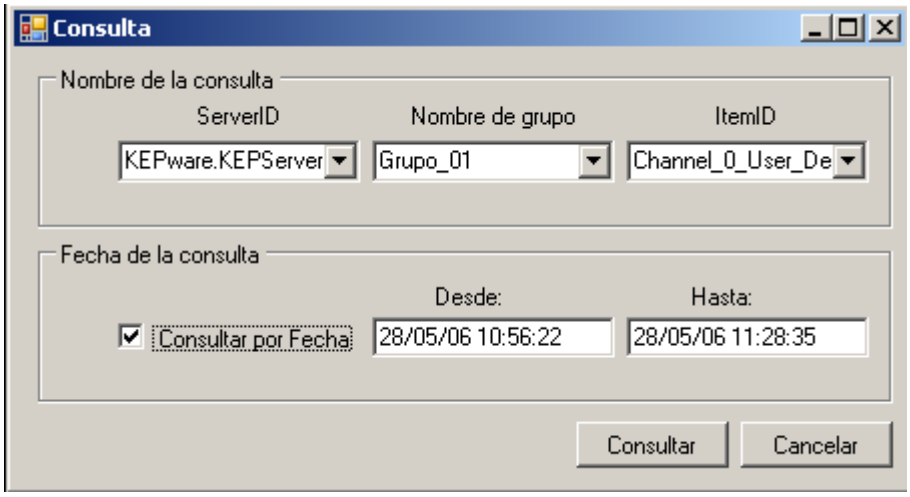
Este formulario contiene los mismos campos que el formulario `Agregar grupo` y permite modificar las propiedades del `grupo`.

XII.1.9. FORMULARIO PROPIEDADES DEL ÍTEM



Este formulario contiene los mismos campos que el formulario `Agregar Item` y permite modificar las propiedades del `Item`.

XII.1.10. FORMULARIO CONSULTA



Este formulario se utiliza para crear una consulta por nombre. Opcionalmente, se puede desarrollar una consulta por nombre y fecha.

`ServerID`

Se selecciona el servidor al cual pertenece el `Grupo` del `Item` a consultar.

`Nombre de grupo`

Se selecciona el `Grupo` al cual pertenece el `Item` a consultar.

`ItemID`

Se selecciona el `Item` a consultar.

`Consultar por fecha`

Se activa la opción de consulta por fecha.

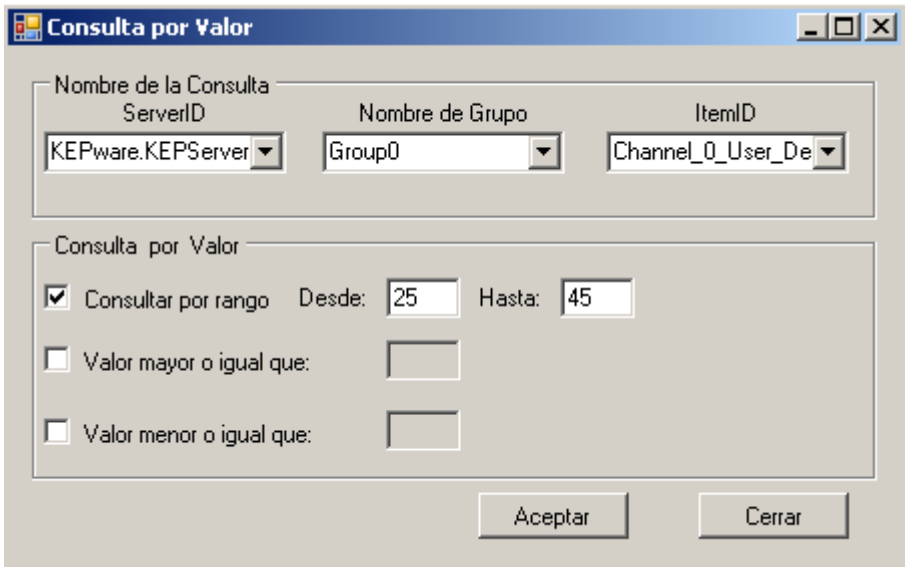
`Desde:`

Especifica la fecha inicial de la consulta.

`Hasta:`

Especifica la fecha final de la consulta.

XII.1.11. FORMULARIO CONSULTA POR VALOR



Este formulario se utiliza para crear una consulta por nombre. A diferencia del formulario anterior, en éste se seleccionan uno o dos valores como delimitadores de la consulta.

`Consultar por rango`

Habilita la opción de consultar valores que están dentro de un rango. El valor mínimo y máximo de la consulta se introducen respectivamente en los campos adyacentes.

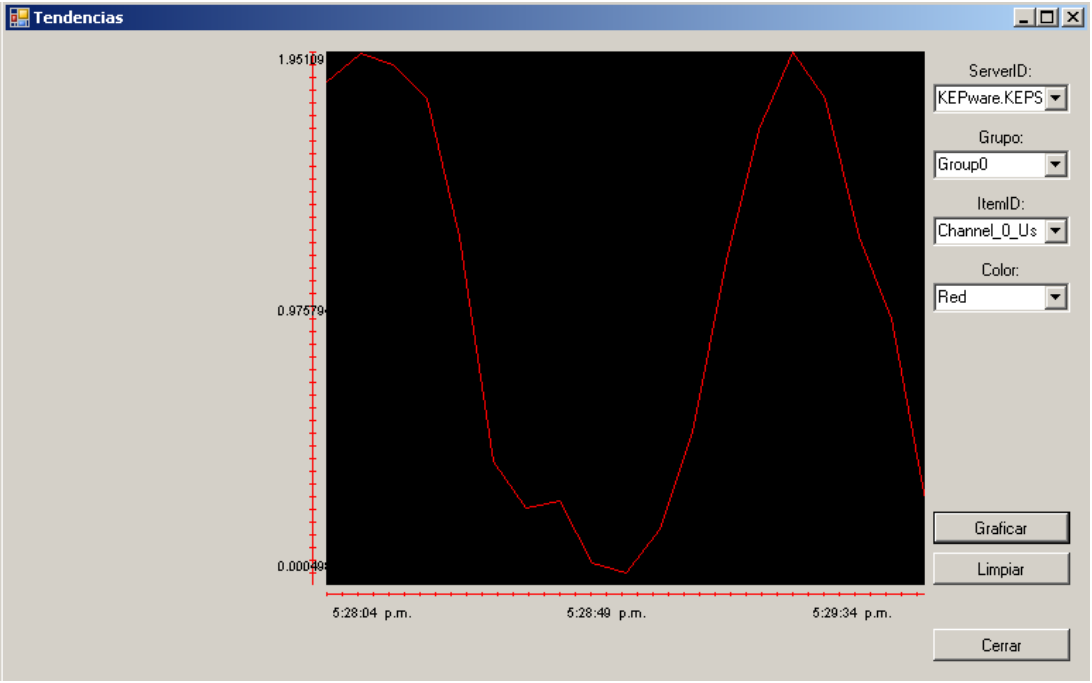
`Valor mayor o igual que:`

Habilita la opción de consultar valores mayores o iguales a una referencia, la cual se introduce en el campo adyacente.

`Valor menor o igual que:`

Habilita la opción de consultar valores menores o iguales a una referencia, la cual se introduce en el campo adyacente.

XII.1.12. FORMULARIO TENDENCIA



En este formulario se visualiza la tendencia de los valores almacenados. La gráfica se realiza en base a una consulta por nombre.

`ServerID`

Se selecciona el servidor al cual pertenece el `Grupo` del `Item` a consultar.

`Nombre de grupo`

Se selecciona el `Grupo` al cual pertenece el `Item` a consultar.

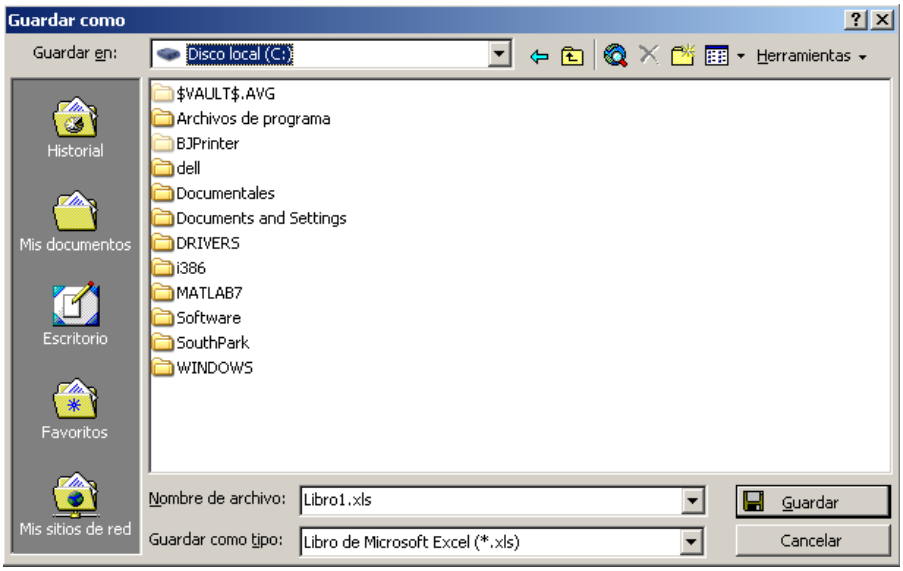
`ItemID`

Se selecciona el `Item` a consultar.

`Color`

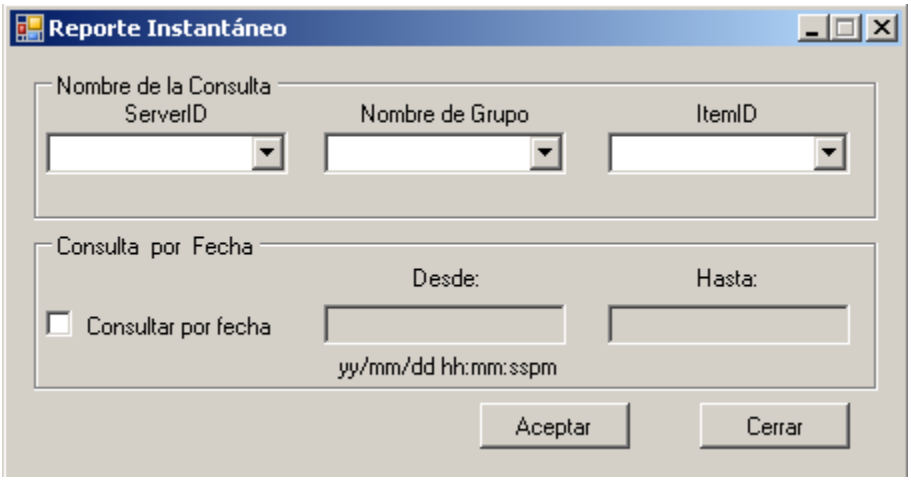
Permite seleccionar el color de la línea de la gráfica.

XII.1.13. FORMULARIO EXPORTAR CONSULTA A MICROSOFT EXCEL



En este formulario se selecciona la ubicación del archivo a exportar.

XII.1.14. FORMULARIO CONFIGURACIÓN DE REPORTE INSTANTÁNEO



Este formulario contiene los mismos campos del formulario consulta y permite crear un reporte en Excel.

XIII. LECTURA DE DATOS EN TIEMPO REAL

Realizar una pequeña aplicación de lectura de datos en tiempo real desde un servidor OPC, es un trabajo sencillo siempre y cuando se manejen los conceptos de programación orientada a objetos y se sepa manejar las propiedades básicas de los objetos del estándar.

La única conexión directa de un cliente OPC con el servidor OPC es el `Item`, sin embargo el estándar especifica que para llegar a él deben haberse creado antes los objetos `Grupo` y `Servidor`. Esto indica que en primer lugar debe crearse el objeto servidor y luego conectarse a él por medio del método respectivo. Luego debe agregarse un grupo a la colección propia del servidor. Una vez creado el grupo ya estamos listos para crear un ítem, agregarlo a la colección propia del grupo creado anteriormente, y podemos hacer con él una lectura con los respectivos métodos. Todos los agregados a partir de aquí, son limitados por el programador.

XIV. ESCRITURA DE DATOS EN TIEMPO REAL

Para poder efectuar una escritura se necesita referenciar adecuadamente los métodos de escritura tal como SynWrite y Write, los cuales están definidos en el estándar. Igualmente se deben invocar adecuadamente cada unos de los parámetros de los métodos de respectivos.

Una vez creado el `Item` por medio de los pasos indicados en el tema anterior, podemos hacer con él una escritura con los respectivos métodos. Todos los agregados a partir de aquí, son limitados por el programador.

XV. ALMACENAMIENTO EN BASES DE DATOS

En está sección se describe cual es el procedimiento utilizado para almacenar históricamente los tag configurados en el cliente que se desarrolla. Para ello de describe en primer momento el lugar donde se almacenan estos valores, lo cual corresponde a una Base de Datos, se describe sus propiedades y características, el siguiente paso es describir como acceder a esos datos. Es necesario mencionar que acceso a datos se refiere al tratamiento[18] y manipulación de los mismos desde un lugar, o una aplicación en específico, que para este caso es Visual Studio Express. Prácticamente solo se necesita utilizar la inserción de valores.

XV.1. CARACTERÍSTICAS DE LA BASE DE DATOS DEL HISTORIADOR

La Base de Datos del Historiador se llama `Opc`, cuenta únicamente con una tabla la cual tiene el nombre de `OpcTable`, una característica muy importante que vale la pena resaltar de la base de datos es que no se realiza ninguna actualización en los registros de datos, es decir únicamente se cargan inicialmente los datos y luego se accede a ellos, esas son prácticamente las operaciones realizadas. A esta peculiaridad se le llama `DataWareHouse`.

XV.1.1. CAMPOS DE LA BASE DE DATOS

Como se menciono previamente una base de datos esta formada por tablas las cuales son el destino de datos. Cada tabla esta formada a la vez por columnas, las cuales tienen un nombre y un tipo de datos específico, y por filas o registros la cual contiene un representante de cada columna.

XV.2. CONEXIÓN A BASE DE DATOS UTILIZANDO `ADO.NET`

Como ya se tiene una base de datos en SQL Server Express 2005, llamada Opc.

Ahora se deben de seleccionar los mecanismos de comunicación entre Visual Studio .Net y SQL Server Express, teniendo como puente ADO.NET.

[18] Inserción, Actualización y borrado de datos.

ADO. Net posee dos componentes esenciales que permiten el acceso y manipulación de datos, estos son:

- Proveedor de Datos de Framework .NET
- El Data Set.

El componente Utilizado para la inserción de datos es el proveedor de datos del Framework. Net. Este componente de ADO .NET esta diseñado prácticamente para la manipulación rápida de datos, generalmente se utiliza en la lectura de datos.

El Framework .Net esta conformado por varios componentes, por lo que a continuación se describen los utilizados en la conexión a la base de datos.

- Objeto `Connection`
- Objeto `Command`

Objeto `Connection`**:** Este Objeto provee un mecanismo de conectividad hacia una fuente de datos.

Objeto `Command`**:** Este objeto permite el acceso a los comandos que se encargan de modificar la base de datos, ya sea con el fin de retornar datos, modificar datos, ejecutar procedimientos almacenados, enviar o recibir información. Prácticamente el objeto `Command` permite ejecutar el lenguaje Estructurado de consultas desde una herramienta de desarrollo.

El componente Framework .Net provee una librería exclusiva para el tratamiento de datos de SQL Server la cual es `System.Data.SqlClient`, por lo que el objeto `Command` se convierte en `SqlCommand` y `Connection` en `SqlConnection`.

XV.2.1. USO DE SQLCONNECTION

SqlConnection, como ya se menciono previamente este objeto permite la conexión hacia una fuente de datos de SQL Server, por lo que su sintaxis es la siguiente:

Sqlconnection (Cadena_de_Conexión)

Donde Cadena_de_Conexión: Es una variable tipo string, en la cual se define el nombre del servidor a conectarse, la instancia SQL instalada, el nombre de la base de datos a la cual se desea conectar y, si se desea utilizar la autentificación de Windows o la de SQL Server. Para ilustrar lo antes mencionado observemos la cadena de conexión siguiente:

```
Cadena_de_Conexión="Data Source = CAMPOS\SQLEXPRESS;
Initial Catalog = Opc; Integrated Security = True".
```

De la cadena de conexión anterior se puede observar que dentro de la cadena de conexión el parámetro Data Source indica el nombre de la máquina y la instancia de SQL a la cual se desea conectar. El parámetro Initial Catalog indica la Base de Datos a la cual se desea conectar y en el parámetro Integrated Security se indica si se utiliza la autentificación de Windows o no.

XV.2.2. USO DE SQLCOMMAND

SqlCommand permite utilizar el lenguaje Transact-Sql, lo cual indica que desde Visual Basic Express se ejecutarán sentencias que normalmente se ejecutan en SQL.

Únicamente cabe mencionar que para ejecutar este comando debe de estar abierta la conexión con la base de datos y luego llamar el método ExecuteNonQuery(), el cual pertenece a la clase SqlCommand.

XV.3. INSERCIÓN DE DATOS CON ADO.NET

Como la herramienta de desarrollo utilizada para construir el Historiador de procesos es Visual Basic Express Edición 2005, se muestra a continuación el uso de ADO.NET desde esta herramienta, así como el uso de la librería system.data.sqlclient del componte FrameWork.Net.

El componente FrameWorks.NET, para poder manipular los datos, debe de estar conectado con la base de datos, ya que de otra manera los cambios realizados no tendrían efecto.

XVI. DESARROLLO DE CONSULTAS PARAMETRIZADAS

Cuando se habla de consulta en Base de Datos se esta refiriendo al hecho de consultar datos contenidos en cada una de las tablas que se encuentran en una Base de Datos determinada. Generalmente una consulta se realiza cuando se desea mostrar a un usuario final los valores de ciertos campos en específico.

En un historiador se configuran ciertas variables de un proceso para poder realizar estudios en un tiempo posterior, en un primer momento estas variables son insertadas en la base de datos. Hasta ese momento se tiene esa información almacenada. En un segundo momento es necesario ver esos valores, por lo que para poder verlos es necesario hacer consultas a la base de datos, esto con el fin de que un usuario final pueda analizar dichos datos haciendo uso de reportes, diagramas de tendencias o una exportación hacia Excel de una variable en particular.

Realizar una consulta se refiere a conectarse con una fuente de datos. En la sección anterior presentamos la forma con la que se conecto una herramienta de desarrollo con una fuente de datos RDBMS, Visual Basic Express 2005 y SQL Express 2005 respectivamente.

Como se mencionó en la sección anterior, ADO.NET posee dos componentes para acceder y manipular una fuente de datos en particular. Para la inserción se utilizo el componente de Proveedor de datos del FrameWork .Net, para el desarrollo de consultas se utilizara el otro componente el cual es el `DataSet`.

XVI.1. TIPOS DE CONSULTA

Hasta esta instancia se debe de tener una idea generalizada sobre una consulta, así como la manera de utilizar el otro componente de ADO.NET llamado “`DataSet`” las dos consultas que se realizaran a la base de datos serán:

- Consulta por Nombre
- Consulta por Nombre y fecha

XVI.1.1. CONSULTA POR NOMBRE

Al referirnos a cada unos de los campos que pertenecen a la tabla `OpcTable`, la cual pertenece a la base de datos del historiador se puede notar que son seis campos existentes, de los cuales los tres primeros se utilizan para Identificar a un Tag en particular por medio del `IdServer`, `IdGroup` e `ItemId`. El primer campo permite identificar un Tag de un Servidor, el `IdGroup` permite identificar al tag de un grupo y el `ItemId` es el nombre que lo identifica, por lo que al realizar una consulta por nombre se debe de especificar estos tres parámetros, y se Obtendrá como resultado de la consulta El nombre, Valor, la Calidad y el tiempo con el cual se ha cargado a la base de datos. Por lo que para realizar la consulta se debe abrir el formulario de consultas, el cual está ubicado en el menú herramientas.

XVI.1.2. CONSULTA POR NOMBRE Y FECHA

La consulta por nombre y fecha es similar a la consulta por Nombre con la diferencia que se agrega una condición mas que es un intervalo en al cual se desea ver un Tag, este tipo de consulta es importante cuando se desea ver el comportamiento de un Tag en un tiempo especifico, los resultados que devuelve son los mismos con la característica que pertenece a un intervalo especifico.

XVII. DIAGRAMAS DE TENDENCIA

Un diagrama de tendencias, permite ver uno o varios `ItemID`, en un plano cartesiano de dos dimensiones. Construir un diagrama de tendencia significa elaborar una grafica que indique como se ha comportado una variable con respecto al tiempo. Como variable dependiente se toma la propiedad `Value` del `Item`, y como variable independiente se toma la propiedad `TimeStamp`.

XVII.1. OBJETO GRÁFICO

Como ya se ha mencionado un diagrama de tendencia es una grafica de los valores de los ítems con respecto al tiempo. Por ello es necesario definir un objeto grafico, el lenguaje de programación que se esta utilizando para desarrollar el historiador de procesos dispone de una Variedad de objetos con los que se puede realizar operaciones graficas, como dibujar líneas, círculos, elipses entre otros, entre todos estos objetos disponibles se ha seleccionado el `PictureBox`, en la superficie de este objeto es donde se crean las curvas de las variables.

Este objeto de graficación posee algunas propiedades y métodos que se asemejan a un objeto de graficación que suelen utilizar otras herramientas de desarrollo como el objeto `Plot` usado en Matlab, para complementar el uso de este objeto se utiliza una de las librerías que trae Visual Basic la cual permite utilizar objetos para crear Líneas, Elipses, Curvas y rectángulos entre otras. Esta librería es `system.Drawing`.

Los gráficos realizados en este trabajo de graduación poseen una limitante, la cual ocurre cuando una cantidad considerable de puntos se grafican en la superficie de graficación. Esta limitante consiste en que a medida que se grafiquen más puntos en una misma zona de graficación, las líneas quedarán más unidas, dificultando la lectura de la gráfica. Esto puede haberse superado haciendo un "`zoom`", lo cual no ha sido incluido en la aplicación.

XVII.2. CREACIÓN DE GRÁFICOS

Para construir graficas en un plano bidimensional del mundo real, se necesita un conjunto de coordenadas “x” y “y”. Al unir cada uno de estos pares coordenados se forma una curva. Prácticamente para la construcción de diagramas de tendencia se ha tratado de seguir este mismo principio, por lo que para la creación de gráficos se necesita lo siguiente:

- Una Superficie de graficación
- Un conjunto de puntos.

Superficie de Graficación: Como ya se ha mencionado anteriormente se ha utilizado un objeto `PictureBox` como objeto grafico que en conjunto con la librería `system.Drawing` conforman una superficie de graficación, la cual tiene por finalidad obtener una seria de puntos para poder graficarlos en su totalidad.

Conjunto de Puntos: Como ya se conoce para poder graficar se necesita conocer una seria de puntos que al unirlos generan una curva, que muestra como se comportan en conjunto, es este caso esta información se obtiene de la base de datos, es a partir del contenido de la tabla que se obtienen dos vectores, uno de ellos es el vector "y" que correspondería al valor del `ItemID`, el restante seria el "x" el cual contiene la diferencia entre dos valores guardados.

XVII.2.1. CREACIÓN DE GRÁFICOS `UNI-TAG`

La aplicación permite la visualización de un único `ItemID` graficado en la superficie de graficación, a esto se le conoce como Diagrama `Uni-Tag`.

Para crear un gráfico, primero se debe cargar el formulario de diagramas de tendencia, el cual está ubicado en el menú "Herramientas", en la opción "Crear tendencia", una vez cargado el formulario se podrá construir los gráficos de los ítem que estén previamente configurados.

Para crear un gráfico actúan ciertas funciones las cuales se describen a continuación:

Función `ploting`: es la función principal, se encarga de crear un grafico en dos dimensiones, su sintaxis es:

```
ploting (x,y,lapiz)
```

Donde:

- X: es un arreglo unidimensional de números, los cuales corresponden al eje de tiempo.
- Y: es un arreglo unidimensional de números, que corresponden a los valores de los ítems configurados.

Esta función no devuelve ningún parámetro y se encarga de unir dos puntos adyacentes mediante líneas, con lo que es generado un gráfico. Tanto `x como y`, son previamente escalados, y se obtiene el valor máximo y mínimo de cada uno de ellos mediante la función `Max_Med_Min`, los cuales sirven para re-escalar los valores.

Función `Max Med Min`: Esta función se utiliza para determinar el valor máximo, mínimo y mediano de un arreglo unidimensional de números. Su sintaxis es:

```
Max_Med_Min (vector)
```

El argumento `vector` puede ser `x` o `y`.

Función `Eje_y`: Se encarga se graficar el eje `y` del gráfico con sus respectivas divisiones, así como con los valores representativos de cada valor. Una función equivalente existe para el eje `x`, la cual se llama `eje_x`.

Función `lapizColor`: Esta función permite seleccionar el color que tendrá el objeto `pen`, el cual pertenece a la clase `drawing` de visual Basic. De esta manera se puede cambiar el color de las líneas.

XVII.2.2. CREACIÓN DE GRÁFICOS MULTI-TAG

La aplicación permite la visualización de un conjunto de gráficos Uni-Tag graficados en la superficie de graficación, a esto se le conoce como Diagramas Multi-Tag. Esto es generado cuando se hace de manera repetitiva el grafico Uni-Tag. Para la creación de gráficos múltiples, se utilizan los mismos formularios, así como las mismas funciones. Para la creación de gráficos múltiples se debe ir creando un gráfico a la vez, de manera que al realizar repetitivamente este proceso se producen una serie de gráficas en una misma zona de graficación.

XVII.3. CREACIÓN DE REPORTES

La generación de reportes consiste en elaborar documentos que contengan información de una variable en particular. Son muchas las tareas que requieren que se estén elaborando cada cierto tiempo algunas de ellas son programadas es decir tienen una frecuencia de repetición. Existen casos que son instantáneas o inmediatas las tareas que se desean realizar. En la industria donde se desarrolla un producto resulta útil, generar reportes, esta clase de documentos ayuda a constatar el funcionamiento de un dispositivo o simplemente el comportamiento de una variable. Por citar un ejemplo, si se desea calibrar un Control PID, se puede generar un reporte de esa variable en periodos definidos de manera que se puede realizar un estudio del comportamiento bajo consideración especificas del área de trabajo

Los reportes han sido clasificados en dos categorías, los cuales son **Reportes No programados** y **Reportes Programados**. Ambos tienen el mismo obedecen al mismo formato la diferencia entre ellos se explica a continuación.

XVII.3.1. REPORTES NO PROGRAMADOS

Son aquellos reportes que son generados cuanto el usuario los necesita no tienen un periodo definido de elaboración. Para poder crear un reporte de este tipo el usuario de la Interfaz grafica decide cuando necesita elaborarla, lo único que

necesita es acceder al menú de tareas de la aplicación y seleccionar reportes no programados, a este tipo de reportes se les conoce con el nombre de **Reportes Instantáneos**. Más adelante se muestra el estilo y presentación de este tipo de servicio

XVII.3.2. REPORTES PROGRAMADOS

Son aquellos reportes que son generados en un periodo definido, el usuario define con que frecuencia serán elaborados. Para poder crear un reporte de este tipo el usuario de la Interfaz grafica debe de decidir el periodo de cuando necesita elaborar el reporte, una vez definido el periodo de elaboración del reporte este se seguirá elaborándose continuamente. Para la elaboración de este tipo de reportes se han previamente definido los posibles periodos de elaboración de reportes los cuales se detallan a continuación:

- Cada 5 minutos: Este periodo de elaboración de reportes permite crear reportes cada 5 minutos en tiempo real, este periodo ha sido seleccionado para prueba. Ya que generalmente este periodo es más largo.
- Diario: Este periodo de elaboración de reportes permite crear reportes cada 24 horas en tiempo real.
- Cada Dos Días: Este periodo de elaboración de reportes permite crear reportes cada 48 horas en tiempo real.
- Semanal: Este periodo de elaboración de reportes permite crear reportes cada 5 días en tiempo real.

De esta misma manera se ha predefinido otro tipo de reportes programados los cuales obedecen a periodos de elaboración ya predefinidos.

Para poder habilitar si desea crear un Reporte Programado al momento de crear un nuevo `ItemID` se establece si se desea crear un reporte y se define el periodo de impresión del mismo.

Este tipo de reportes son guardados en el disco duro de la máquina en una ubicación predefinida. El nombre con el se guarda el reporte obedece el siguiente formato:

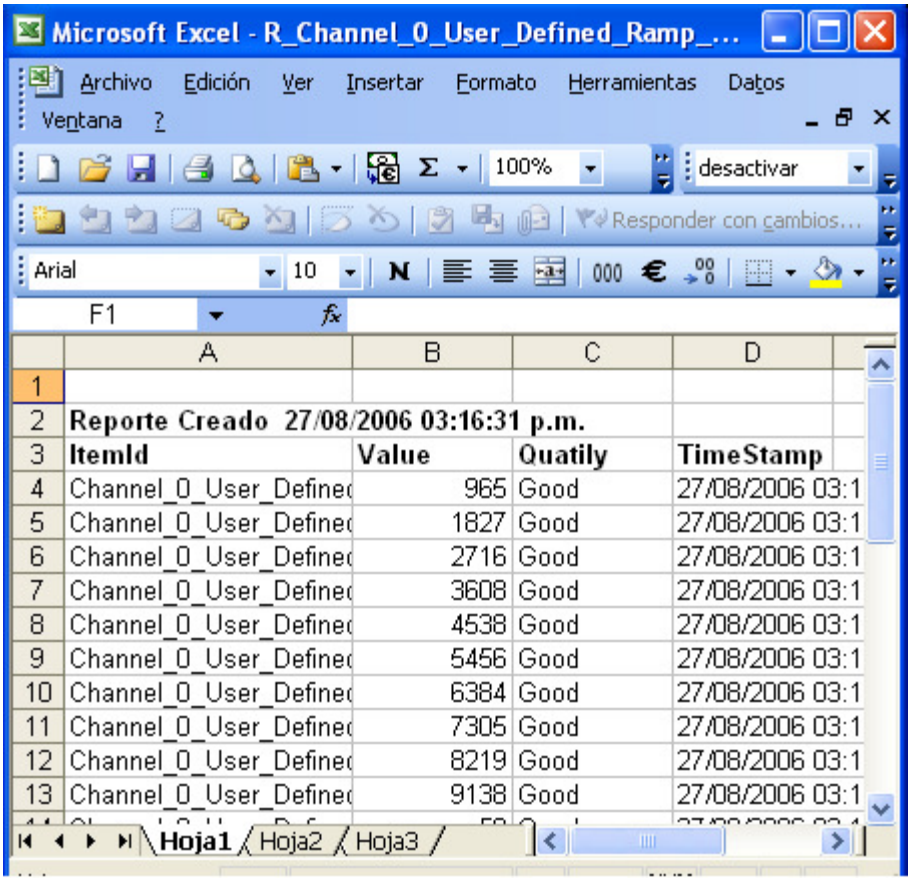
```
Nombre_delItemID_Dia_Hora
```

La ruta predefinida de la ubicación donde se almacenará el reporte es la siguiente:

```
C:\Reportes
```

De esta manera no habría confusión al momento de crear archivos posteriores.

A continuación se muestra el estilo y presentación de este tipo de servicio.



XVIII. CONCLUSIONES

- El presente trabajo de graduación demuestra que las herramientas proporcionadas por la Fundación OPC son suficientes para el desarrollo de clientes OPC DA.

- La librería “OPC Automation Wrapper” resulta ser básica para la elaboración de aplicaciones de almacenamiento histórico, ya que en sus definiciones de objetos no cuenta con funciones útiles para este propósito.

- La aplicación al cumplir con la especificación OPC DA puede comunicarse con cualquier Servidor OPC DA que cumpla con la misma especificación o con una superior a ella.

- Si bien las herramientas proporcionadas por la Fundación OPC son suficientes para el desarrollo de aplicaciones clientes OPC DA, se necesita tener conocimientos de programación orientada a objetos y bases de datos para poder desarrollar una aplicación de almacenamiento histórico.

- La aplicación elaborada muestra una manera económica y práctica para el desarrollo de aplicaciones basadas en estándares. Económica porque requiere pocos costos para su desarrollo en comparación con la adquisición de software de fabricantes reconocidos, y práctica porque cuenta con los mecanismos básicos y necesarios para realizar su labor.

- La aplicación desarrollada permitirá el estudio de procesos automatizados mediante la visualización de valores históricos de la planta, gráficas de tendencia y la exportación de datos hacia Excel.

- El estudio de esta aplicación servirá para la exploración de nuevas áreas de tecnología ligadas a la programación y a la industria, de manera que se fortalezca el conocimiento local para integrar software y hardware industrial.

XIX. RECOMENDACIONES

- Se recomienda desarrollar la lectura y almacenamiento de datos de tipo “String”.
- Se recomienda implementar acercamiento y alejamiento en el diagrama de tendencias.
- Se recomienda implementar graficas en tiempo real.
- Se recomienda realizar un estudio de los recursos utilizados por el servidor al no tener implementado el método de desconexión.

XX. REFERENCIAS

Documentos:

1. OPC Task Force: “OPC Overview V 1.0”, OPC Foundation, octubre 27 de 1998.
   Disponible en:
   http://www.opcfoundation.org
2. OPC FOUNDATION: “OPC DATA ACCESS AUTOMATION SPECIFICATION”, enero 6 de 1999. Disponible para miembros de OPC Foundation en:
   http://www.opcfoundation.org

Sitios Web:

3. Sitio oficial de la Fundación OPC donde se describe el significado de OPC. Disponible en
   http://www.opcfoundation.org/Default.aspx/01_about/01_whatis.asp?MID=AboutOPC
   Consultado el 23-Nov-2005, 10:30 p.m.
4. Charte Ojeda, Francisco. Programación con Visual Basic.Net. Editorial Anaya Multimedia, Lugar de Edición: España, 743 páginas.
   i. Capitulo 20. Fundamentos de tratamientos de Datos, página 539.
   ii. Capitulo 21. Acceso de Datos con ADO.NET, página 565.
5. Sitio oficial de la Fundación OPC donde se describe la historia del OPC. Disponible en:
   http://www.opcfoundation.org/Default.aspx/01_about/01_history.asp?MID=AboutOPC
   Consultado el 23-Nov-2005, 11:00 p.m.
6. Definición de "Object Linking and Embedding". Enciclopedia libre.
   Disponible en:
   http://en.wikipedia.org/wiki/Object_linking_and_embedding
   Consultada el 25-Nov-2005, 6:00pm

XXI. GLOSARIO

- **ActiveX:** Componente de software que se puede insertar en una página Web para ofrecer una funcionalidad que no está disponible en HTML. Los controles ActiveX se pueden implementar en diferentes lenguajes de programación y deben descargarse al disco duro del computador para que los documentos que los utilizan puedan visualizarse.
- **Cliente:** Un cliente es un programa que utiliza los servicios de otro programa. El programa cliente se utiliza para contactar y obtener datos u obtener un servicio a partir de un servidor.
- **COM:** Es el modelo de objetos basados en componentes de Microsoft (*Component Object Model*), una tecnología para construir aplicaciones a partir componentes binarias de software.
- **DCOM:** Modelo distribuido de objetos y componentes, es un protocolo y método remoto de invocación de componentes COM, incluyendo controles ActiveX. DCOM es una plataforma que permite integrar componentes UNIX o MVS e interactuar con componentes existentes en ambientes Windows. DCOM está basado en estándares como Microsoft Windows DCE RPC y ha sido publicado como un protocolo abierto para Internet por Internet Engineering Task Force (IETF).
- **DCE:** *Distributed Computing Environment*. Es un tipo de RPC.
- **DCS:** *Distributed Control System*. Sistema de control originado a partir de los ordenadores centrales de control de proceso utilizados en los años 60. Estos sistemas fueron desarrollados para los procesos de flujo continuo que requerían lazos de regulación analógicos. Se trata de sistemas en tiempo real y tolerantes a fallos para aplicaciones complejas de producción por lotes. A lo largo del tiempo han ido evolucionando, pasando del uso de software y hardware propietarios a sistemas basados en sistemas operativos como UNIX y Windows NT.

- **DDE:** Dynamic Data Exchange. Es un protocolo de software muy similar a un protocolo de PLC, diseñado para transferir bytes individuales de datos entre programas diferentes ya sea en la misma computadora (DDE), o bien a través de una red (NetDDE).
- **DLL:** Es el acrónimo de *Dynamic Linking Library* (*Bibliotecas de Enlace Dinámico*), término con el que se refiere a los archivos con código ejecutable que se cargan bajo demanda del programa por parte del sistema operativo. Esta denominación se refiere a los sistemas operativos Windows siendo la extensión con la que se identifican los ficheros, aunque el concepto existe en prácticamente todos los sistemas operativos modernos.
- **Driver:** Es un programa que interactúa con un dispositivo en particular, normalmente ese dispositivo, como por ejemplo una tarjeta de video, se entrega con el driver correspondiente que está en un disquete o CD-ROM. Contiene todos los datos necesarios del dispositivo con el que se entregó para que el resto de programas sepan cómo han de utilizarlo. Normalmente, los drivers son DLL's.
- **Fieldbus:** Es un tipo de red industrial para el control distribuido en tiempo real.
- **HMI:** Paquete de software que utiliza una interfaz gráfica para permitir a un operador controlar una máquina u operación.
- **OLE:** Compendio de documentos con un marco de trabajo basado en COM. Originalmente conocidos como "object linking and embedding", OLE fue el primer nombre que adoptó la tecnología de componentes Microsoft. OLE permitió la integración de herramientas mediante la posibilidad de utilizar características inmersas de una aplicación en otra, contribuyó de manera especial en el éxito del uso de los productos Microsoft como Office. OLE es utilizado hoy en día por un amplio número de aplicaciones Windows. COM fue desarrollado para soportar OLE 2.0 y proveer facultades de integración con documentos PC.
- **OPC:** OLE *for Process Control*. OLE que trata los datos como colecciones de objetos para ser compartidos por aplicaciones que soportan las especificaciones

OLE. OPC provee extensiones para OLE para soportar el intercambio de datos en el control de procesos.

- **PLC:** *Programmable Logic Controller*. Dispositivo electrónico de propósito especial utilizado en la industria como elemento de control y monitoreo de máquinas, motores, válvulas, sensores, medidores, etc. Este dispositivo tiene características de elemento programable y la capacidad de poder conectarse a una red. Área de aplicación: automatización de industrias y el control de máquinas industriales, control de líneas de producción, bancos de pruebas.
- **RPC**: es un protocolo que permite a un programa de ordenador ejecutar código en otra máquina remota sin tener que preocuparse por las comunicaciones entre ambos. El protocolo fue propuesto inicialmente por Sun Microsystems como un gran avance sobre los sockets usados hasta el momento. De esta manera el programador no tenía que estar pendiente de las comunicaciones, estando éstas encapsuladas dentro de las RPC.
- **SCADA:** *Supervisory Control and Data Adquisition*. Un tipo de sistemas que utiliza una computadora convencional en una aplicación de control de procesos y donde un PLC desarrolla las funciones de control pero que son monitoreadas y supervisadas por una computadora. Área de aplicación: líneas de producción, energía eléctrica, petróleo y tanques de almacenamiento, tanques de abastecimiento de agua.
- **Script:** Es un tipo de programa que consiste de una serie de instrucciones que serán utilizadas por otra aplicación.
- **Servidor:** Un servidor es una aplicación que trata las peticiones de datos, el correo electrónico, la transferencia de ficheros, y otros servicios realizados por aplicaciones cliente.
- **SOAP:** Es un estándar del World Wide Web Consortium (W3C). Un protocolo ligero para el intercambio de información en un ambiente descentralizado y distribuido, es un protocolo basado en XML que consiste en tres pastes: un sobre que define un sistema de trabajo para describir qué es lo que hay en un mensaje

y cómo procesarlo; un conjunto de reglas de codificación para expresar instancias de tipos de datos definidos para aplicaciones; y una convención para representar llamadas a procedimientos remotos y respuestas.

- **Tag:** Etiqueta.
- **MVS:** Multiple Virtual Storage. Fue el sistema operativo más usado en los modelos de mainframes System/370 y System/390 de IBM. No tiene ninguna relación con VM/CMS, otro sistema operativo de IBM.
- **XML:** Extensible Markup Language. Es una forma flexible de crear formatos de información y compartir tanto el formato como los datos en la World Wide Web, intranets y otras redes.

XXII. ANEXOS

A. LA ESPECIFICACIÓN OPC DATA ACCESS

Esta especificación sirve como material de referencia para desarrolladores de servidores y clientes OPC. Trata de facilitar el desarrollo de servidores OPC en C y C++, y el desarrollo de aplicaciones cliente OPC en el lenguaje de preferencia.

La especificación OPC Data Access contiene información tanto para la interfaz "`Custom`", la cual describe las interfaces y métodos de los componentes y objetos OPC, como también información de la interfaz "`Automation`", la cual facilita el uso de Visual Basic, Delphi y otros lenguajes para relacionarse con servidores OPC.

I. LA INTERFAZ `DATA ACCESS CUSTOM`

Como se mencionó anteriormente esta parte de la especificación sirve de guía a todos aquellos desarrolladores de servidores OPC Data Access. Como en el presente trabajo se muestra el desarrollo de un cliente OPC Data Access, sólo se hará una breve explicación de esta parte del estándar.

Los objetos de uso establecido de la interfaz OPC `Custom` son los siguientes:

- `OPCServer`
- `OPCGroup`

Las interfaces y el comportamiento de dichos objetos son descritos a detalle en la especificación. Los desarrolladores de servidores OPC deben de implementar dichos objetos para proveer la funcionalidad definida en la especificación [véase XX].

La especificación también hace referencia y define el comportamiento esperado para las interfaces del estándar OLE. Interfaces que todos los servidores y clientes OPC deben implementar cuando se desarrollan para complementarse.

II. LA INTERFAZ DATA ACCESS AUTOMATION

ANTECEDENTES

La motivación principal de esta parte de la especificación es establecer un mecanismo estándar para la comunicación de numerosas fuentes de datos, inclusive dispositivos de planta o bases de datos en el centro de control. Este mecanismo consistiría en una interfaz estándar de automatización que permitiera a aplicaciones en Visual Basic, así como también a otras aplicaciones de automatización, comunicarse con las fuentes de datos mencionadas.

Las fábricas necesitan acceder a los datos desde la planta e integrarlos en sus sistemas de negocios. Deben ser capaces de utilizar herramientas como sistemas SCADA, bases de datos y hojas de cálculo, para ensamblar un sistema que cumpla con todas sus necesidades. La clave es una arquitectura de comunicación abierta y efectiva que se concentre en el acceso a datos, y no en el tipo de datos. La fundación OPC ha resuelto esta necesidad por medio del diseño y la especificación del estándar de la interfaz de automatización con la interfaz `Custom`, para facilitar el desarrollo de aplicaciones que utilizan una interfaz de automatización para acceder a los datos de la planta.

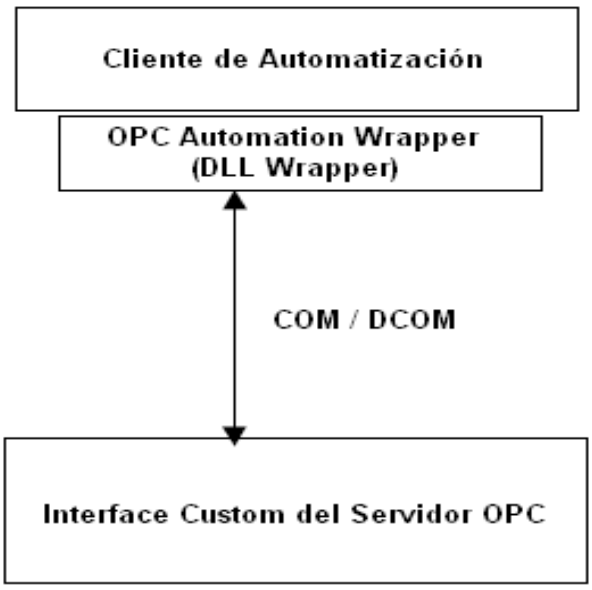
PROPÓSITO

Lo que se necesitaba era una forma común para que las aplicaciones de automatización accedieran a datos desde cualquier fuente como dispositivos y bases de datos.

La interfaz de automatización OPC define un estándar, el cual indica la manera en la que las aplicaciones de automatización pueden acceder a datos de dispositivos físicos como PLC los cuales forman parte del control de un proceso [para mayor referencia véase XX]. Esta interfaz provee la misma funcionalidad que la interfaz `Custom`, pero de una manera mas amigable para la automatización; permitiendo un uso común para acceder a otros ambientes de software, por ejemplo aplicaciones

MS Office y objetos Web. Esta interfaz ha sido diseñada para facilitar el uso de aplicaciones sin sacrificar la funcionalidad definida por la interfaz `Custom`.

La figura de abajo muestra a un cliente de automatización llamando a un servidor OPC Data Access por medio de una DLL "Wrapper"[19]. Esta librería traduce entre la interfaz `Custom` dada por el servidor y la interfaz de automatización deseada por el cliente. Note que en general la conexión entre el cliente de automatización y el servidor de automatización, llamado DLL "Wrapper", estará en proceso mientras la conexión entre el servidor de automatización y el servidor `Custom` esté también en proceso ya sea de manera local o remota.



ALCANCES DEL ESTÁNDAR

Los alcances más importantes del estándar son los siguientes:

- Hacer la interfaz más fácil de usar para los programadores de Visual Basic
- Tomar ventaja de las nuevas características de Visual Basic

[19] Librería que permite la comunicación entre dos aplicaciones servidor/cliente, sirve como interfaz de datos entre ambas.

- Permitir la creación de una DLL “Wrapper” común, la cual puede ser compartida por todos los vendedores.

ARQUITECTURA

El objetivo fundamental del diseño es que la interfaz esta dirigida a trabajar como una envoltura de la interfaz `Custom` del servidor permitiendo un mecanismo amigable de automatización con la funcionalidad proveída por la interfaz `Custom`.

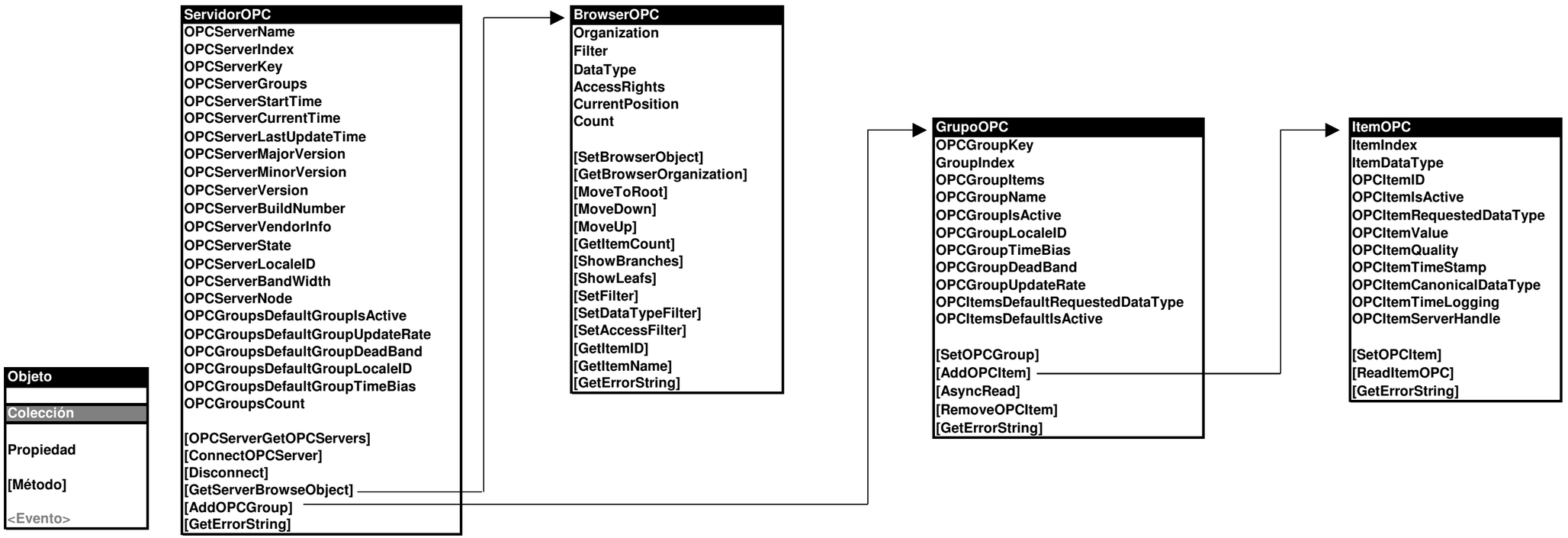
REQUERIMIENTOS FUNCIONALES

La interfaz de automatización provee casi toda la funcionalidad de las interfaces requeridas y opcionales de la interfaz `Custom`. Si el servidor Data Access `Custom` soporta una interfaz, sus funciones y propiedades a nivel de automatización funcionarán. Las interfaces de automatización generalmente no soportan las capacidades opcionales de la misma manera que lo hace la interfaz `Custom`. Si la interfaz `Custom` omite algunas funcionalidades opcionales, entonces las correspondientes funciones y propiedades en la automatización exhibirán fallas razonables.

Las interfaces son soportadas completamente por VC++ y Visual Basic 5.0. Estas permiten a cualquier aplicación que tenga una interfaz de automatización OLE acceder a las interfaces OPC, de acuerdo con las limitaciones de la respectiva aplicación.

La interfaz descrita en esta especificación no soporta VBScript o Java Script. Se puede desarrollar una “Wrapper” por separado que se acomode a las necesidades de VBScript o Java Script. Sin embargo, dicho esfuerzo esta fuera del alcance de esta especificación.

Modelo de objetos del servidor de automatización OPC



Objeto	Descripción
`OPCServer`	Una instancia de un servidor OPC. Se debe crear un objeto `OPCServer` antes de poder obtener referencias de otros objetos. Contiene la colección `OPCGroups` y permite crear los objetos `OPCBrowser`.
`OPCGroups`	Una colección de automatización que contiene a todos los objetos `OPCGroup` que el cliente ha creado en el interior del servidor al cual se ha conectado por medio del método `OPCServer.Connect()`
`OPCGroup`	Una instancia de un grupo OPC. El propósito de este objeto es mantener información del estado y proveer el mecanismo para los servicios de adquisición de datos por medio del objeto colección `OPCItems` al cual hace referencia.
`OPCItems`	Una colección de automatización que contiene a todos los objetos `OPCItem` que el cliente ha creado en el interior del servidor, y su correspondiente objeto `OPCGroup` que se ha creado en la aplicación de automatización.
`OPCItem`	Un objeto de automatización que mantiene la definición del ítem, el valor actual, la información del estado y del tiempo de actualización. Note que la interfaz `Custom` no provee un objeto `Item` por separado.
`OPCBrowser`	Un objeto que ramifica los nombres de los ítems en la configuración del servidor. Sólo puede existir una instancia de este objeto por cada instancia del objeto `OPCServer`.

La librería OPC DATA ACCESS AUTOMATION WRAPPER

La fundación OPC ha desarrollado una librería muestra completa con el código fuente en Visual C++, para proveer una interfaz de automatización para las interfaces de los servidores Data Access `Custom`. Esta librería trae definidos todos los objetos de la especificación y se instala con cualquier servidor OPC, sin embargo es probable que los desarrolladores del servidor hayan desarrollado su propia librería para cumplir con la función de ser la interfaz de automatización.

LOS OBJETOS E INTERFACES DATA ACCESS AUTOMATION

OBJETO OPCSERVER

El cliente debe crear este objeto, luego conectarlo con la interfaz Data Access Custom. Una vez creado y conectado, el cliente puede manipular la información general y la colección de objetos OPCGroup.

Sumario de propiedades

StartTime	CurrentTime	LastUpdateTime
MajorVersion	MinorVersion	BuildNumber
VendorInfo	ServerState	LocaleID
Bandwidth	OPCGroups	PublicGroupNames
ServerName	ServerNode	ClientName

Sumario de métodos

GetOPCServers	Connect	Disconnect
CreateBrowser	GetErrorString	QueryAvailableLocaleIDs
QueryAvailableProperties	GetItemProperties	LookupItemIDs

Sumario de eventos

ServerShutDown		

Propiedades más importantes

- StartTime: Es una propiedad de sólo lectura que retorna la hora en la que el servidor comenzó a correr. Si varios clientes se conectan al mismo servidor puede asegurarse que cada cliente leerá el mismo valor para esta propiedad.
- CurrentTime: Es una propiedad de sólo lectura que retorna la hora actual para el servidor.
- ServerState: Es una propiedad de sólo lectura que retorna el estado del servidor, el cual será uno de los siguientes:

Estado	Descripción
OPC_STATUS_RUNNING	El servidor esta corriendo normalmente. Es el estado usual del servidor.

Estado	Descripción
OPC_STATUS_FAILED	Un error fatal especificado por el vendedor ha ocurrido al interior del servidor.
OPC_STATUS_NOCONFIG	El servidor esta corriendo, pero no tiene cargado la información de configuración y por eso no puede funcionar normalmente. Los servidores que no requieren información de configuración no deben retornar este estado.
OPC_STATUS_SUSPENDED	El servidor ha sido suspendido temporalmente por medio de algún método especificado por el vendedor y por tanto no envía ni recibe ningún dato.
OPC_STATUS_TEST	El servidor esta en modo de pruebas

- `OPCGroups`: Es una propiedad de solo lectura. Es la colección de los objetos `OPCGroup`.
- `ServerName`: Es una propiedad de solo lectura que retorna el nombre del servidor al cual el cliente ha sido conectado por medio del método `Connect()`.
- `ServerNode`: Es una propiedad de solo lectura que retorna el nombre del nodo del servidor al cual el cliente ha sido conectado por medio del método `Connect()`.

Métodos más importantes

- `GetOPCServers`: Retorna los nombres de los servidores OPC registrados. Estos nombres son devueltos en un arreglo de cadenas.

  Esta función tiene como parámetro opcional el nombre del nodo remoto del que se desea saber la lista de servidores OPC instalados. En caso de no tener comunicación remota, no se debe especificar el parámetro opcional para que la función devuelva la lista que obtuvo de la computadora de la cual ha sido llamada.
- `Connect`: Esta subrutina debe ser llamada para establecer la conexión con el servidor OPC DA. Acepta dos parámetros; el primero, es el nombre o `ProgID` del

servidor a conectar y el segundo, es opcional y sirve para especificar el nodo con el cual se establecerá una conexión remota.

- `Disconnect`: Esta subrutina permite desconectarse del servidor. Hay que tomar en cuenta que si se desea remover el objeto `OPCServer`, la función `Disconnect()` no realiza la limpieza de los objetos creados por el servidor, esto debe hacerlo la aplicación cliente.
- `CreateBrowser`: Esta función devuelve un objeto `OPCBrowser` único para el `OPCServer`.

Eventos más importantes

- `ServerShutDown`: Este evento es activado cuando el servidor esta en proceso de apagarse y quiere avisarle a sus clientes. Los clientes tienen este evento para que cuando el servidor vaya a ser apagado, ellos puedan desconectarse de él.

Objeto `OPCBROWSER`

Este objeto es una colección de ramas y nombres de ítems que existen en el servidor. Este objeto es opcional y en algunos casos no es soportado por los servidores. Es bastante útil para la selección de los ítems a conectar, ya que estos nombres se componen del nombre del servidor y del grupo y pueden llegar a ser muy largos.

Sumario de propiedades

`Organization`	`Filter`	`DataType`
`AccessRights`	`CurrentPosition`	`Count`

Sumario de métodos

`Item`	`ShowBranches`	`ShowLeafs`
`MoveUp`	`MoveToRoot`	`MoveDown`
`MoveTo`	`GetItemID`	`GetAccessPaths`

Propiedades más importantes

- `Organization`: Propiedad de solo lectura. Retorna la organización del servidor de forma tanto jerárquica como plana.
- `Filter`: Propiedad de lectura/escritura. Aplica un filtro a los métodos de `ShowBranches()` y `ShowLeafs()`. Su valor por defecto es " ". Los servidores pueden usar este filtro para hacer más corta la lista de nombres de `Items`.
- `DataType`: Propiedad de lectura/escritura. Tiene la misma función que un filtro, a excepción que con esta propiedad se filtra con el tipo de datos.

Métodos más importantes

- `Item`: Este método es requerido por todas las colecciones. Retorna un nombre indexado por un especificado de `Item`. El nombre será el de una rama o una hoja, dependiendo de si la llamada fue a la función `ShowBranches()` o `ShowLeafs()`.
- `ShowBranches`: Llena la colección con los nombres de la ramas en la posición actual del Browser.
- `ShowLeafs`: Llena la colección con los nombres de las hojas en la posición actual del Browser. El valor por defecto para una organización plana es `FALSE`.
- `MoveUp`: Se mueve un nivel hacia arriba en el árbol.
- `MoveToRoot`: Se mueve hasta el primer nivel en el árbol.
- `MoveDown`: Se mueve un nivel hacia abajo en el árbol.
- `MoveTo`: Se mueve hacia una posición absoluta.
- `GetItemID`: Dado un nombre, retorna un `ItemID` válido que puede pasar como argumento para la función `Add ()` del objeto `OPCItems`.

OBJETO `OPCGROUPS`

El objeto `OPCGroups` es una colección de objetos `OCGroup`, y los métodos que los crean, eliminan y los administran.

Este objeto tiene además valores por defecto para el objeto `OPCGroup`.

Sumario de propiedades

`Parent`	`DefaultGroupIsActive`	`DefaultGroupUpdateRate`
`DefaultGroupDeadband`	`DefaultGroupLocaleID`	`DefaultGroupTimeBias`
`Count`		

Sumario de métodos

`Item`	`Add`	`GetOPCGroup`
`Remove`	`RemoveAll`	`ConnectPublicGroup`
`RemovePublicGroup`		

Sumario de eventos

`GlobalDataChange`		

Propiedades más importantes

- `Parent`: Propiedad de solo lectura. Retorna el objeto `OPCServer` padre del objeto `OPCGroup`.
- `DefaultGroupIsActive`: Propiedad de lectura/escritura. Esta propiedad provee el estado activo por defecto para los objetos `OPCGroup` creados con el método `Add()`.
- `DefaultGroupUpdateRate`: Propiedad de lectura/escritura. Esta propiedad provee el tiempo de actualización por defecto para los objetos `OPCGroup` creados con el método `Add()`.
- `DefaultGroupDeadband`: Propiedad de lectura/escritura. Esta propiedad provee el valor del porcentaje de banda muerta por defecto para los objetos `OPCGroup` creados con el método `Add()`.

- `Count`: Propiedad de sólo lectura requerida por todas las colecciones. Lleva el conteo de la cantidad de objetos de la colección.

Métodos más importantes

- `Item`: Este método es requerido por todas las colecciones. En este caso retorna un `OPCGroup` indexado por el especificador de `Item`. El especificador de `Item` no es más que un entero que indica la posición de determinado `Item` dentro de la colección.
- `Add`: Es una función que sirve para crear un objeto `OPCGroup` y agregarlo a la colección. Las propiedades de este nuevo grupo son determinadas por los valores por defecto que actualmente posee el servidor.
- `GetOPCGroup`: Realiza la misma función que el método `Item()`.
- `Remove`: Elimina un objeto `OPCGroup` determinado por su nombre clave.
- `RemoveAll`: Elimina todos los objetos `OPCGroup` y `OPCItems` para prepararse para el apagado del servidor.

Eventos más importantes

- `GlobalDataChange`: Este evento facilita la creación de un manejador de eventos para recibir y procesar cambios en los datos a lo largo de todos los grupos.

OBJETO `OPCGROUP`

Los objetos `OPCGroup` proveen al cliente una manera de organizar datos. Por ejemplo, el grupo debe representar `Items` de una pantalla de operación en particular o en un reporte.

Sumario de propiedades

`Parent`	`Name`	`IsPublic`
`IsActive`	`IsSuscribed`	`ClientHandle`
`ServerHandle`	`LocaleID`	`TimeBias`

Parent	Name	IsPublic
DeadBand	UpdateRate	OPCItems

Sumario de métodos

SyncRead	SyncWrite	AsyncRead
AsyncWrite	AsyncRefresh	AsyncCancel

Sumario de eventos

DataChange	AsyncReadComplete	AsyncWriteComplete
AsyncCancelComplete		

Propiedades más importantes

- `Parent`: Propiedad de solo lectura que retorna el objeto `OPCServer` padre del grupo.
- `Name`: Propiedad de lectura/escritura que representa el nombre del grupo.
- `IsActive`: Propiedad de lectura/escritura que controla el estado del grupo. Un grupo activo adquiere datos. Un grupo inactivo no continúa la adquisición de datos, a menos que sea requerido. Usando esta propiedad se configura directamente cuantos datos esta tratando de leer el servidor de los dispositivos. Cuando un grupo se coloca en `off` se puede reducir la cantidad de datos que el servidor esta leyendo del dispositivo. Esto da como beneficio un incremento en el desarrollo del servidor para adquirir los datos de lo `Items` que sí es necesario adquirir. Se recomienda usar esta propiedad, en lugar de remover `Items` continuamente.
- `IsSuscribed`: Propiedad de lectura/escritura que controla las notificaciones asíncronas del grupo. Un grupo suscrito recibe los cambios de datos desde el servidor.
- `ClientHandle`: Propiedad de lectura/escritura cuyo propósito es para que el cliente ubique rápidamente el destino de los datos.

- `LocaleID`: Propiedad de lectura/escritura que identifica la localidad que debe ser usada localizar cadenas enviadas por el servidor. El valor por defecto de esta propiedad depende del valor puesto a la colección `OPCGroups`.
- `DeadBand`: Propiedad de lectura escritura. Permite limitar la cantidad de datos enviados por el servidor, limitando el cambio de los datos enviados. Esto se logra estableciendo la razón de cambio que debe ocurrir en el valor de un ítem especificado, antes de que el servidor envíe dicho valor al cliente con el que esta conectado. Los valores de esta propiedad van desde 0 a 100 por ciento. A diferencia de las propiedades "`IsActive`" y "`UpdateRate`", esta propiedad no ayuda a la optimización del ancho de banda del servidor. `DeadBand` es una propiedad muy útil para prevenir que un historiador almacene cada pequeño cambio en el valor de un ítem debido a un ambiente ruidoso.
- `UpdateRate`: Propiedad de lectura/escritura que indica al servidor cada cuanto tiempo debe adquirir los datos del grupo. Es Introducida en milisegundos. El rango va desde 0 hasta 2147483647 milisegundos, aproximadamente 596 horas. Introduciendo un valor de 0, se le indica al servidor que los datos en este grupo se quieren adquirir tan rápido como sea posible. Esta propiedad es poderosa para optimizar la comunicación en una aplicación. Si a todos los grupos agregados a un servidor se les coloca un valor de 0, el servidor intentará adquirir todos los datos tan rápido como sea posible, lo cual es correcto si todos los datos necesitan ser actualizados de esa manera. Pero si algunos datos no cambian sino hasta los 10 minutos, leerlos así de rápido haría perder una considerable cantidad de ancho de banda.
- `OPCItems`: Es una colección de objetos `OPCItem`.

Métodos más importantes

- `SyncRead`: Esta función lee el valor, la calidad y el tiempo de estampado de uno o más ítems en el grupo.

- `AsyncRead`: Lee uno o más ítems en un grupo. El resultado es devuelto vía el evento `AsyncReadComplete` asociado al objeto `OPCGroup`.

Eventos más importantes

- `DataChange`: Este evento es accionado cuando un valor o la calidad de un valor de un ítem dentro del grupo ha cambiado. Hay que notar que el evento no se accionará más rápido que el `UpdateRate` del grupo. Por lo tanto, los valores del ítem serán almacenados por el servidor hasta que el tiempo actual + `UpdateRate` sea mayor que el tiempo del accionamiento anterior.
  Esta propiedad también es afectada por los estados tanto del grupo como de los ítems. Sólo los valores de ítems activos que pertenecen a grupos activos serán enviados al cliente en el evento.
- `AsyncReadComplete`: Este es el evento que se acciona cuando la lectura asíncrona es completada.

OBJETO `OPCITEMS`

Este objeto es una colección de objetos `OPCItem`, además tiene los valores por defecto de los mismos. Cuando un ítem es agregado, este toma por defecto los valores que son declarados en el objeto `OPCItems`. Una vez creado el ítem, se pueden cambiar estos valores. Esta metodología que asigna valores por defecto sirve para que la función `Add` no sea llamada con tantos parámetros.

Sumario de propiedades

`Parent`	`DefaultRequestedDataType`	`DefaultAccessPath`
`DefaultIsActive`	`Count`	

Sumario de métodos

`Item`	`GetOPCItem`	`AddItem`
`AddItems`	`Remove`	`Validate`
`SetActive`	`SetClientHandles`	`SetDataType`

Propiedades más importantes

- `Parent`: Propiedad de solo lectura que devuelve el padre del objeto como un objeto `OPCGroup`.
- `DefaultRequestedDataType`: Propiedad de lectura/escritura que representa el tipo de dato requerido que será usado en la llamada a la función `Add()`. El valor por defecto de esta propiedad es `VT_EMPTY`, lo cual significa que el servidor envía datos del tipo canónico.
- `DefaultIsActive`: Propiedad de lectura/escritura que define el estado que será usado en la llamada a la función `Add()`. El valor por defecto es `True`.

Métodos más importantes

- `Item`: Retorna un objeto `OPCItem` por medio de su especificador de ítem. El especificador de ítem es un índice base 1 que indica la posición dentro de la colección.
- `GetOPCItem`: Retorna un objeto `OPCItem` por medio del `ServerHandle` que regresa la función `Add()`.
- `AddItem`: Crea un nuevo objeto `OPCItem` y lo agrega a la colección. Las propiedades de este nuevo objeto son determinadas por los valores por defecto actuales de la colección `OPCItems`. Luego de que el objeto es creado y agregado, estas propiedades pueden cambiarse.
- `Remove`: Elimina un objeto `OPCItem`.
- `Validate`: Determina si uno o más objetos `OPCItem` podrían ser creados correctamente por medio del método `Add()`, pero no los crea ni agrega.
- `SetActive`: Permite la activación o desactivación de objetos `OPCItem` individuales.

OBJETO `OPCITEM`

Un objeto `OPCItem` representa una conexión con la fuente de datos al interior del servidor. Asociado a cada ítem se encuentra un valor, una calidad y un tiempo de estampado.

Sumario de propiedades

`Parent`	`ClientHandle`	`SercerHandle`
`AccessPath`	`AccessRights`	`ItemID`
`IsActive`	`RequestedDataType`	`Value`
`Quality`	`TimeStamp`	`CanonicalDataType`
`EUType`	`EUInfo`	

Sumario de métodos

`Read`	`Write`	

Propiedades más importantes

- `Parent`: Propiedad de solo lectura que hace referencia al `OPCGroup` padre.
- `ClientHandle`: Propiedad de lectura/escritura cuyo propósito es para que el cliente rápidamente localice el destino de los datos. El manejador es típicamente un índice. Este manejador será retornado al cliente por los eventos del `OPCGroup` a través de datos o cambios de estado. A diferencia del `ServerHandle`, esta propiedad puede ser cambiada a conveniencia del cliente para el manejo de los ítems.
- `ServerHandle`: Propiedad de solo lectura. Es el manejador asignado por el servidor para el `OPCItem`. Es una variable de tipo "`long`" que identifica de manera única al `OPCItem`.
- `ItemID`: Propiedad de lectura/escritura de tipo "cadena" que identifica de manera única al `OPCItem`.
- `IsActive`: Propiedad de lectura/escritura que indica el estado de la adquisición de datos para el ítem.

- `RequestedDataType`: Propiedad de lectura/escritura que indica el tipo de dato en el que el valor del ítem debe ser devuelto. Si esta propiedad no es correcta el `OPCItem` será inválido, hasta que el valor este correcto.
- `Value`: Propiedad de solo lectura que retorna el último valor leído del servidor.
- `Quality`: Propiedad de solo lectura que retorna el último valor de calidad leído desde el servidor.
- `TimeStamp`: Propiedad de solo lectura que retorna el último valor de "`TimeStamp`" leído desde el servidor.
- `CanonicalDataType`: Propiedad de solo lectura que devuelve el tipo de dato nativo en el servidor.

Métodos más importantes

- `Read`: Realiza una llamada en bloque para leer un ítem del servidor.